

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

**Automatická tvorba testov
zo stromu znalostí**

Diplomová práca

2019

Bc. Dominik Sečka

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

**Automatická tvorba testov
zo stromu znalostí**

Diplomová práca

Študijný program: Informatika
Študijný odbor: 9.2.1. Informatika
Školiace pracovisko: Katedra počítačov a informatiky (KPI)
Školiteľ: Ing. Miroslav Biňas, PhD

Košice 2019

Bc. Dominik Sečka

Názov práce: Automatická tvorba testov zo stromu znalostí

Pracovisko: Katedra počítačov a informatiky, Technická univerzita v Košiciach

Autor: Bc. Dominik Sečka

Školiteľ: Ing. Miroslav Biňas, PhD

Dátum: 26. 4. 2019

Kľúčové slová: generátor otázok, automatická tvorba testov, strom znalostí, graf znalostí, spracovanie prirodzeného jazyka, generovanie prirodzeného jazyka, e-learning, lingvistika, lexikálna analýza, morfológická anotácia, syntaktická anotácia, CoNLL formát

Abstrakt: Automatická tvorba testovacích otázok predstavuje v oblasti súčasného vzdelávania veľký posun v spôsobe hodnotenia vedomostí študentov. Z uvedeného dôvodu je obsahom tejto práce prieskum a analýza existujúcich riešení z problematiky automatického generovania faktických otázok a štruktúrovaného uchovávaní vedomostí. Hlavným cieľom práce je návrh a implementácia systému s uvedenými charakteristickými vlastnosťami a požiadavkami na automatizovanú transformáciu odborného textu na faktické otázky. Prínosom práce je existencia implementácie generátora faktických otázok prepozičného typu a škálovateľnej vedomostnej grafovej štruktúry založenej na štandardizovanom CoNLL formáte. Tento výskum podporuje myšlienku, že spracovanie prirodzeného jazyka a generovanie prirodzeného jazyka môže učiteľom účinne pomôcť pri vytváraní učebného obsahu.

Thesis title: Automatic creation of test question from knowledge tree

Department: Department of Computers and Informatics, Technical University of Košice

Author: Bc. Dominik Sečka

Supervisor: Ing. Miroslav Biňas, PhD

Date: 26. 4. 2019

Keywords: question generator, automatic creation of tests, knowledge tree, knowledge graph, natural language processing, natural language generation, e-learning, linguistics, lexical analysis, morphological annotation, syntactic annotation, CoNLL format

Abstract: Automatic creation of test questions represents a major shift in the field of assessing students' knowledge in contemporary education. For this reason, this work explores and analyses existing solutions to define principles of automatic generation of factual questions and structured knowledge retention. The main aim of the thesis is to design and implement a system with the characteristic features and requirements for automated transformation of technical text into factual questions. The benefit of this work is the existence of an implementation of a factual question generator of a prepositional type and a scalable knowledge graph structure based on standardised CoNLL format. This research supports the idea that natural language processing and natural language generation can help teachers to create learning content.

TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
Katedra počítačov a informatiky

ZADANIE
DIPLOMOVEJ PRÁCE

Študijný odbor: **Informatika**

Študijný program: **Informatika**

Názov práce:

Automatická tvorba testov zo stromu znalostí
Automatic creation of test question from knowledge tree

Študent: **Bc. Dominik Sečka**

Školiteľ: **Ing. Miroslav Biňas, PhD.**

Školiace pracovisko: **Katedra počítačov a informatiky**

Konzultant práce:

Pracovisko konzultanta:

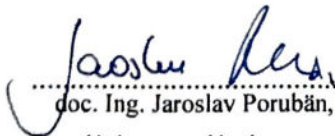
Pokyny na vypracovanie diplomovej práce:

1. Preskúmať a porovnať existujúce riešenia predmetnej problematiky.
2. Navrhnuť a implementovať systém na extrakciu testových otázok z odborných textov.
3. Overiť a zhodnotiť dosiahnuté výsledky na vybraných vzorkách textu.
4. Vypracovať dokumentáciu v typografickom systéme LaTeX.

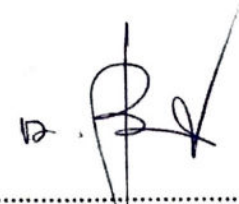
Jazyk, v ktorom sa práca vypracuje: slovenský

Termín pre odovzdanie práce: 26.04.2019

Dátum zadania diplomovej práce: 31.10.2018


doc. Ing. Jaroslav Porubán, PhD.
vedúci garantujúceho pracoviska




prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 26.4.2019

.....

Vlastnoručný podpis

Podakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce za jeho čas a odborné vedenie počas riešenia mojej záverečnej práce.

Rovnako by som sa rád poďakoval svojim rodičom a priateľom za ich podporu a povzbudzovanie počas celého môjho štúdia.

V neposlednom rade by som sa rád poďakoval pánom *Donaldovi E. Knuthovi* a *Leslie Lamportovi* za typografický systém $\text{\LaTeX}$, s ktorým som strávil množstvo nezabudnuteľných večerov.

Obsah

Motivácia	2
1 Formulácia úlohy	4
2 Súvisiaca práca	5
2.1 Automatické generovanie otázok	5
2.1.1 Automatické generovanie skúšok v jazyku R	5
2.1.2 Automatické generovanie otázok: od NLU po NLG	7
2.1.3 Systém automatickej generácie pravdivých alebo nepravdivých otázok	10
2.1.4 Automatické generovanie faktických otázok z textu	11
2.2 Vytváranie grafov znalostí	21
2.2.1 Identifikácia grafu znalostí	21
2.2.2 Freebase: Kolaboratívne vytvorená grafová databáza na štrukturovanie ľudských vedomostí	25
2.2.3 Wikidata - vedomostná základňa	30
3 Metodológia	34
3.1 Návrh riešenia	34
3.2 Metódy výskumu	36
3.2.1 Generátor otázok	37
3.2.2 Graf znalostí	41
4 Implementácia	45
4.1 Architektúra systému	45
4.2 Generátor otázok	47
4.3 NLP analyzátor	48

4.4	Grafová platforma	50
4.5	Orchestrátor	53
4.6	Používateľské rozhranie	54
5	Dosiahnuté výsledky	56
5.1	Analýza a prieskum	56
5.2	Návrh a implementácia systému	57
5.3	Overenie a zhodnotenie generovania otázok	58
5.4	Dokumentácia	59
6	Interpretácia výsledkov	60
6.1	Transformácia vstupného textu na graf	60
6.2	Vygenerované faktické otázky	61
6.3	Koreňový graf slovenského jazyka	63
6.4	Mobilná aplikácia	64
7	Ďalšia práca	65
7.1	Rozšírenie detekcie otázkových vzorov	65
7.2	Obohatenie grafovej analýzy	66
7.3	Multi-platformové používateľské rozhranie	67
8	Záver	68
	Literatúra	70
	Zoznam skratiek	77
	Slovník	79
	Zoznam príloh	83
A	Systémová príručka Generátora otázok	84
A.1	Základné informácie o systéme	84
A.2	Používané knižnice a balíčky	84
A.3	Štruktúra projektu	85
B	Používateľská príručka Generátora otázok	88
B.1	Inštalácia Neo4j databázy	88

B.2	Inštalácia aplikácie	90
B.3	Spustenie aplikácie	92
C	Systémová príručka Orchestrátora	94
C.1	Základné informácie o systéme	94
C.2	Použité knižnice a balíčky	94
C.3	Štruktúra projektu	97
D	Používateľská príručka Orchestrátora	100
D.1	Inštalácia závislostí	100
D.2	Spustenie aplikácie	100
E	Systémová príručka mobilnej aplikácie	102
E.1	Základné informácie o systéme	102
E.2	Použité knižnice a balíčky	102
E.3	Štruktúra projektu	104
F	Používateľská príručka mobilnej aplikácie	107
F.1	Inštalácia závislostí	107
F.2	Spustenie aplikácie	107
G	CD médium – záverečná práca v elektronickej podobe	108

Zoznam obrázkov

2.1	Trojstupňový rámec pre automatické generovanie otázok (Heilman, 2011).	17
2.2	Proces transformácie deklaratívnych viet na otázky (Heilman, 2011).	19
2.3	Ilustrácia príkladu, ktorý popisuje spôsob riešenia konfliktov v extrakčnom grafe (Pujara J., 2013).	23
2.4	<i>Freebase</i> - používateľské rozhranie, snímka obrazovky úpravy typovej schémy (Bollacker; Tufts a kol., 2007).	28
2.5	wikidata database caption	32
3.1	Konceptuálny návrh riešenia (vlastné spracovanie).	35
3.2	Štruktúra grafových uzlov a hrán (vlastné spracovanie).	43
4.1	Architektúra riešenia (vlastné spracovanie).	46
4.2	Štruktúra generátora otázok (vlastné spracovanie).	47
4.3	Reprezentácia vety z výpisu 3 v grafovej databáze po morfolologickej analýze (vlastné spracovanie).	51
4.4	Neo4j - databázové rozhranie (vlastné spracovanie).	52
4.5	GraphQL API - dokumentačné a dotazovacie rozhranie (vlastné spracovanie).	53
6.1	Graf vytvorený analýzou príkladu 7 (vlastné spracovanie).	61
6.2	Graf reprezentujúci komplexnejší diskurz (vlastné spracovanie).	63
6.3	Obrazovky aplikácie <i>Fairgrade</i> (vlastné spracovanie).	64
A.1	Diagram tried <i>core</i> modulu (vygenerované v prostredí <i>PyCharm</i>).	86
A.2	Diagram tried <i>database</i> modulu (vygenerované v prostredí <i>PyCharm</i>).	87

C.1	Diagram tried <i>Orchestrátora</i> (vygenerované v prostredí <i>WebStorm</i>). . .	99
E.1	Diagram tried <i>Orchestrátora</i> (vygenerované pomocou nástroja <i>WebStorm</i>).	106

Zoznam tabuliek

2.1	Rozloženie vzorov v testovacích súboroch A a B	8
2.2	Rozloženie vzorov v testovacích súboroch A a B	8
3.1	Výsledky testovania presnosti klasifikátorov textu	40
3.2	Správa o klasifikácii pre príslušne triedy	41
4.1	Presnosť značkovania segmentovaného textu	50

Zoznam zdrojových kódov

1	MQL dotaz	29
2	Odpoveď na MQL dotaz	29
3	Príklad značkovanej prepozičnej vety.	38
4	Príklad vytvorenej faktickej otázky s prepozičnou odpoveďou.	39
5	Formát tréningových dát po morfolologickej analýze.	39
6	Formát tréningových dát po identifikácii typov viet.	40
7	Príklad odborného vstupného textu.	60
8	Príklad vygenerovaných otázok.	62

Motivácia

Celosvetovo, sa podľa UNESCO, 2017 zápis do vysokoškolského vzdelávania neustále zvyšuje. Medzi rokmi 2000 až 2014 sa počet študentov vo vysokoškolských inštitúciách viac ako zdvojnásobil, pričom sa zvýšil zo 100 miliónov na 207 miliónov a existuje predpoklad, že v budúcnosti bude mať naďalej stúpajúci trend. V uvedenom štatistickom období sa rovnako celkový podiel vysokoškolského vzdelávania na celkovom vzdelávaní zvýšil z 19% na 34%. Tento pomer vyjadruje zápis ako percento obyvateľov, ktorý sú v päťročnej vekovej skupine bezprostredne po ukončení štúdia na strednej škole (zvyčajne vo veku od 19 do 23 rokov). Automatické generovanie testov prostredníctvom analýzy znalostných grafov má teda vo sfére použiteľnosti jednoznačne svoje miesto hlavne v edukačných aktivitách ľudstva. Automatizácia generovania testov dokáže zjednodušiť a uľahčiť prácu mnohým pedagógom a rovnako vytvorenie znalostných grafov s určitými hladinami dôležitosti znalostí z konkrétnych publikácií dokáže pomôcť študentom s rýchlejšou extrakciou vedomostí z požadovanej literatúry. Na základe opísaných poznatkov je možné potvrdiť potrebu existencie systému, ktorý sprostredkúva takúto funkcionality. Vo svojej podstate je teda požadovaný systém rozdelený na dve etapy a to na analýzu textu za účelom vytvárania znalostných grafov a následné generovanie otázok, ktoré dokážu overiť vedomosti študentov.

V súčasnosti existuje množstvo nástrojov, ktoré sprostredkujú automatické generovanie testovacích otázok¹. No tieto nástroje stále vyžadujú vysokú časovú náročnosť na vytváranie štruktúry otázok, a pedagógovia tak prichádzajú o čas pri tvorbe testov, ktorý by mohli v konečnom dôsledku využiť na výskum alebo zlepšovanie edukačného procesu. Študenti naopak častokrát tápajú v literatúre pri hľadaní podstatných pojmov, ktoré tvoria najdôležitejšiu vedomostnú mapu konkrétnych tém, či absolvovaných predmetov na univerzitách. Predstava automa-

¹Generovanie otázok: <https://www.capterra.com/exam-software/>

tickej extrakcie vedomostí z literatúry spolu s automatickou tvorbou unikátnych testovacích previerok napĺňa viacero akademických myslí. Potreba tejto funkcionality teda dokáže zvýšiť kvalitu vzdelávania vo viacerých smeroch. Najväčšou problematikou takéhoto systému je jednoznačne reprezentácia a uchovávanie vedomostí, ktoré vytvárajú základ a podstatu vzdelávania vo všeobecnosti.

Grafy znalostí ako štruktúry údajov tvoria základný pilier tejto práce a reprezentujú doménu znalostí. V konferenčnom príspevku Blumauer, 2018 sa uvádza, že tieto grafy majú za úlohu popísať vzťahy medzi slovami ako vecami a systematicky spájajú veci rôzneho druhu. Kódujú poznatky usporiadané v sieti uzlov a odkazov namiesto tabuliek, riadkov a stĺpcov. Prostredníctvom takýchto štruktúr môžu ľudia a stroje ťažiť z dynamicky sa rozvíjajúcej sémantickej siete faktov o veciach a môžu ich využiť na integráciu dát, objavovanie poznatkov a hĺbkové analýzy. Takéto grafy sú všade okolo a ani si to poriadne neuvedomujeme. Prestížne spoločnosti ako *Facebook*², *Microsoft*³ a *Google*⁴ prevádzkujú svoje vlastné grafy ako súčasť svojej vlastnej znalostnej štruktúry. Na povrchu, spoločnosti využívajú informácie z grafu znalostí na obohatenie výsledkov vyhľadávania a na rozšírenie umelej inteligencie spracovania požiadaviek hľadania pri odpovedaní na priame hovorené otázky v hlasových dopytoch služby *Google Asistent*⁵ a *Domov*⁶, no rovnako aj na zefektívnenie strojového učenia vyhľadávačov.

Vyššie uvedené skutočnosti vzhľadom na stúpajúci trend počtu študentov vo vyššom vzdelávaní (ale aj na nižších úrovniach vzdelávania) vytvárajú potrebu vytvorenia automatického generovania otázok testov používaných na hodnotenie vedomostí študentov prostredníctvom moderných postupov a technológií založených na dátových štruktúrach znalostných grafov, ktoré pre túto doménu vytvárajú bohatý potenciál.

²Facebook graph: <https://www.facebook.com/graphsearcher/>

³Microsoft graph: <https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/>

⁴Google knowledge graph: <https://developers.google.com/knowledge-graph/>

⁵Google asistent <https://assistant.google.com/>

⁶Google domov: https://store.google.com/gb/product/google_home

1 Formulácia úlohy

Prvou z úloh tejto záverečnej práce je vytvorenie prieskumu a porovnania existujúcich riešení pre automatickú tvorbu testov zo stromu znalostí. Druhou najpodstatnejšou úlohou je vzhľadom na uvedený prieskum a analýzu vytvorenie návrhu a implementácie systému na extrakciu testových otázok z odbornej literatúry. Rovnako dôležitou súčasťou tejto práce je aj zhodnotenie dosiahnutých výsledkov na vybraných vzorkách textu. Neoddeliteľnou súčasťou práce je taktiež aj vypracovaná dokumentácia k jednotlivým častiam systému. Doplnujúcim bodom riešenia záverečnej práce je existencia jednoduchého používateľského rozhrania, ktoré má za úlohu v jednoduchosti sprostredkovať uvedenú funkcionality. Účelom tohto posledného bodu je sprostredkovať vizualizáciu jednotlivých krokov extrakcie testových otázok z odborných textov. Riešenie je podmienené splnením základných a doplnujúcich úloh.

2 Súvisiaca práca

Kapitola súvisiaca práca konkrétne popisuje už existujúce riešenia, ktoré spadajú do domény diplomovej práce. Vedecké práce z rôznych univerzít a vedeckých inštitúcií vytvárajú podklady pre návrh riešenia diplomovej práce. Nakoľko sme nenašli štúdiu, ktorá by presne popisovala a riešila znenie zadania, bolo potrebné existujúce riešenia zatriediť do dvoch hlavných skupín. V prvej z nasledujúcich podkapitol sa nachádzajú riešenia zamerané na generovanie testovacích otázok a druhá podkapitola svojim obsahom porovnáva práce z oblasti analýzy textu za účelom vytvorenia grafov znalostí.

2.1 Automatické generovanie otázok

Generovanie otázok, ako také zastupuje koncový výsledok riešenia, nakoľko práve tento proces vytvára výstup potrebný pri skúškach, previerkach, či priebežných testovaniach. Výskumné tímy a pracoviská z rôznych kútov sveta sa zaoberali touto problematikou. Vybrali sme niekoľko zaujímavých prác, ktoré nám pomohli pri ďalšom spracovaní. Práce sme porovnali, popísali výhody a nevýhody jednotlivých konkrétnych riešení procesu generovania testovacích otázok. Výsledky porovnania tak tvoria predpoklad na správne smerovanie riešenia tohto procesu v diplomovej práci.

2.1.1 Automatické generovanie skúšok v jazyku R

Výskumníci z univerzity *Wirtschaftsuniversität vo Viedni* vytvorili prostredníctvom svojho výskumu balík s názvom *exams*¹, ktorý poskytuje rámec na automatické generovanie štandardizovaných štatistických testov, ktorý je obzvlášť užitočný pri

¹exams: <https://cran.r-project.org/web/packages/exams/index.html>

veľkých skúškach. Doménový základ pre výskumníkov boli prednášky z predmetu *Úvod do štatistiky*² na viedenskej univerzite *Wirtschaftsuniversität*, na ktorých sa každoročne zúčastňuje 1 000 až 1 500 študentov (väčšinou študenti prvého ročníka). Niekoľko lektorov z odboru štatistiky a matematiky vyučuje tento kurz paralelne. S cieľom zabezpečiť efektívnu, dôslednú a transparentnú organizáciu bol formát vyučovania a všetky jeho učebné materiály (prezentačné snímky, zbierky cvičení, skúšky atď.) prepracované v spolupráci s fakultou v priebehu rokov 2006 - 2007. Spomedzi mnohých ďalších aspektov, ako je špecifikácia zoznamu tém alebo definície výsledkov vzdelávania atď., toto prepracovanie zahŕňalo niekoľko technologických výziev. Balík *exams* bol preto navrhnutý tak, aby riešil tieto výzvy a uľahčil tak diskusiu o obsahu nového kurzu. Presnejšie definované, balíček sa zameriava na poskytovanie softvérovej štruktúry pre (Grun B., 2009):

- **Škálovateľné skúšky:** Automatické generovanie veľkého množstva rôznych skúšok s cieľom poskytnúť individuálny test každému študentovi.
- **Pridružené samo-štúdijné materiály:** Zbierky cvičení a riešenia z toho istého zoznamu príkladov.
- **Spoločný vývoj:** Vývoj a údržba veľkého množstva cvičení v prostredí viacerých autorov a viacerých platforiem.

Na použitie nástroja je potrebné, aby používatelia dodali škálu rôznych cvičení a jeden hlavný $\text{\LaTeX}$ súbor s niekoľkými prídavnými riadiacimi príkazmi, ktorý kontroluje rozloženie konečného PDF dokumentu. Cvičenia sú špecifikované v samostatných *Sweave*³ súboroch (obsahujúcich kód R pre generovanie údajov a kód $\text{\LaTeX}$ pre popis problému a riešenia). *Sweave* ako súčasť jazyka R reprezentuje výkonný a flexibilný systém na vytváranie dynamických správ a reprodukovateľného výskumu pomocou $\text{\LaTeX}$ u (RStudio, 2018).

Uvedené riešenie spĺňa funkcionality pre generovanie testov na skúšky. Prevedenie a implementácia riešenia je veľmi elegantná a uvedený balíček je možné jednoducho integrovať aj do iných systémov. Hlavným nedostatkom riešenia je ale potreba vytvárania veľkého množstva vstupných súborov obsahujúcich popis

²Úvod do štatistiky: <http://www.wu-wien.ac.at/>

³Sweave: <https://www.rdocumentation.org/packages/utils/versions/3.5.3/topics/Sweave>

znenia problému a riešenia cvičení. Tento proces tvorby zdrojových súborov na fungovanie automatických skúšok zaberá značné množstvo času a preto hodnotíme tento spôsob návrhu vstupu ako neefektívny.

2.1.2 Automatické generovanie otázok: od NLU po NLG

Vedci z Oddelenia informatiky a strojárstva z univerzity v Severnom Texase, Denton v USA (Mazidi a kol., 2016) poukázali na skutočnosť, že otázky týkajúce sa výsledkov vzdelávania a automatické vytváranie otázok výrazne uľahčujú začlenenie otázok do učebných technológií, ako sú inteligentné systémy výučby. Väčšina existujúcich systémov na generovanie otázok používa algoritmy na analýzu a transformačné algoritmy na vytvorenie otázok. Práca opisuje prístup, ktorý prináša pochopenie prirodzeného jazyka Natural Language Understanding (NLU) do procesu generovania prirodzeného jazyka Natural Language Generation (NLG).

Rus V., 2012 opisuje automatické generovanie otázok ako dialógovú a diskusnú úlohu, vykresľujúcu oba pojmy: pochopenie prirodzeného jazyka a generovanie prirodzeného jazyka. Tento opis je obzvlášť vhodný pre systém, ako je ten, ktorý je navrhnutý v tomto systéme, ktorý berie surový text ako vstup, vykonáva NLU analýzu (Allen, 1995), a potom vykoná NLG tak, že transformuje reprezentáciu na otázky v anglickom jazyku. Naopak Indurkha a kol., 2010 opisuje NLG ako proces, pomocou ktorého sa myšlienka prekladá do jazyka. Pri vytváraní otázok z textu sa vyberajú myšlienky zastúpené v deklaratívnom texte a transformujú sa do inej formy myšlienok. Vytvára sa tak takzvané spochybňovanie pôvodných, surových myšlienok. Piwek a kol., 2012 pozorovali, že generovanie otázok možno považovať za vyhľadávanie algoritmov na transformáciu vstupov na určité typy výstupov, a že kombinácie vstupov, výstupov a algoritmov sú už v tejto rozvíjajúcej sa oblasti výskumu pomerne rozdielne.

Fúzia NLG s NLU

Preskúmanie súčasného stavu techniky generovania otázok odhaľuje, že NLU je nedostatočne rozvinutým predpokladom. Typické prístupy generovania otázok, ako je popísané vyššie, analyzujú vety a potom usporadúvajú vety tak, aby transformovali vetu na toľko otázok, ako to umožňujú anglické gramatické pravidlá. V kontraste vedci v Severnom Texase definovali prístup ako najskôr klasifikovať to,

Tabuľka 2.1: Rozloženie vzorov v testovacích súboroch A a B

Vzor	Zmysel	Frekvencia
S-V-acomp	adjektívny doplnok, ktorý opisuje predmet	8 %
S-V-attr	nominálny predikatívny doplnok definujúci predmet	14 %
S-V-ccomp	klauzulačný doplnok, ktorý uvádza návrh témy	7 %
S-V-dobj	označuje vzťah medzi dvoma entitami	28 %
S-V-iobj-dobj	označuje vzťah medzi tromi entitami	< 1 %
S-V-parg	fráza opisujúca, ako/čo/kde akciu	17 %
S-V-xcomp	nefinitný klazulový doplnok	8 %
S-V	označuje činnosť subjektu	14 %
other	kombinácie zložiek	4 %

Tabuľka 2.2: Rozloženie vzorov v testovacích súboroch A a B

No	Téma	No. Vety	% Generované	Úroveň hodnotenia	% Súhlas
A1	Lymfatický systém	136	65 %	14	62 %
A2	Eukaryotické bunky	165	43 %	14	52 %
A3	Federalizmus	117	70 %	14	46 %
A4	Medzinárodný obchod	83	65 %	12	56 %
B1	Chemické väzby	138	46 %	14	54 %
B2	Plánované ekonomiky	67	50 %	15	70 %
B3	Vojna v Toledu	83	63 %	13	50 %
B4	Tornáda	151	38 %	13	48 %
	Priemer	118	53 %	14	55 %

čo veta komunikuje, analýzou typu a usporiadania syntaktických a sémantických zložiek. Vetné vzory podľa štúdie spadajú prekvapivo do malého počtu kategórií. Pre každú vetu tak systém na generovanie otázok klasifikuje vzor vety priamo pred fázou generovania otázok. Vzor vety je kľúčom k určeniu, aký typ otázky by sa mal položiť na konkrétnu vetu. Analýza v riešení bola založená na texte

extrahovanom z voľne dostupných učebníc, ako aj z *Wikipédie*. Každá textová pasáž pozostávala z textu jednej časti kapitoly alebo z textu *Wikipédie* ekvivalentnej dĺžky. Tabuľka 2.1 popisuje rozloženie vzorov vety v testovacej skupine, popisanej v tabuľke 2.2. Aby sa identifikovali vzory, ktoré majú byť zahrnuté do systému generovania otázok, boli použité nasledujúce kritériá (Mazidi a kol., 2016):

- Vyskytuje sa vzťah vety v pasážach v rôznych doménach?
- Sú sémantické informácie vety prenášané v súlade s rôznymi inštanciami?
- Určuje vektorový vzor dôležitý obsah v zdrojových vetách, aby generované otázky boli zmysluplné a nie triviálne?

Systém pre každú vetu v textovej pasáži zhromažďuje dáta z analýzy závislosti a skúmania vzťahov entít s cieľom vytvoriť prechodné zobrazenie, z ktorého sa následne môže generovať otázka. Po vytvorení objektu vety sa vzor pre každú nezávislú klauzulu každej vety porovná s približne päťdesiatimi šablónami. Ak teda šablóna zodpovedá vzoru, môže sa vytvoriť otázka.

Podľa štúdie neexistuje štandardný spôsob hodnotenia automaticky generovaných otázok. Avšak nedávne práce zaoberajúce sa generovaním otázok a aplikáciami pracujúcimi na úrovni spracovávania prirodzeného jazyka podporujú zhodnotenie pomocou *davov*, ktoré sa ukázali ako konštantné a časovo efektívne a dosiahli výsledky porovnateľné s ľudskými hodnotiteľmi (Heilman; Smith, 2010, Snow a kol., 2008). Výsledky hodnotenia generovaných otázok riešením štúdie dosahuje úroveň 55 % prijateľných otázok, čo je veľmi úspešný výsledok vzhľadom na ďalšie existujúce riešenia (Heilman, 2009), ktoré dosahovali 52 % prijateľných otázok, keď zvažili iba 20 % svojich automaticky zaradených otázok. Toto riešenie je elegantným spôsobom ako generovať otázky z akéhokoľvek vstupného textu. Za ďalšiu z výhod tohto riešenia je určite považované poukázanie na potrebu hodnotenia prijateľnosti vygenerovania otázok. Na druhú stranu za veľkú nevýhodu môže byť považované chýbajúce štrukturované uchovávanie vedomostí po analýze textov prostredníctvom NLU.

2.1.3 Systém automatickej generácie pravdivých alebo nepravdivých otázok

Keďže kladenie otázok z textu je časovo náročná a zdĺhavá úloha pri diaľkovom vzdelávaní. Vedci z *Oddelenia počítačového inžinierstva na univerzite Hallym v Kórei* navrhli automatický systém generovania otázok (Eom a kol., 2016), ktorý by viedol k pravdivým alebo nepravdivým otázkam z textu. Hlavným cieľom systému je, že po prečítaní informačnej línie z textu, ktorý je pravdivou vetou, sa vytvoria rôzne nepravdivé vety, nielen výmenou hlavnej terminológie v pravdivej vete opozitom ale aj negovaním pravdivej vety. Nakoniec je test komponovaný z náhodných pravdivých a nepravdivých otázok.

Navrhnuté riešenie sa skladá z klientskej časti, servera a databázy. Server je zostavený z dvoch návrhových častí: prvá časť sa stará o generovanie otázok a naopak druhá sa stará o výber otázok do testov. Proces tvorby pravdivých a nepravdivých otázok začína čítaním informačného riadku, ktorý je pravdivou vetou z textu. Aby bola vytvorená nepravdivá veta, algoritmus nájde v pravdivej vete kľúčové slovo a vymení ho za antonymum zo slovníka, alebo pridá do slovesa negatívnu príslovku. Na konci tohto procesu sa do databázy ukladá pravdivá i nepravdivá veta. Druhá časť následne reaguje na požiadavku vytvorenia testov z náhodnej konštalácie pravdivých a nepravdivých viet z databázy. Navrhnutý algoritmus bol implementovaný v prostredí Java Development Kit (JDK). Napríklad vstupný text *“UDP je protokol bez spojenia.”*, algoritmus vytvorí dve nepravdivé vety: *“TCP je protokol bez spojenia.”* a *“UDP nie je protokol bez spojenia.”*. Každá sada pravdivého vyhlásenia s niekoľkými nepravdivými výroky je uložená v databáze, na základe extrahovaných viet a vygenerovaných negácií, nepravdivých viet sa vytvára otázka na tieto výroky aj spolu so správnou odpoveďou.

Takto navrhnutý systém vytvára jednoduchú štruktúru na tvorbu otázok s odpoveďou *pravda* alebo *nepravda*. Návrh ilustruje naozaj jednoduché riešenie, ktoré môže podporiť návrh generovania otázok uvedeného typu. Nevýhodou však v tomto systéme je, že sa do databázy ukladajú priamo výroky a nie určité inteligentné objekty, na ktorých by sa dala vytvoriť hĺbková analýza získaných poznatkov či vedomostí.

2.1.4 Automatické generovanie faktických otázok z textu

Heilman, 2011 z *Ústavu jazykových technológií - Škola počítačovej vedy na Carnegie Mellon univerzite* predstavil obširnú prácu, ktorej cieľom bolo vytvoriť systém pre generovanie otázok (QG). Tento nástroj je teda schopný prijať text ako vstup (napríklad webovú stránku alebo encyklopédiu) a vytvoriť tak výstup ako hodnotený zoznam faktických otázok. Používateľ je následne schopný tieto otázky vyberať a revidovať za účelom vytvorenia praktického cvičenia alebo časti určitého kvízu s cieľom posúdiť, či napríklad študenti čítali požadovaný text a zachovali si vedomosti o danej téme. Navrhnutý systém sa zameriava na QG o informatívnych textoch - to znamená, že texty použité ako vstup hovoria skôr o faktických informáciách ako o názoroch. Aby bol faktický QG systém všeobecne užitočný, vyhýba sa využívaniu poznatkov špecifických pre danú oblasť (napríklad o historických udalostiach alebo geografických lokalitách) a namiesto toho sa zameriava na modelovanie pomerne všeobecných lexikálnych a syntaktických javov súvisiacich s otázkami a prezentáciou faktických informácií.

Typy otázok

Systém sa zameriava predovšetkým na QG o explicitných faktických informáciách, preto je potrebné presnejšie definovať typy otázok, ktoré bude generovať. Existuje však veľké množstvo typov otázok špecifických pre jednotlivé lexikálne konštrukcie rôznorodých jazykov a rovnako výskumníci navrhli rôzne taxonómie na ich organizáciu. Obzvlášť užitočnú a stručnú diskusiu o dimenziách, ktorými možno otázky klasifikovať, poskytuje štúdia Graesser a kol., 2008. Diskusia sa zameriava na problematiku, ako môžu byť otázky organizované podľa nasledujúcich charakteristík:

- **účel:** korekcia nedostatku vedomostí, monitorovanie spoločnej základne, sociálna koordinácia akcie, kontrola konverzácie a pozornosti
- **druh informácií, ktoré hľadajú:** súbor 16 kategórií pre otázky založené na type príslušných informácií, od jednoduchých až po zložité otázky, systém sa zameriava na generovanie dvoch typov otázok: otázky dokončenia konceptu (napr. Kto slúžil ako prvý štátny tajomník pod Georgeom Washingtonom?) a v menšej miere aj otázky overovania (napr. Bol štátny tajomník Thomas Jefferson?)

- **zdroje informácií:** táto práca si kladie za cieľ vytvárať otázky, pre ktoré sú zdrojom odpovede doslovné informácie v texte - ktoré sú zamerané skôr na úroveň vety alebo odseku, než celého dokumentu
- **dĺžka očakávanej odpovede:** očakávané odpovede sú krátke, zvyčajne jedno slovo alebo krátka fráza
- **kognitívne procesy, ktoré zahŕňajú:** otázky, ktoré táto práca prináša, hodnotia najmä vyvolanie informácií a v oveľa menšej miere pochopenie, z hľadiska taxomómie vzdelávacích cieľov sa teda zameriava hlavne na úroveň „vedomostí“

Zameraním sa na explicitné faktické informácie sa obmedzuje rozsah užitočných otázok, ktoré by mohli byť vytvorené. Faktické otázky tvoria len malú časť otázok, ktoré napríklad učitelia používajú v praxi pri hodnotení. Pedagógovia vo všeobecnosti kladú otázky, aby posúdili schopnosť študentov robiť závery, ako aj ich schopnosť vykonávať určité strategické extrakcie vedomostí, ako sú napríklad zhrnutia alebo prepojenia (asociácie) s predchádzajúcimi vedomosťami ((US) a kol., 2000). Hlboké a plytké otázky sa môžu navzájom dopĺňať: otázky na nižšej úrovni zabezpečujú, že študenti majú základné vedomosti potrebné na zodpovedanie otázok vyššej úrovne. Otázky na vyššej úrovni vyžadujú, aby študenti manipulovali s rôznymi časťami predtým získaných informácií, aby mohli formulovať odpoveď. Tieto otázky však implicitne predpokladajú, že študenti už získali základné informácie, s ktorými sa dá manipulovať kogníciou vyššej úrovne - čo nemusí byť pravda.

Automatické generovanie faktických otázok je možné efektívne modelovať pomocou prístupu generácie a štatistického hodnotenia. Techniky, ktoré sa týkajú extrakcie zjednodušeného faktického vyhlásenia a hodnotenia otázok, poskytujú merateľné výhody oproti alternatívnym technológiám z hľadiska kvality a výnosu výstupov produkovaných systémom. Nástroje QG majú potenciál podporovať realistické vzdelávacie aplikácie a širšie tak ovplyvniť vzdelávacie technológie.

Výzvy pri automatickom generovaní otázok

Výpočtové a jazykové výzvy, ktoré vznikajú pri automatickom vytváraní faktických otázok by mali slúžiť na nasledujúce účely:

- ilustrácia, že faktický QG je náročný, mnohostranný problém
- identifikácia rozsahu, pretože nemusí riešiť všetky existujúce výzvy
- napojenie sa na iné problémy v oblasti výpočtovej lingvistiky a návrh, ako by techniky a nástroje pre riešenie týchto problémov mohli podporiť automatické QG

Výzvy QG sú rozdelené do štyroch kategórií: lexikálne výzvy, syntaktické výzvy, výzvy súvisiace s diskurzom a ďalšie výzvy súvisiace s používaním nástrojov QG v jednotlivých triedach.

Lexikálne výzvy

Sémantika odpovede na otázku ovplyvňuje formu otázky. V odpovedi a konkrétne uvádza, či ide o otázku, "Kto?", "Kde?", atď. Správna identifikácia sémantiky na vysokej úrovni si vyžaduje podstatné lexikálne znalosti. A zatiaľ čo NLP ponúka potenciálne nástroje na riešenie tohto problému - vrátane rozpoznávania identifikovaných entít (Finkel a kol., 2005) a lexikálnych zdrojov, ako tie vo *WordNet-e*⁴ (Miller a kol., 1990) - tieto nástroje sú v značnej miere náchylné na chyby. Druhým problémom v tejto problematike je, že správna identifikácia sémantického typu nie je vždy dostatočná.

Ďalšou výzvou v QG je, že v ideálnom prípade by mali otázky vykazovať určité variácie od vstupného textu. Vzniká tu teda taktiež potreba vytvárania rôznych formulácií pre konkrétnu otázku. Samozrejme toto nie je len lexikálny problém - dôležitá je aj syntaktická variácia (napr. konverzia medzi pasívnymi a aktívnymi hlasmi, nominalizácia slovies atď.). Jedným z možných spôsobov, ako dosiahnuť takúto zmenu, by bolo parafrázovanie vstupných viet v počiatočnom kroku algoritmu a potom možnosť prevedenia parafrázovaných verzií do otázok (Callison-Burch, 2008; Kok a kol., 2010). Tieto techniky však sú ale stále relatívne nové a náchylné na chyby.

Záverečný problém, ktorý je v práci diskutovaný je, že význam fráz a viet nie je vždy dobre modelovaný ako kompozičný proces - to znamená, že význam frázy nie je len jednoduchým zjednotením významu jej jednotlivých slov. Problém, ako by sme si mohli myslieť však nie je v zložení, pretože druhy reprezentácií, ktoré sú

⁴WordNet: <https://wordnet.princeton.edu/>

užitočné v QG a v NLP, často vytvárajú predpoklady zloženia. Napríklad syntaktická reprezentácia vytvorená väčšinou syntaktických analyzátorov reprezentuje každý symbol slova, ktorý je nezávislý od jeho syntaktických súrodencov, daných jeho rodičom.

Pre faktické otázky zvyčajne existuje viacero možných realizácií správnej odpovede a rovnako na druhej strane často existujú početné odpovede, ktoré vykazujú rôzne stupne nesprávnosti. Používatelia QG by pravdepodobne chceli automaticky vyhodnotiť odpovede na automaticky generované tázky. Automatické triedenie je možné vykonávať v dvoch hlavných možnostiach:

1. **Pokus o vytvorenie položky s viacerými možnosťami vytváraním rušivých faktorov (t.zn. prijateľné nesprávne odpovede)** (Mitkov a kol., 2003)
2. **Umožnenie študentom odpovedať prirodzeným jazykom a aplikovanie techník na automatické skrátenie odpovedí.** (Sukkarieh a kol., 2008; Nielsen a kol., 2008; Bailey a kol., 2008; Mohler a kol., 2009)

Nakoľko obidva tieto prístupy sú trochu ortogénne k procesu tvorby otázok, ponechávajú otázku hodnotenia odpovedí na budúcu prácu.

Syntaktické výzvy

Táto časť sa zaoberá niektorými z hlavných výziev v QG, ktoré súvisia so syntaktickou štruktúrou prirodzeného jazyka. Na generovanie otázok musia systémy QG modelovať niektoré zložitosti syntaxe prirodzeného jazyka: identifikácia frázových hraníc (napr. ktoré slová sú súčasťou odpovede), identifikácia predikátov a ich argumentov a extrakcia informácií z vnútorných štruktúr, ako sú napríklad relatívne formy. Na vytvorenie syntaktického modelu sa používajú štandardné nástroje na syntaktickú analýzu frázových štruktúr ako krok predspracovania. Dochádza tu teda k potrebe vytvorenia široko používaných syntaktických reprezentácií prostredníctvom automatickej analýzy. Bohužiaľ, parsery nie sú ani zďaleka dokonalé a výkonnosť analýzy mimo domény je zvyčajne oveľa nižšia (Foster, 2010) a keď syntaktické analyzátory robia chyby, tak ich určite budú robiť aj QG systémy. V práci sa však autor spolieha na analyzátor syntaxe a študuje, ako často nesprávne analýzy vedú k neprijateľným otázkam.

Aj keď je syntaktický analyzátor schopný poskytnúť presnú reprezentáciu syntaktickej štruktúry vety, extrahovanie informácií z tejto štruktúry alebo jej trans-

formácia sú netriviálne procesy. Zložité vety používajú rôzne syntaktické konštrukcie na prezentovanie viacerých individuálnych faktov, na ktoré by sme chceli položiť otázky. Faktické otázky týkajúce sa napríklad materiálov na čítanie sú zvyčajne krátke a zameriavajú sa na jednu informáciu. Vety v informačných textoch sú však v priemere oveľa dlhšie, pretože rôzne konštrukcie, ako sú spojky, podriadené klauzuly a apozície, umožňujú spisovateľovi sprostredkovať viac informácií v jednej vete.

Dôležitým problémom v QG je teda, ako generovať stručné otázky z komplexných viet. Jedným z mnohých riešení je použitie techník kompresie viet (Knight a kol., 2000; Dorr a kol., 2003). Cieľom kompresie je vziať ako vstup dlhú a zložitú vetu a vyprodukovať ako výstup jednu skrátenú verziu vety, ktorá prenáša hlavnú časť informácie na vstupe. Ako je však uvedené vyššie, vety často prenášajú viacero informácií. V QG však môžeme chcieť generovať otázky nielen o informáciách v hlavnej klauzule, ale aj o informáciách vložených do rôznych vnorených konštrukcií. Úloha extrahovania jednoduchých viet z komplexnej vstupnej vety je v podstate úlohou generovania určitej podmnožiny viet, ktoré by čitateľ považoval za pravdivé po prečítaní vstupu.

V iných problémoch generovania textu z iného textu, ako je kompresia viet alebo strojový preklad, je zvyčajne postačujúce generovanie jedného prijateľného výstupu. Avšak v prípade QG, kde často existuje viacero potenciálnych odpovedí vo vete pre individuálnu vstupnú vetu, existuje niekoľko odlišných výstupov - to znamená, že výstupy, ktoré nie sú len parafrázami, ale zameriavajú sa na rôzne odpovede. Nemôžeme však jednoducho vybrať každú odstatnú frázu vo vete a formulovať otázku vykonaním niekoľkých transformácií. Niekedy existujú zložité jazykové obmedzenia, ktoré ovplyvňujú, ktoré frázy vo vete môžu podstúpiť transformáciu a ignorovanie týchto obmedzení môže viesť k negramatickým otázkam. Dôležitým bodom v tejto problematike je, že neexistuje žiadny veľký korpus s otázkami, ktoré porušujú uvedené obmedzenia, a ktoré by mohli byť použité ako negatívne príklady pre tréning štatistického modelu týchto javov.

Diskurzné výzvy

Táto sekcia opisuje niektoré výzvy QG, ktoré zahŕňajú diskurz na vyššej úrovni a hlavne pragmatické javy. Je ich možné organizovať do dvoch kategórií: problémy s nejasnosťami, ktoré vyplývajú z kontextu a problémy, ktoré vyplývajú z impli-

citných diskurznych vzťahov a nedostatku vedomostí počítačov.

Texty sú zvyčajne písané ako kompletne samostatné diskurzy. Keď sú zavedené nové entity alebo koncepty, spisovatelia zvyčajne používajú veľmi špecifické opisy. Jednotlivé vety však vychádzajú z predchádzajúcich viet a spisovatelia využívajú jazykové prostriedky, ako je anafora, aby sa vyhli opakovaniu informácií. V dôsledku toho, že individuálna veta alebo klauzula môže závisieť od kontextu diskurzu, vziať fakty z jej kontextu a vytvárať otázky môže viesť k nejasným a nepríjemným otázkam.

Zámenné anafory sú jedným z jasných príkladov toho, ako môžu vznikať nejasné kontextové otázky. Otázka by mohla byť platná v určitých situáciách, napríklad by bola položená okamžite po prečítaní vety. Jednou z možností je jednoducho nezahrnúť otázky, ktoré obsahujú zámená do výstupu systému QG. Takáto možnosť sa však zdá byť trochu neuspokojivá, pretože zámená sú celkom bežným javom vo vetách. V predbežných testoch uvedeného systému dokonca zistili, že zámená sa objavujú približne v jednej tretine viet z textov z *Wikipédie* a *Wall Street* žurnálu. Ak by sa ale nezaoberali identifikáciou odpovedí, bolo by možné vygenerovať otázku, na ktorú by odpoveďou bolo zámeno. Ďalšou možnosťou je použitie automatických techník detekovania anafory a na identifikáciu predchodcov zámen a potom nahradenie týchto zámen ich konkrétnejšími predchádzajúcimi frázami.

Zámená samozrejme nie sú jediným typom anafory. Frázy obsahujúce podstatné mená taktiež môžu odkazovať na predchádzajúce informácie v diskurze, čo vedie k potenciálne nejasným otázkam. Existujú určité charakteristiky fráz s podstatnými menami, ktoré čiastočne, ale nie úplne vedú k nejasným otázkam (napríklad podstatné meno obsahuje vlastné mená). Tieto charakteristiky sú však dosť zložité a vzájomne sa ovplyvňujú. Preto namiesto kódovania takýchto charakteristík ako pravidiel ich zaraďujú ako znaky štatistickej prijateľnosti otázok, ktorý používajú na klasifikáciu otázok.

Niekedy to nie je prítomnosť slov závislých od kontextu vo vetách, ktoré vedú k nejasnostiam, ale skôr k absencii kontextuálnych informácií. Predpokladajú sa tu najmä informácie o časovom a lokačnom kontexte, ktoré sú zavedené na začiatku diskurzu. Štatistický model pre hodnotenie otázok, čiastočne rieši tento problém a pomáha systému vyhnúť sa niektorým z nejasných otázok tohto typu.

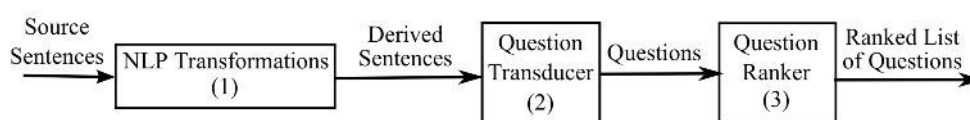
Jednou z najťažších výziev pri vytváraní otázok je to, že autori využívajú roz-

siahle diskurzové vzťahy, spoliehajúc sa na svetové vedomosti a inferenčné schopnosti čitateľov, aby im dávali zmysel. Súčasné NLP technológie, žiaľ, nemajú svetové vedomosti a inferenčné schopnosti, takže je prakticky nemožné vytvoriť množstvo užitočných otázok práve v tomto kontexte. Tento nedostatok explicitných informácií je v prirodzenom jazyku celkom bežný. Spisovatelia a rečníci často predpokladajú značné vedomosti o svete a inferenčné schopnosti. Vytvorenie automatického QG o implicitných faktických informáciách by bolo oveľa náročnejšie bez zavedenia značných znalostí špecifických pre danú oblasť.

Nie všetky otázky majú rovnakú hodnotu. Hodnota otázky závisí od konkrétneho kontextu, v ktorom sa bude používať. Napríklad učiteľ, ktorého študenti čítajú text o meste *Dublin*, by sa mohol rozhodnúť, či sa zamerajú na základné faktické informácie o meste, veľmi podobné fakty alebo rôzne iné druhy informácií. Učiteľ by sa mohol taktiež zamerať na konkrétne aspekty textu, pretože sa týkajú iných tém v jeho učebných osnovách. Možným prístupom k odhadu relevantnosti a dôležitosti informácií by mohlo byť prispôbenie NLP techník pre sumarizáciu extrakčného textu (Erkan a kol., 2004; Nenkova a kol., 2006; Nenkova a kol., 2006). Algoritmy, ako napríklad maximálny okrajový význam, by sa mohli použiť aj na zabránenie nadbytočnosti v súbore vygenerovaných otázok. Avšak dôležitosť otázky môže do veľkej miery závisieť od konkrétneho prostredia, v ktorom sa má otázka použiť.

Rámec pre generovanie vecných otázok

Po predstavení problémov generovania faktických otázok a definovaní jednotlivých výziev, ktoré s nimi súvisia, prezentuje autor v tejto práci trojstupňový rámec pre generovanie vecných otázok, tak ako je to znázornené na obrázku 2.1.



Obr. 2.1: Trojstupňový rámec pre automatické generovanie otázok (Heilman, 2011).

V prvej fáze v uvedenom systéme dochádza k transformácii vety alebo množiny viet do jednoduchších deklaratívnych výrokov zmenou alebo transformáciou lexi-

kálnych položiek, syntaktickej štruktúry alebo sémantiky. Realizácia zahŕňa operácie na extrakciu a zjednodušenie komplexných viet a na riešenie zámenných anafor. Ako rozšírenie uvádzajú možnosť prispôsobenia iných NLP transformácií, ako je napríklad komprimácia viet, fúzia viet alebo generácia parafráz.

V druhej fáze je deklaratívna veta premenená na súbor otázok vykonaním série dobre definovaných syntaktických transformácií. Je viac-menej pravdepodobné, že rôzne vety zo vstupného textu, ako aj rôzne transformácie týchto citácií povedú k vysokokvalitným otázkam, no v poslednej tretej fáze sa otázky hodnotia a radia podľa symbolov zdrojových viet, vstupných viet, otázok a transformácií, ktoré vznikli pri generovaní.

Generovanie otázok

Druhá fáza v systéme berie ako vstup deklaratívnu vetu a produkuje množinu možných otázok ako výstup. Identifikuje odpovede na nasledujúce typy fráz:

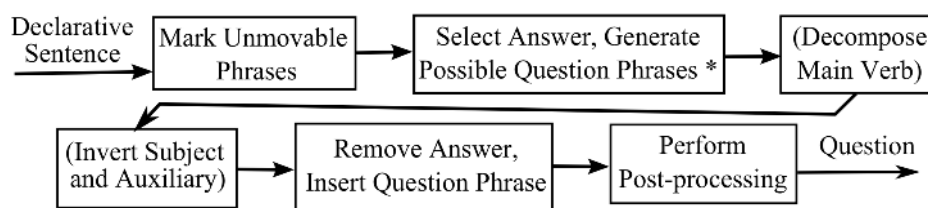
- **frázy podstatných mien** (napr. *Thomas Jefferson*)
- **predložkové frázy** (napr. *V roku 1801*)
- **podriadené doložky** (napr. *, že Thomas Jefferson bol 3. prezidentom USA*)

Systém nevytvára otázky zo všetkých možných fráz týchto typov. Existujú určité obmedzenia pre súbor možných odpovedných fráz. A rovnako systém nevytvára otázky o slovesných frázach. Z možných odpovedí, systém generuje otázky s nasledujúcimi opytovacími zámenami: kto, čo, kde, kedy, ktorého a koľko. Pre každú vetu môže byť vytvorených viacero otázok"v určitých vetách existuje niekoľko možných odpovedí a frázy otázok pre každú odpoveď. Pravidlá transformácie v 2. etape, ktoré kódujú značné množstvo jazykových znalostí, sa uplatňujú v nasledujúcom poradí:

1. Označenie frázy, ktorá by mohla predstavovať odpovedovú frázu.
2. Výber frázy odpovede a generovanie súboru fráz vhodných ako otázky.
3. Rozloženie hlavného slovesa.
4. Inverzia predmetu a pomocného slovesa.

5. Odstránenie odpovedovej frázy a vloženie jednej z otázkových fráz na začiatku hlavnej klauzuly.
6. Postproces na zabezpečenie správneho formátovania.

Tento uvedený proces ilustruje obrázok 2.2.



Obr. 2.2: Proces transformácie deklaratívnych viet na otázky (Heilman, 2011).

Zátvorky označujú kroky, ktoré nemusia byť potrebné pre určité otázky.

* označuje kroky, ktoré môžu produkovať nulové alebo viacnásobné výstupy.

Hodnotenie otázok

Etapy 1 a 2 vytvárajú veľký počet otázok, z ktorých mnohé budú pravdepodobne neprijateľné. Mnohé otázky môžu byť najmä neprijateľné vzhľadom na vyššie uvedené výzvy. Väčšina používateľov QG nástrojov zváži len relatívne krátky zoznam kandidátskych otázok, a preto je potrebné identifikovať otázky, ktoré sú s najväčšou pravdepodobnosťou prijateľné a prezentovať ich v hornej časti zoznamu. Tretia etapa systému preto zaraďuje kandidátske otázky z 2. stupňa pomocou štatistického modelu prijateľnosti otázok. Preto je potrebné najprv popísať štatistický model a potom sadu funkcií ktoré používa.

Štatistický model

Na modelovanie kvality otázok používajú lineárnu regresiu najmenších štvorcov: kvalitu modelu ako lineárnu funkciu od vektora hodnôt vlastností ku kontinuálnej premennej v rozsahu od 1 do 5, pričom vyššie čísla označujú vyššie úrovne prijateľnosti otázok podľa jazykových faktorov, ako sú napr. gramatickosť, nejasnosť, použitie vhodného slova, atď. Akceptovateľnosť y otázky x je modelovaná nasledovne:

$$y = \mathbf{w}^T \mathbf{f}(x) \quad (2.1)$$

kde f je funkcia, ktorá berie otázku ako vstup a vracia vektor reálne hodnotených čísel vzťahujúcich sa na rôzne aspekty otázky a w je vektor reálnych hodnôt váhových parametrov pre každú funkciu vo vrátenom znakovom vektore f . Účelom je vyhľadať vektor w , ktorý minimalizuje súčet štvorcových chýb na tréningových údajoch s výhradou penalizácie na norme L2 váhového vektora, aby sa zabránilo preplneniu:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^N \left(y_i - \mathbf{w}^T \mathbf{f}(x_i) \right)^2 + \lambda \|\mathbf{w}\|_2^2 \quad (2.2)$$

Vo vyššie uvedenej rovnici je x_i jedinou inštanciou v tréningových údajoch s hodnotou prijateľnosti y_i a N je celkový počet tréningových prípadov. Táto forma penalizovanej lineárnej regresie sa často nazýva hrebeňová regresia (Hastie a kol., 2009). V uvedených experimentoch sa regulácia hyperparametra λ vybrala krížovou validáciou na tréningových údajoch.

Uvedená práca je veľmi prínosnou ukážkou vytvorenia QG systému. Vo svojej podstate analyzuje niektoré problémy, ktoré môžu nastať pri snahe o automatické generovanie otázok v určitom jazyku z rôznych uvedených hľadísk. Prácu a poznatky obsiahnuté v jadre tejto práce teda považujeme za veľmi prínosné vzhľadom na problematiku našej práce. Za nevýhodu teda môžeme považovať len špecifickosť riešenia v spojení s použitím anglického jazyka a neexistujúcu databázovú základňu pre poznatky. Vzhľadom na vyššie uvedené skutočnosti je možné veľmi elegantne a jednoducho identifikovať štruktúru a jednotlivé kroky procesu, ktorý je súčasťou QG systému. Definované výzvy sú do určitej miery zjavne integrovateľné aj do gramatických, lexikálnych a syntaktických štruktúr iných jazykov. Analýza v uvedenej práci predstavila rôzne jazykové a výpočtové výzvy spojené s automatickým faktickým QG, čo je relatívne nový problém, ktorý súvisí s množstvom ďalších problémov s generovaním textu na inú formu textu. Návrh umožňuje využívať existujúce NLP nástroje na syntaktickú analýzu, na využívanie jazykových znalostí pri vytváraní otázok a aplikovanie strojového učenia na hodnotenie otázok. Kľúčovou črtou tohto prístupu je štatistické hodnotenie otázok a vo všeobecnosti táto práca podporuje myšlienku technického spracovania prirodzeného jazyka.

2.2 Vytváranie grafov znalostí

Štruktúra alebo model, v ktorom sú uložené informácie zohráva najdôležitejšiu časť riešenia, keďže práve údajové štruktúry ovplyvňujú návrh funkcií systému. Vzhľadom na túto skutočnosť sme rovnako analyzovali a porovnávali existujúce riešenia z univerzít a inštitúcií, ktoré konkrétne odhaľujú problematiku znalostných grafov.

2.2.1 Identifikácia grafu znalostí

Veľké systémy na spracovanie informácií dokážu extrahovať masívne kolekcie vzájomne súvisiacich faktov, ale bohužiaľ, transformácia týchto údajov do užitočných vedomostí predstavuje obrovskú výzvu. Vedci z *Oddelenia informatiky na univerzite v Marylande* uvádzajú možnosť ako transformovať neisté extrakcie o entitách a ich vzťahoch do grafu vedomostí. (Pujara J., 2013) Extrakcie tvoria extrakčný graf, ktorý odkazuje na úlohu odstránenia šumu, odvodenie chýbajúcich informácií a určenie, ktoré kandidátske fakty by mali byť zahrnuté do znalostného grafu ako identifikácia grafu poznatkov. Aby mohla byť táto úloha splnená, je potrebné odvodniť kandidátske fakty a súvisiacu dôveru extrakcie, identifikovať korektné entity a začleniť ontologické obmedzenia. Navrhovaný systém využíva prístup *pravdepodobnostnej mäkkej logiky* (Probabilistic Soft Logic (PSL)), nedávno zavedenej pravdepodobnostnej modelovej štruktúry, ktorá s ľahkosťou konkretizuje milióny faktov. Systém demonštruje silu metódy na syntetickom korpuse *Linked Data*⁵ odvodenom z hudobnej komunity *MusicBrainz*⁶ a skutočnom súbore extrakcií z projektu *Never-Ending Language Learning (NELL)*⁷, ktorý obsahuje viac ako jeden milión extrakcií a sedemdesiat tisíc ontologických vzťahov. Systém dokazuje, že je schopný v porovnaní s inými existujúcimi metódami dosiahnuť zlepšenie Area Under Curve (AUC)⁸ a *F1*⁹ s výrazne nižšou dobou trvania.

Vo všeobecnosti vo svojej práci prezentujú kandidátske fakty zo systému extrakcie informácií ako graf znalostí, kde entity sú uzly, kategórie sú označenia

⁵Linked Data: <https://www.greekcoinage.org/what-is-linked-data.html>

⁶MusicBrainz: <https://musicbrainz.org>

⁷NELL: <https://ieeexplore.ieee.org/document/7004203>

⁸AUC: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>

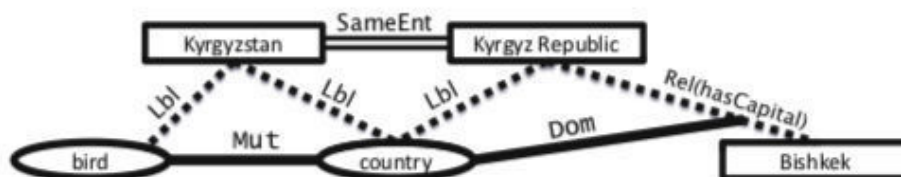
⁹F1: <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>

spojené s každým uzlom a vzťahy sú riadené hrany medzi uzlami. Systémy extrakcie informácií dokážu extrahovať takéto kandidátske fakty a tieto výňatky môžu byť použité na vytvorenie extrakčného grafu. Nakoľko graf extrakcie je často nesprávny, s chybami, ako sú falošné a chýbajúce uzly, hrany a chýbajúce alebo nepresné označenia uzlov. Identifikácia grafov vedomostí ako prístup spája úlohy riešenia entity, kolektívnej klasifikácie a predikcie odkazov sprostredkovanej pravidlami založenými na ontologických informáciách. Extrakcia entít je známy problém: viaceré textové odkazy, ktoré pôvodne vyzerajú inak, môžu odkazovať na rovnakú entitu v reálnom svete. Napríklad NELL vedomostná základňa obsahuje kandidátske fakty zahŕňajúce subjekty *“kyrgyzstan”*, *“kyrgyzstan”*, *“kyrgystan”*, *“kyrgyz republic”*, *“kyrgyzstan”*, a *“kyrgistan”*, čo sú všetky varianty alebo nesprávne napísané slová krajiny *Kyrgyzstan*. V extrahovanom grafe znalostí takéto nesprávne tvary zodpovedajú rôznym uzlom. Prístup v uvedenej práci využíva uznesenie entity na určenie ko-referenčných entít v znalostnom grafe a vytvára konzistentný súbor značiek a vzťahov pre každý vyriešený uzol.

Ďalšou výzvou v konštrukcii grafov vedomostí je dôsledné odvodenie značiek. Napríklad extrakcie NELL priradujú *Kyrgyzstanu* označenie *“krajina”*, ako aj *“vták”*. Použitie štítkov súvisiacich entít v znalostnom grafe umožňuje určiť štítok správnej entity. Uvedený prístup využíva *kolektívnu klasifikáciu* na označovanie uzlov spôsobom, ktorý berie do úvahy ontologické informácie a susedné štítky.

Tretím problémom, ktorý sa bežne vyskytuje v grafoch znalostí je určovanie vzťahov medzi entitami. NELL má veľa faktov o lokalizácii *Kyrgyzstanu* vzhľadom na iné subjekty. Medzi tieto kandidátske vzťahy patria vyhlásenia, že *Kyrgyzstan* sa nachádza v *Kazachstane*, *Kyrgyzstan* sa nachádza v *Rusku*, *Kyrgyzstan* sa nachádza v bývalom *Sovietskom zväze*, *Kyrgyzstan* sa nachádza v *Ázii* a *Kyrgyzstan* sa nachádza v *USA*. Niektoré z týchto možných vzťahov sú pravdivé, zatiaľ čo iné sú jasne falošné. Na riešenie tejto problematiky sa používa predikcia spojenia na predpovedanie okrajov spôsobom, ktorý berie do úvahy ontologické informácie a zvyšok dedukovanej štruktúry.

Obrázok 2.3 ilustruje jeden komplexný príklad. Ako už bolo uvedené, ontológia NELL zahŕňa obmedzenie, že označenia *“vták”* a *“krajina”* sa navzájom vylučujú. Kolektívne zdôvodnenie umožňuje vyriešiť, ktorý z uvedených štítkov sa pravdepodobnejšie vzťahuje na *Kyrgyzstan*. Príklad sa dá vyriešiť pomocou NELL, ktorý je veľmi presvedčivý, že *Kyrgyzská republika* má hlavné mesto, *Bishkek*. On-



Obr. 2.3: Ilustrácia príkladu, ktorý popisuje spôsob riešenia konfliktov v extrakčnom grafe (Pujara J., 2013).

tológia NELL špecifikuje, že doména vzťahu “*hasCapital*” má označenie “*country*”. Rozpoznanie entít umožňuje vyvodiť záver, že “*Kyrgyzská republika*” označuje rovnakú entitu ako “*Kyrgyzsko*”. Rozhodnutie o tom, či je *Kyrgyzsko* vtákom alebo krajinou zahŕňa predpoveď, v ktorej sú zahrnuté hodnoty dôveryhodnosti zodpovedajúcich faktov od spoločných subjektov, ako aj kolektívne znaky z ontologických vzťahov týchto spoločných subjektov, ako sú hodnoty dôveryhodnosti vzťahu “*hasCapital*”.

Pravdepodobnostná mäkká logika

Pravdepodobnostná mäkká logika (PSL) (Broecheler M., 2010, Kimmig a kol., 2012) je nedávno zavedený rámec, ktorý umožňuje používateľom špecifikovať bohaté pravdepodobnostné modely nad nepretržitými náhodnými premennými. Podobne ako ostatné štatistické jazyky výučby vzťahov, ako napríklad Markov Logic Networks (MLN)¹⁰, používa logiku prvého rádu na popis funkcií, ktoré definujú sieť. Na rozdiel od iných prístupov PSL používa skôr náhodné premenné než sú binárne premenné a odvodzuje najpravdepodobnejšie vysvetlenie ako konvexný optimalizačný problém, ktorý je výrazne efektívnejší na vyriešenie ako jeho kombinačný kontrapunkt (polynomický vs. exponenciálny).

Model PSL sa skladá zo súboru vážených logických pravidiel prvého rádu, kde každé pravidlo definuje súbor funkcií *Markov*-ej siete zdieľajúcich rovnakú váhu. Model je popísaný vzorcom:

$$P(A, B) \wedge Q(B, C) \stackrel{w}{\Rightarrow} R(A, B, C) \quad (2.3)$$

čo je príklad pravidla PSL. Kde w je váha pravidla, A , B a C sú univerzálne kvantifikované premenné a P , Q a R sú predikáty. Zakotvenie pravidla pochádza zo sub-

¹⁰MLN: <https://homes.cs.washington.edu/~pedrod/papers/mlj05.pdf>

stitúcie konštantných pre univerzálne kvantifikované premenné v atónoch pravidiel. V tomto prípade by priradenie konštantných hodnôt a , b a c k príslušným premenným vo vyššie uvedenom pravidle vytvorilo základné atómy $P(a, b)$, $Q(b, c)$, $R(a, b, c)$. Každý atóm má hodnotu mäkkej pravdivostnej hodnoty v rozsahu $[0, 1]$.

Identifikácia grafu znalostí pomocou PSL

Grafy poznatkov obsahujú tri typy faktov:

- fakty o subjektoch
- fakty o označeníach subjektov
- fakty o vzťahoch.

Na základe to ho je možné uvažovať entity s logickým predikátom $ENT(E)$ a štítikami s logickým predikátom $LBL(E, L)$, kde entita E má označenie L . Vzťahy sú reprezentované logickým predikátom $REL(E1, E2, R)$, kde vzťah R drží medzi subjektmi $E1$ a $E2$, napr. $R(E1, E2)$. Cieľom pri identifikácii grafov vedomostí je identifikovať skutočný súbor atómov z množstva nejasných extrakcií. Metóda identifikácie grafov vedomostí zahŕňa tri zložky:

- zachytenie neistých extrakcií
- vykonávanie rozlišovania entít
- presadzovania ontologických obmedzení

Uvedená štúdia formuluje problém identifikácie grafov vedomostí, opisuje odvodenie grafu poznatkov z nejasných výstupov systému extrakcie informácií prostredníctvom kombinovaného procesu určovania ko-referentných entít. Rovnako popisuje spôsob ako predvídať relačné vzťahy a kolektívnu klasifikáciu označenia entít a presadzovanie ontologických obmedzení. Výskumníci použili ako vstup syntetické dáta z *MusicBrainz* a zahltené reálne extrakty webových stránok NELL. Na oboch súboroch údajov identifikácia znalostných grafov vykazovala vynikajúce výsledky (možné zhliadnuť priamo v literatúre Pujara J., 2013) tým, že kombinuje silné stránky ontologického uvažovania s identifikáciou grafu. Vzhľadom na uvedené skutočnosti môžeme klasifikovať túto štúdiu ako užitočnú v oblasti vytvárania a identifikovania znalostných grafov potrebných na ďalšie spracovanie a generovanie testovacích otázok.

2.2.2 Freebase: Kolaboratívne vytvorená grafová databáza na štruktúrovanie ľudských vedomostí

Bollacker; Evans a kol., 2008 zo spoločnosti *Metaweb Technologies* vo svojej práci predstavujú *Freebase*, praktickú, škálovateľnú n-titnú databázu, ktorá sa používa na štruktúrovanie všeobecných ľudských znalostí. *Freebase* sprostredkúva vytváranie, štruktúrovanie a udržiavanie údajov. V čase predstavenia obsahovala viac ako 125 000 000 n-tíc, viac ako 4 000 typov a viac ako 7 000 vlastností. Verejný prístup na čítanie alebo zápis je povolený prostredníctvom API rozhrania s grafickým dopytom na báze HTTP pomocou jazyka MQL. MQL poskytuje ľahko použiteľné objektovo orientované rozhranie pre n-ticu údajov vo *Freebase* a je navrhnutý tak, aby uľahčil vytváranie webových aplikácií.

Freebase (Bollacker; Tufts a kol., 2007) je vo všeobecnosti databázový systém navrhnutý tak, aby bol verejným repozitárom vedomostí z celého sveta. Jeho dizajn je inšpirovaný široko využívanými informačnými komunitami, ako je sémantický *Web* (Berners-Lee a kol., 2001) a *Wikipédia*¹¹. Ako taký musí podporovať veľmi rôznorodé a heterogénne údaje súčasne so zabezpečením vysokej škálovateľnosti. Myšlienka existencie práve takéhoto systému je založená na základe princípov a existencie *World Wide Webu*. *World Wide Web* sa stal systémom pre šírenie a organizovanie informácií. V ideálnom prípade by všetky znalosti na webe boli dôsledne organizované a webové stránky by boli štruktúrované pre jednoduchý prístup už vo svojej štruktúre, reprezentácii a kvalite. Táto nekonzistentnosť často sťažuje prístup na web sofistikovanej analýze dát, vyhľadávaniu, organizácii či zobrazovaniu. Niektoré zo špecifických problémov zahŕňajú:

- Vnútorne štruktúrované informácie sú v „*v pasci*“ v neštruktúrovaných dokumentoch, ktoré a nedajú ľahko čítať pomocou automatizovaných procesov.
- Existencia nedostatku kanonických odkazov na entity (napr. žiadne URI pre ľudí). Často sú prítomné viaceré, neprepojené reprezentácie tej istej entity v reálnom svete.
- Nedostatočná separácia významu od prezentácie. Väčšina neštruktúrovaných a mnoho štruktúrovaných údajových zdrojov kombinujú sémantiku so

¹¹Wikipédia <https://www.wikipedia.org/>

zobrazovanými informáciami.

- Heterogénne zastúpenie informácií naprieč zdrojmi údajov a dokonca aj v rámci jedného zdroja.
- Webové informácie sú často prezentované bez explicitného kontextu iných, potenciálne užitočných informačných zdrojov.

Hoci väčšina webových stránok nemá vlastné funkcie, ktoré by tvorili dobrý základ na riešenie týchto problémov. Na základe uvedeného vzniklo niekoľko rozsiahlych výskumných prác o pridanie takýchto funkcií alebo o vytvorení rozšírených dátových systémov. Jedným z najvýznamnejších je práve *sémantický web*, ktorý predstavuje rámec na pridanie bohatej, hlbkej a hlavne štruktúrovanej sémantiky na webové stránky a informačné úložiská. *Sémantický web* sa vyvinul aby zahŕňal súbor štandardných formátov, protokolov a organizačných konvencií, ktoré umožňujú vytvorenie ontologicky a organizačne bohatých informačných štruktúr. Toto úsilie nachádza zmysel v nahradení kľúčových slov, ktoré sa bežne používajú na indexovanie a organizovanie informácií na webových stránkach.

Wikipédia, je ďalším významným prispievateľom v úsilí o všeobecné štruktúrovanie webových informácií. Táto internetová encyklopédia, založená na kolaborácii, inšpirovala výskumné úsilie v takých smeroch, ako je extrakcia entít (Auer a kol., 2007), hľadanie disambiguácie (Strube a kol., 2006) (Gabrilovich a kol., 2007) a sémantické štruktúrovanie (Krötzsch a kol., 2005).

Vo väčšine výskumných smerov smerujúcich k vytvoreniu štruktúrovaného Webu, existuje všeobecný názor, že riešenie rozsiahlej integrácie informácií, extrakcie entít a problémov pri štandardizovaní údajov vyžaduje systémy, ktoré automaticky vykonávajú štruktúralne a aextrakčné úlohy vo veľkých už existujúcich súboroch údajov. V takomto prípade sa pracovný postup uvedených a navrhovaných systémov pri štruktúrovaní informácií pokúša extrahovať, ukladať alebo rozširovať štruktúry na existujúcom webe, no samotné dáta na samotnom webe sú považované za nemené. Nakoľko kontrola údajov je vysoko distribuovaná a nekoordinovaná. *Sémantický web* poskytuje mechanizmi na prípravu štruktúry k pôvodným webovým dátam, ale jej decentralizovaná povaha prináša výzvy nerovnomerného osvojenia, ťažkosti pri implementácii netechnickými používateľmi a problémy s výkonom dotazov z jeho prirodzene distribuovanej štruktúry.

Štruktúra systému

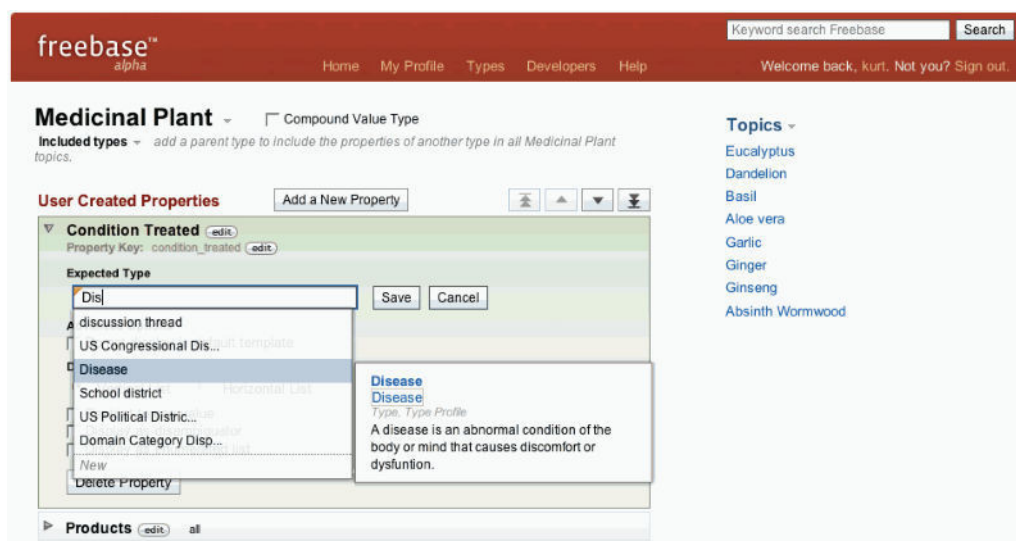
Funkcie *Freebase* sú navrhnuté tak, aby uľahčovali vysokú hustotu spolupráce medzi jej používateľmi a integráciami veľkých, rôznorodých súborov údajov. V abstrakcii datových priestorov (Halevy a kol., 2006), *Freebase* môže byť považovaný za DSSP s funkciami na zvýšenie „sémantickej integrácie“. *Freebase* má klienta založeného na AJAX pre ľudí a API JSON / HTTP pre stroje, tak aby obe skupiny mohli spolupracovať na vytváraní a upravovaní komplexných, vyvíjajúcich sa, štruktúrovaných dát. Konkrétne sa *Freebase* skladá z nasledujúcich komponentov a funkcií:

- **Úložisko údajov v grafickom tvare:** škálovateľná n-tica na ukladanie s určitým zabudovaním plánovaním a optimalizáciou dopytov so schopnosťami umožňujúcimi používanie rýchlych, hlbokých, naivne konštruovaných strojových dotazov.
- **Veľkokapacitné ukladanie dátových objektov (LOB):** úložisko veľkých dátových objektov, ako napríklad textové dokumenty, obrázky, zvukové súbory a softvér. Objekty sú v ňom indexované a anotované v ukladaacom priestore grafov a sú prístupné cez API.
- **Verejné API založené na protokole HTTP:** primárna metóda strojového prístupu k *Freebase* cez verejné API založené na HTTP. Dotazy a odpovede sú formulované pomocou jazyka Metaweb Query Language (MQL).
- **Webové používateľské rozhranie s jednoduchým ovládaním:** prostredníctvom webového rozhrania je používateľ schopný vyhľadávať, prezerať, upravovať a organizovať údaje reprezentované v ľudskej čitateľnej forme.
- **Podstatný základový údajový set:** údaje vo *Freebase* sa skladajú z miliónov konceptov (tém) a desiatok miliónov vzťahov medzi týmito témami.
- **Licencia na kreatívne poznatky:** jeden z problémov pri používaní a redistribuovaní údajov na webe sú obmedzenia a nejednoznačnosť autorských práv k týmto údajom. Riešením je preto existencia všeobecnej licencie, s ktorou všetci používatelia natívne súhlasia s distribúciou ich príspevkov.

- **Jednoduchý typový systém:** voľný zber štruktúrnych mechanizmov a konvencií, jednotlivé ontologické a reprezentačné systémy. Typový systém podporuje spoluprácu typov a vlastností, a neexistuje žiadny určitý svetový kanonický pohľad na všetky vedomosti.

Príklad interakcie so systémom

Príklad interakcie používateľa so systémom *Freebase* je názorne vizualizovaný na obrázku (2.4). Snímok obrazovky poukazuje na možnosť editácie schémy pre pojem „Liečivá rastlina“. V tomto prípade existuje vlastnosť „Podmienka ošetrenia“ ktorá je nastavená na hodnotu „Choroba“. Môžeme si všimnúť ako automatické dokončovanie navrhuje „Choroba“ ako výber, vrátane opisu, čo samotný typ reprezentuje. Toto vysoko kontextové zobrazenie informácií je jedným z kľúčových prvkov, ktoré znižujú vytváranie nadbytočných subjektov a tým napomáhajú v procese zmiernenia duplikácií a chybných anotácií.



Obr. 2.4: *Freebase* - používateľské rozhranie, snímka obrazovky úpravy typovej schémy (Bollacker; Tufts a kol., 2007).

Použitie *Freebase* API

Zatiaľ čo predchádzajúci príklad odzrkadľoval interakciu vo webovom používateľskom rozhraní, nasledujúci príklad poukazuje na komunikáciu s *Freebase* API. Toto rozhranie totižto podporuje vysoko expresívne štruktúrované dotazy. Pro-

stredníctvom API sa MQL dotazy v JSON syntaxi odosielajú cez HTTP. Pre bližšiu predstavu a vizualizáciu komunikácie uvádzame jednoduchý príklad MQL dotazu:

```
{
  "query": [{
    "album": [],
    "name": "The Police",
    "type": "/music/artist"
  }]
}
```

Výpis 1: MQL dotaz

Predchádzajúci dopyt ziada o albumy hudobného umelca a výsledok tohto dopytu má rovnaký tvar ako dopyt:

```
{
  "q1": {
    "status": "/mql/status/ok",
    "result": [{
      "album": [
        "Outlandos d'Amour",
        ...
      ],
      "type": "/music/artist",
      "name": "The Police"
    }]
  }, "status": "200 OK"
}
```

Výpis 2: Odpoveď na MQL dotaz

Služba *Freebase* je vybudovaná ako praktická platforma pre vytváranie a vývoj štrukturovaných údajov. Webové používateľské rozhranie a API podporujú množstvo interaktívnych pracovných postupov, ktoré umožňujú pridávať špecifickým

potrebám globálne použiteľnú štruktúru, ktorá môže byť okamžite verejne použitá. Táto platforma teda predstavuje ukážkový príklad systému na spracovanie štrukturovaných údajov do grafovej formy. V súčasnosti je však táto platforma označená ako zastaralá a bola nahradená no v našom prípade svojou štruktúrou a architektúrou dokáže napomôcť pri návrhu nášho systému. Za určitú nevýhodu v tomto prípade považujeme práve jazykovú bariéru, a teda chýbajúcu podporu pre ďalšie jazyky ako napríklad Slovenčina.

2.2.3 Wikidata - vedomostná základňa

Wikipédia, pôvodne koncipovaná ako prevažne textový zdroj, zhromažďuje rastúce množstvo štruktúrovaných údajov: čísla, dátumy, súradnice a mnoho typov vzťahov z rodokmeňov k taxonómii druhov. Tieto údaje sa stali zdrojom nesmiernej hodnoty, s potenciálnymi aplikáciami vo všetkých oblastiach vedy, techniky a kultúry. Tento vývoj je sotva prekvapujúci vzhľadom na to, že *Wikipédia* je poháňaná všeobecnou víziou „sveta, v ktorom sa každá ľudská bytosť môže slobodne podieľať na súhrne všetkých vedomostí“. Dnes nie je pochyb o tom, že táto zbierka musí obsahovať údaje, ktoré možno vyhľadávať, analyzovať a opätovne použiť (Vrandečić a kol., 2014).

Preto môže byť prekvapujúce, že *Wikipédia* neposkytuje priamy prístup k väčšine týchto údajov, a to ani prostredníctvom dotazovacích služieb, ani prostredníctvom exportov dát na stiahnutie. Skutočné využívanie údajov je zriedkavé a často sa obmedzuje na veľmi špecifické informácie, ako sú napríklad geografické značky článkov *Wikipedia* používaných v *Mapách Google*. Dôvodom tejto pozoruhodnej priepasti medzi víziou a realitou je, že údaje *Wikipédie* sú pochované v 30 miliónoch článkov v 287 jazykoch, odkiaľ je ich veľmi ťažké extrahovať.

Cieľom *Wikidata* je prekonať tieto problémy vytvorením nových spôsobov, ako spravovať údaje v globálnom meradle. Vysledok týchto snáh je možné vidieť na wikidata.org.

- **Otvorené úpravy:** Podobne ako *Wikipedia*, aj *Wikisource* umožňujú každému používateľovi webu rozšíriť a upraviť uložené informácie, dokonca aj bez vytvorenia účtu.
- **Kontrola spoločenstva:** Komunita prispievateľa kontrolu nielen aktuálne údaje, ale aj schému údajov.

- **Pluralita:** Bolo by naivné očakávať globálnu dohodu o pravdivých údajoch, pretože mnohé skutočnosti sú sporné alebo jednoducho neisté. *Wikidata* umožňuje koexistovať konfliktné údaje a poskytuje mechanizmy na organizovanie tejto plurality.
- **Sekundárne údaje:** *Wikidata* zhromažďuje fakty zverejnené v primárnych zdrojoch spolu s odkazmi na tieto zdroje.
- **Viacjazyčné údaje:** Väčšina údajov nie je viazaná na jeden jazyk: čísla, dátumy a súradnice majú univerzálny význam. *Wikidata* je viacjazyčná už na úrovni návrhu. Kým *Wikipédia* má pre každý jazyk nezávislé vydania, vo *Wikidata* existuje len jedno miesto.
- **Jednoduchý prístup:** Cieľom *Wikidata* je umožniť používanie údajov vo *Wikipédii* aj v externých aplikáciách. Údaje sa exportujú prostredníctvom webových služieb v niekoľkých formátoch, vrátane JSON a RDF. Údaje sa zverejňujú za právnych podmienok, ktoré umožňujú čo najširšie možné použitie.
- **Nepretržitá evolúcia:** Namiesto vývoja dokonalého systému, ktorý je svetu prezentovaný v priebehu niekoľkých rokov, sa nové funkcie rozrastajú postupne a čo najskôr.

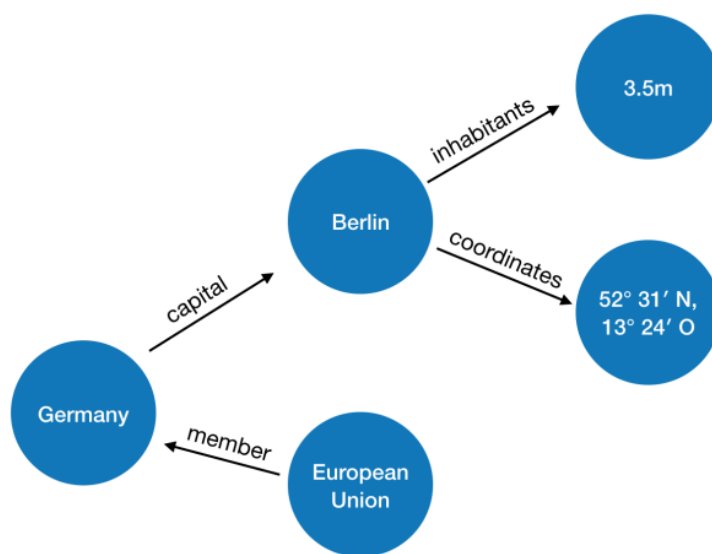
Tieto vlastnosti charakterizujú *Wikidata* ako špecifický druh kurátorskej databázy (Buneman a kol., 2008).

Na ukladanie štruktúrovaných údajov nad rámec textových textov a jazykových odkazov používa *Wikidata* jednoduchý dátový model. Údaje sú v podsate opísané pomocou párov vlastností a hodnôt. Vlastnosti sú tak vlastnými objektami, ktoré vlastní stránky *Wikidata* spolu s menovkami, aliasmi a popismi. Existuje tu len malý počet dátových typov, hlavne množstvo, položka, reťazec, dátum a čas, geografické súradnice a adresa URL.

Aby používatelia mohli zadávať na mieru šité dotazy na báze znalostí, *Wikimedia* začala hostovať oficiálnu dotazovaciu službu (Hernández a kol., 2016), ktorá podľa interných štatistík dostáva stovky tisíc dopytov denne. Služba dotazov beží cez RDF reprezentáciu *Wikidata*, ktorá je indexovaná v *Blazegraph SPARQL*¹² úložisku (predtým známom ako *BigData*) (Thompson a kol., 2016). Vo *Wikidata* sú

¹²Blazegraph: <https://www.blazegraph.com/>

položky prepojené buď s príslušnými položkami alebo s údajmi typu *datatype* pomocou riadených, pomenovaných vzťahov. Vzťahy môžu byť anotované pármí atribútov a hodnôt, ako sú napríklad kvalifikátory a odkazy, na určenie ďalšieho kontextu pre vzťah (napr. platnosť, príčina, stupeň atď.). Obrázok 2.5 názorne vizualizuje reprezentáciu štruktúrovaných dát v uvedenej Wikidata databáze, určenej na ukladanie vedomostí.



Obr. 2.5: Príklad zobrazenia Wikidata RDF databázy¹³.

Wikidata, ako slobodná vedomostná báza údajov, ktorú môže každý upraviť, nás môže priviesť o krok bližšie k svetu, v ktorom sa každý môže slobodne podieľať na súhrne všetkých vedomostí. Analýzu práve tohto projektu sme vybrali za účelom analýzy existujúcej služby s obrovskou bázou dát, ktorá slúži na ukladanie vedomostí a vzťahov medzi nimi do grafov. *Wikidata* nám rovnako sprostredkúva pohľad na možné štruktúrovanie modelov pre návrh nášho riešenia. Ako veľké pozitívum môžeme pokladať existenciu viac-jazyčnej štruktúry a veľkej podpory množiny odkazov a prepojení. Existujú aj ďalšie služby, ktoré sprostredkujú podobnú funkcionality no väčšinou nie sú otvorené a ich mišlienka a účel nie sú totožné s *Wikidata*, medzi ne patrí: *Knowledge Vault* (Dong a kol., 2014), *Google Kno-*

¹³A Brief Introduction to Wikidata: <https://towardsdatascience.com/a-brief-introduction-to-wikidata-bb4e66395eb1>

*wledge Graph*¹⁴, *YAGO*¹⁵, *NELL*¹⁶, *DBpedia*¹⁷, *DeepDive*¹⁸ a samozrejme komerčné projekty od spoločností *Microsoft*¹⁹, *Facebook*²⁰, atď. Môžeme teda tvrdiť, že používanie znalostných grafov a stromov je v súčasnej dobe do značnej miery rozšírené. Jednotlivé grafy z uvedených spoločností alebo projektov tak pre spoločnosť ako takú predstavujú určitý základ súhrnných vedomostných databáz. Dotazy na takto štrukturované databázy sa jednoznačne tešia určitému úspechu, nakoľko sú veľmi rýchle a výkonné. Aj keď sa uvedené projekty používajú na rôzne účely, jedno majú spoločné, snažia sa o sumarizáciu vedomosti vo všeobecnosti na celom svete a už v súčasnosti obsahujú milióny prvkov.

¹⁴Google Knowledge Graph API: <https://developers.google.com/knowledge-graph/>

¹⁵YAGO: <https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/>

¹⁶NELL: <http://rtw.ml.cmu.edu/rtw/overview>

¹⁷DBpedia: <https://wiki.dbpedia.org>

¹⁸DeepDive: <http://deepdive.stanford.edu/kbc>

¹⁹Microsoft Knowledge base: <https://support.microsoft.com/en-us/help/242450/how-to-query-the-microsoft-knowledge-base-by-using-keywords-and-query>

²⁰Facebook Graph API: <https://developers.facebook.com/docs/graph-api/>

3 Metodológia

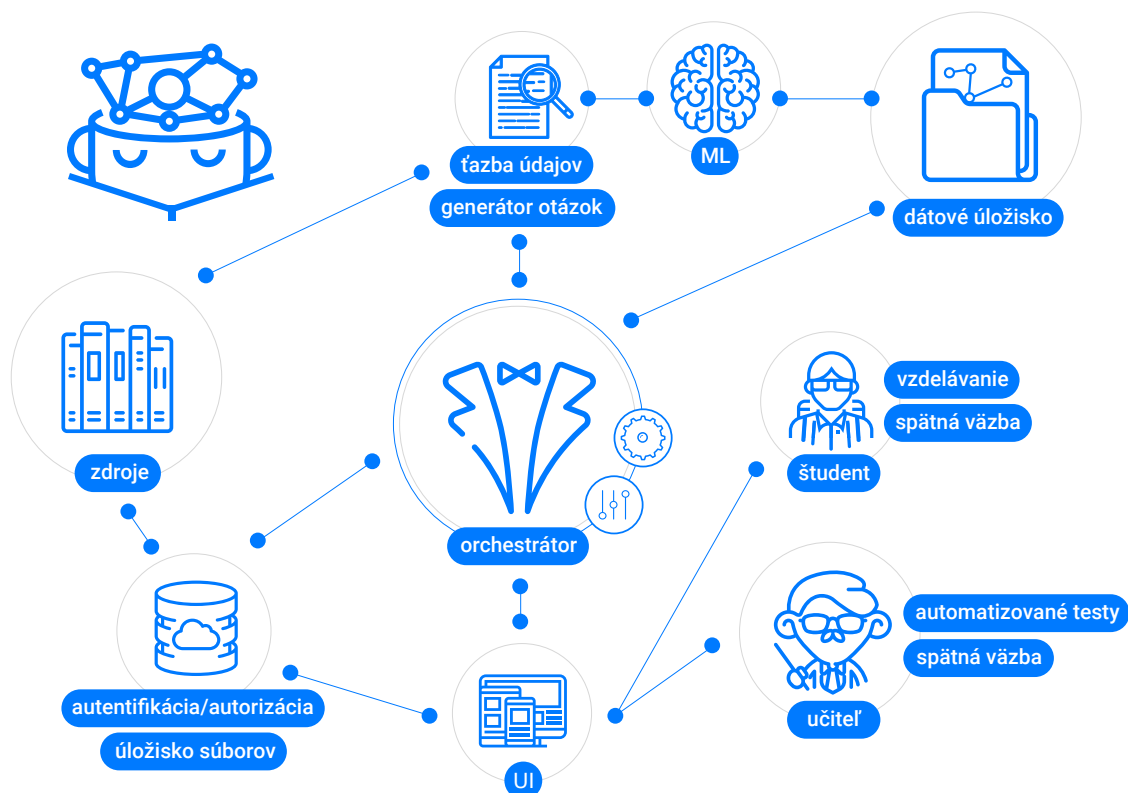
V tejto časti práce sa budeme venovať konceptuálnemu návrhu riešenia a jednotlivým výskumným metódam, ktoré budú použité v rámci implementácie jednotlivých podsystémov. Metodológia použitá v rámci tejto práce bola navrhnutá na základe analýzy už existujúcich riešení, ktorej sme sa venovali v predchádzajúcej kapitole 2. Analýza priniesla mnoho použiteľných poznatkov, ktoré budú pri návrhu a implementácii veľmi užitočným zdrojom informácií a postupov potrebných na splnenie stanovených cieľov. Získané poznatky v kontexte generovania otázok a tvorby znalostných grafov dosahujú vysokú zložitosť a postupy pri analýze textu prostredníctvom NLP nástrojov zložitosť riešenia ešte viac komplikujú. V nasledujúcich podkapitolách preto uvádzame metódy a postupy použité v riešení v čo najzrozumiteľnejšej forme.

3.1 Návrh riešenia

Nami navrhnuté riešenie nesie názov *Fairgrade*, a bolo navrhnuté za účelom generovania faktických otázok, vytvorenia znalostného grafu a sprístupnenia získaných poznatkov z analýzy do Slovenského jazyka. Pomocou zistení a výsledkov z analýzy existujúcich riešení sme vybrali technológie, postupy a myšlienky, ktoré by pravdepodobne mali z celkového všeobecného technologického a výpočtového pohľadu priniesť optimálne celkové alebo čiastočné riešenie uvádzanej problematiky. Obrázok 3.1 názorne ilustruje konceptuálny návrh systému na generovanie faktických otázok.

Hlavným podsystémom ilustrovaného systému je tzv. *orchestrátor*, ktorý je reprezentovaný prostredníctvom API. Tento nástroj má slúžiť na ovládanie a monitorovanie všetkých ostatných podsystémov. Navrhli sme teda hlavný monitorovací bod, ktorý posluží na validáciu a verifikáciu fungovania jednotlivých súčastí

platformy. Ďalšou neoddeliteľnou súčasťou systému je cloudový systém, ktorý posluži ako autentifikačný a autorizačný sprostredkovateľ a rovnako ako úložisko vstupných textových súborov. Cloudový systém aj spolu s *orchestrátorom* vytvárajú základné programové rozhranie pre používateľské rozhranie (UI).



Obr. 3.1: Konceptuálny návrh riešenia (vlastné spracovanie).

Definovali sme aj cieľovú skupinu používateľov, ktorý by mohli navrhovaný systém používať. Rozdelili sme používateľskú množinu na dve kategórie a to menovite na:

- **Študenti:** využívanie vedomostnej bázy údajov a hodnotenie vedomostí prostredníctvom automaticky generovaných testov.
- **Učitelia:** generovanie automaticky generovaných faktických testov z odbornej literatúry a štatistické hodnotenie pedagogických výkonov.

Uvedení používatelia nadobudnú prístup k jednotlivým zdrojom systému prostredníctvom cloudového systému a orchestrátor dokáže poskytnúť monitorovacie

body potrebné pri sledovaní postupnosti krokov potrebných pri spracovávaní textových dokumentov.

Najdôležitejšiu časť navrhovaného systému tvorí systém na ťažbu údajov, generovanie otázok a dátové úložisko. Táto časť má za úlohu spracovávať vstupný text, analyzovať jednotlivé syntaktické, lexikálne a morfológické charakteristiky a následne ich vyhodnocovať a triediť pomocou nástrojov strojového učenia (ML). Takto analyzované vetné štruktúry budú následne ukladané do dátového úložiska vo forme grafu aby sa zabezpečilo zachovanie dôležitosti jednotlivých vzťahov a prepojení. Analyzované vetné konštrukcie rovnako poslúžia aj na generovanie faktických otázok. Analyzátor teda predstavuje najdôležitejšiu súčasť celkového systému a záleží len na jeho stavbe či otázky a analyzované vedomosti budú dosahovať určitú stanovenú úroveň kvality. Tento nástroj bude sprostredkúvať funkcionality lexikálneho, syntaktického, či morfológického analyzátora za účelom identifikovania vetných a slovných konštrukcií, určitých definovaných vzorov a vzťahov voči už existujúcim štruktúram.

Vstup pre navrhovaný systém je reprezentovaný textom s určitou úrovňou faktických konštrukcií a diskurzívnych štruktúr. Po zálohovaní vstupných textov v cloudovom súborovom úložisku sú v systéme navrhnuté procesy extrakcie textu z týchto súborov ich prvotná analýza a čistenie a následná analýza už spomínaným analyzátorom. Textové zdroje by mali byť nahrávané priamo z používateľského rozhrania, čo najjednoduchšími operáciami.

Navrhovaný systém teda na prvý pohľad určite splňa jeho vstupné požiadavky, a stáva sa tak určitou predlohou do ďalšej fázy implementácie a integrácie analyzovaných výpočtových procesov.

3.2 Metódy výskumu

V tejto podkapitole sa detailnejšie venujeme vybraným a navrhovaným metódam prístupu k jednotlivým častiam systému a teda hlavne k častiam s problematikou generovania faktických otázok, analýzy vstupných textov a štruktúrovaní výstupných analyzovaných dát, ktoré poslúžia na ďalšie možné rozšírenia spracovania a detailnejšiu detekciu podstatných vedomostí. Zamerali sme sa teda hlavne na overené postupy z analytickej časti práce spolu s možnými vylepšeniami a rozširujúcim výskumom. Výskum ako taký sme v týchto oblastiach špeci-

ficky zacielili na analýzu práve Slovenského jazyka. Naša snaha sa teda špecializovala na použitie postupov zistených v analytickej časti práce na štruktúru nášho materinského jazyka. Náš výskum sa zaoberá hlavne analýzov na úrovni NLP a NLG v morfológických a syntaktických vodách Slovenčiny.

3.2.1 Generátor otázok

Existenciu modulu (QG), ktorý má zabezpečiť funkcionálnosť generovania otázok sme podmienili postupom opísaným v kapitole 2.1.4, v ktorej Heilman, 2011 detailne popísal jednotlivé problémy vznikajúce pri snahe o automatické generovanie faktických otázok. Nakoľko je generovanie všetkých možných typov otázok veľmi rozsiahly jazykovedný výskum, rozhodli sme sa pre skúmanie primárne faktických otázok, ktorých odpovede sa skladajú z predložkových fráz (napr. v roku 1801). Rovnako sme sa inšpirovali technikou ktorá bola podobne použitá v existujúcom voľne dostupnom jednoduchom nástroji s názvom *otazkovač*¹. Z jazykovedného hľadiska v Slovenčine, ak veta začína časovou predponou alebo predponou miesta, potom sa táto veta dá ľahko previesť na otázku nahradením jej predponovej časti opytovacím zámenom *Kedy?* alebo *Kde?*. Aby sme však identifikovali vety, ktoré by sa mohli transformovať na otázky uvedeného typu sú potrebné dve fázy:

1. **rozdelenie textu do viet**
2. **zistenie, či veta začína predponou.**

Na to aby sme tieto dva prvotné kroky mohli vykonať je potrebné použiť funkcie NLP nástroja, ktorý by v prvom kroku rozdelil a detekoval jednotlivé vety v súvislosti textu a v druhom kroku morfológicky anotoval text v Slovenskom jazyku, kvôli detekcii predpôn v analyzovaných vetách. NLP nástroj teda musí vykonať lematizáciu a taggovanie jednotlivých vetných členov, tak aby sme následne boli schopní analyzovať typ vety a typy jednotlivých vetných členov. lematizácia je proces vytvárania lémy z určitého tvaru slova a morfológická anotácia - taggovanie identifikuje charakteristické vlastnosti jednotlivých slov vo vete.

¹Otazkovač: <https://github.com/mrshu/otazkovac>

Morfologická anotácia textov Slovenského národného korpusu

V prvých krokoch pri analýze vstupných textov prichádza na rad lexikálny, syntaktický a morfologický analyzátor, ktorý jednotlivé vetné členy označí a kategorizuje prostredníctvom určitých značiek. Tieto značky a potrebné vedomosti o štruktúre Slovenského jazyka sme získali priamo po registrácii a certifikovaní nášho prístupu zo stránok a webových aplikácií *Jazykovedného ústavu Ľudovíta Štúra*². Získali sme tak prístup ku korpusu Slovenského jazyka (Ľ. Štúra SAV, 2018) ktorý v súčasnej dobe obsahuje takmer **1,5 mld.** tokenov. Využitie tohto korpusu je bezplatné na vedecko-výskumné účely. Nakoľko všetky možné taggovacie značky jednotlivých slovných druhov v slovenskom jazyku predstavujú špecifickú množinu a pretože sú reprezentované rozsiahlymi tabuľkami (JÚLS, 2019). No pre účely nášho generovania faktických časových a miestnych prepozičných otázok je potrebné poznamenať, že tagy prepozičného javu začínajú značkou *E*. Príklad takto značkovanej vety je ilustrovaný vo výpise 3.

V|Eu6 prvom|NAis6 zápase|SSis6 majstrovstiev|SSnp2 sveta|SSis2
nebudú|VBepc- hrať|VIe+ Slováci|Gkis4x s|Eu7 plnou|AAfs7x
súpiskou|SSfs7 .|Z

Výpis 3: Príklad značkovanej prepozičnej vety. (Spracované pomocou nástroja *Dagger*³)

Ako vidíme na príklade, značka prvého slova v uvedenej vete začína značkou *E* a preto môžeme toto slovo považovať za predložku. Môžeme si však rovnako všimnúť, že značky nasledujúcich dvoch slov zdieľajú číslo 6. Táto skutočnosť indikuje, že tieto slová sú v 6. slovenskom páde (*Lokál*). Rovnako aj ďalšie nasledujúce slová pred hlavným slovesom vety zdieľajú číslo 2, ktoré indikuje že tieto slová zdieľajú pád *Genitív*. Ako môžeme vidieť, týchto 5 spomínaných slov je možné nahradiť opytovacím zámenom *Kedy*, následne bodku na konci vety nahradiť otáznikom a vyprodukovali by sme celkom dobrú užitočnú otázku (4).

²Korpus Slovenského jazyka: <https://korpus.sk/>

³Dagger: <https://github.com/hladek/dagger>

Kedy nebudú hrať Slováci s plnou súpisťou?

Výpis 4: Príklad vytvorenej faktickej otázky s prepozičnou odpoveďou (vlastné spracovanie).

Detekcia typov viet

Vyššie uvedené predpisové klauzuly však nemusia byť vždy spojené s časovým odkazom. Existuje určitá úroveň pravdepodobnosti že detekovaná klauzula môže byť naopak spojená s miestom a nie časom. Preto vy v takýchto prípadoch bolo lepšie nahradiť opytovacie zámeno *Kedy* zámenom *Kde*. Z týchto príkladov môžeme teda dedukovať, že na to, aby sme vytvorili správne a relevantné otázky, analyzátor musí byť schopný zistiť, aký typ otázky možno vygenerovať z danej predloženej vety. Práve kvôli takýmto problémom sme z korpusu slovenského jazyka vyfiltrovali a zhromaždili súbor viet, ktoré by sa mohli transformovať na otázky, ako je to ukázané a popísané vyššie. Následne sme prostredníctvom *nlp* značkovača, formátovali uvedený súbor do formátu, ktorý je uvedený vo výpise 5, tak aby bola detekcia odolnejšia a prístupnejšia variáciám v prirodzenom jazyku.

veta \t (lema|token)+ \n

...

Výpis 5: Formát tréningových dát po morfolologickej analýze (vlastné spracovanie).

Žiaľ predpoklad uvedený vyššie vo všeobecnosti vďaka variabilite v prirodzenom jazyku nie je dokonalý a existujú aj vety, ktoré nespádajú ani do skupiny viet s predponou miesta (označované ako *P - Place*) a ani do kategórie viet s časovou predponou (označované ako *T - Time*). Tieto vety by v takomto prípade mali byť ignorované, a preto sme ich v súbore údajov označili značkou *I - Ignored*. Po takomto značkovani a úprave tréningových údajov, súbor nadobudol štruktúru ktorá je uvedená vo výpise 6.

S takto štruktúrovanými dátami je následne možné natrénovať model strojového učenia za účelom detekcie výskytu typu vety s určitou pravdepodobnosťou. Nasledujúca podkapitola pojednáva o výbere typu modelu, ktorý by bol vhodný

```
veta \t (lema|token)+ \t (P|T|I) \n
```

```
...
```

Výpis 6: Formát tréningových dát po identifikácii typov viet.

práve pre túto uvedenú situáciu.

Výber typu modelu strojového učenia

Pokiaľ ide o klasifikáciu textu, existuje viacero vhodných modelov. Ako východiskovú hodnotu sme použili multinomiálny klasifikátor *Naive Bayes (NB)*⁴. No vyskúšali sme aj klasifikátor *Random forest (RF)*⁵ ako príklad modelu, ktorý nemá tendenciu preplnenia, a *SVM*⁶, ktorý je jedným z odporúčaných modelov, pokiaľ ide o klasifikáciu textu na malých súboroch údajov. Všetky modely boli testované v kombinácii so znakmi opísanými vyššie s použitím 10-násobnej krížovej validácie. Výsledná presnosť je uvedená v tabuľke 3.1.

Tabuľka 3.1: Výsledky testovania presnosti klasifikátorov textu

	NB	RF	SVM
2-3 normal	86,37	84,92	84,85
2-4 normal	87,21	85,56	85,17
1-4 specia	88,49	86,72	88,16
2-4 revers	88,39	86,36	87,32
2-3 revers	87,28	89,88	89,44

Čísla sú stupne *gramov*, ktoré boli použité (2-3 *gramy* znamená, že boli použité *bigramy* a *trigramy*). *Normal* znamená používanie iba lematizovaných slov, *specia* znamená použitie typovej premennej a *revers* značí použité značiek ako slov. Ako sa ukázalo funkcie sú v podstate rovnaké ako značky s *fourgram*-ami.

Pri tréňovaní na celom dátovom súbore vybraných 927 viet z korpusu slovenského jazyka sme získali presnosť modelu 0,9848975188781014. Správa v tabuľke 3.2, reprezentuje výsledky klasifikácií pre jednotlivé klasifikačné triedy.

⁴Naive Bayes: <https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c>

⁵Random forest: <https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd>

⁶SVM: <http://smnd.sk/mcibula/alg/SVM.html>

Tabuľka 3.2: Správa o klasifikácii pre príslušne triedy

	presnosť	vyvolanie	f1-skóre	podpora
I	0,98	0,93	0,96	165
P	0,98	1,00	0,99	419
T	0,99	0,99	0,99	343
micro priemer	0,98	0,98	0,98	927
macro priemer	0,98	0,97	0,98	927
vážený priemer	0,98	0,98	0,98	927

Tento natrénovaný model sa následne použije na detekciu typov predponových viet. K tejto klasifikácii v systéme dochádza po detekcii viet v súvislom texte a po lematizácii a morfolologickej analýze a detekcii prepozičných viet. Následne na základe tejto klasifikácie dôjde s určitou pravdepodobnosťou k nahradeniu prepozičnej časti vety správnym opytovacím zámenom.

Na detekciu a generovanie ďalších typov otázok je určite potrebná rozširujúca jazykovedná analýza a podpora nejakého šikovného jazykovedca by bola určite veľkým prínosom. Z technického hľadiska je táto metodológia určite integrovateľná aj na generovanie a klasifikáciu iných typov otázok. Tieto skutočnosti však ponechávane na ďalšiu budúcu prácu, nakoľko postup spracovania je pri každom type otázok do určitej miery iný a vykazuje určité obmeny v jednotlivých krokoch spracovania textu.

3.2.2 Graf znalostí

Bondy a kol., 1976 vo svojej knihe o teórii grafov uvádzajú, že mnohé situácie v reálnom svete možno pohodlne opísať pomocou schémy pozostávajúcej zo sady bodov spolu s líniami spájajúcimi určité páry týchto bodov. Napríklad body by mohli reprezentovať ľudí, pričom čiary spájajú páry priateľov; alebo body môžu byť komunikačné centrá s linkami reprezentujúcimi komunikačné spojenia. Je možné si všimnúť, že v takýchto diagramoch je záujem kladený hlavne na to, či sú dva dané body spojené čiarou; spôsob, akým sú spojené, je nepodstatný. Matematická abstrakcia situácií tohto typu vedie k pojmu graf.

Táto podkapitola pojednáva o metodológii spojenej s vytváraním grafových

znalostných štruktúr, ktoré sme zahrnuli do návrhu systému za účelom vytvorenia bázy údajov, ktorá následne môže poslúžiť na detekciu vedomostí a následné generovanie otázok z tejto štruktúry. Na základe analýzy projektu *Wikidata* v podkapitole 2.2.3 sme získali poznatky ohľadom štruktúrovania informácií v grafoch vytvorených na základe párových spojení atribútov a hodnôt a popisovateľných hrán jednotlivých uzlov v grafe. Metóda vytvárania modelu, či štruktúry teda uvádza možnosť uchovávať informácie ukotvené v uzloch alebo aj priamo v hranách grafu.

Vrcholy

Pri lematizácii vznikajú lemy ako základné formy slov z jednotlivých vetných členov. Vzhľadom na túto skutočnosť a po určitých pokusoch pri testovaní a skúmaní štruktúr z analyzátora sme vytvorili novú metódu štruktúrovania znalostných grafov. Vrcholy v našej štruktúre sú reprezentované lemy, ktoré reprezentujú základnú inštanciu konkrétneho slova alebo pojmu v grafe. Tvorba lemy pre jednotlivé vetné členy úzko súvisí s modelom značkovača, ktorý je použitý v NLP nástroji, ktorý vykonáva analýzu v procese lematizácie. Jednotlivé vrcholy v navrhovanom grafe je rovnako možné anotovať značkami alebo obohatiť ich vnútornú štruktúru ďalšími vlastnosťami prostredníctvom atribútov, tak ako je to ilustrované na diagrame 3.2.

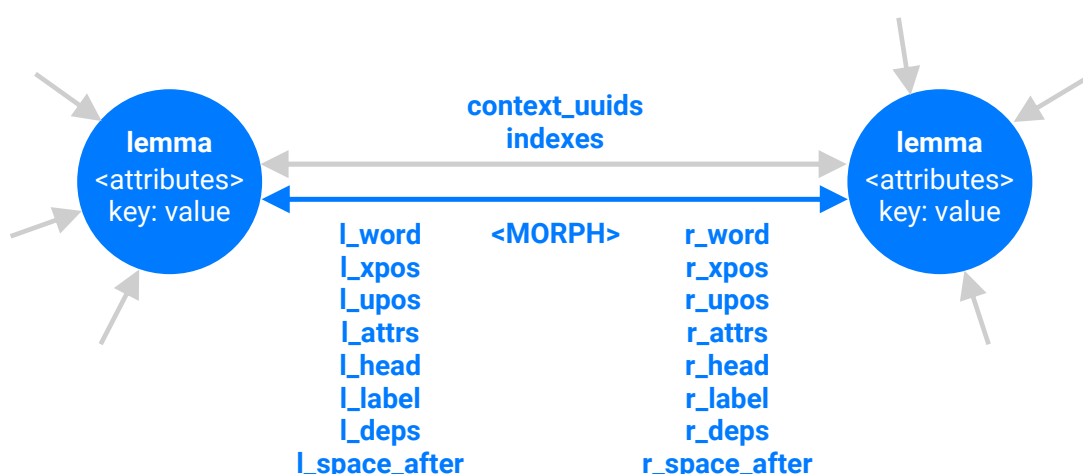
Hrany

Nasledujúcim krokom po lematizácii vetných členov je morfológická analýza a značkovanie jednotlivých slov značkami, ktoré opisujú ich slovný druh, číslo, pád, osobu a iné v závislosti od konkrétnych vetných členov. Všetky detekované charakteristiky identifikujú a popisujú hranu, teda vzťah medzi danými slovami, ktoré po sebe nasledujú vo vstupnej vete. V našej metóde sme teda definovali referenčný vzťah medzi vrcholmi grafu pod názvom *MORPH* tak ako je to názorne vizualizované na diagrame 3.2. Jednotlivé atribúty tohto vzťahu sú rovnako súčasťou CoNLL formátu⁷ a majú nasledujúci význam (Boroş a kol., 2018):

- **context_uuids:** pole unikátnych identifikátorov kontextov, v ktorých sa vyskytla daná väzba

⁷CoNLL formát: <https://universaldependencies.org/format.html>

- **indexes:** pole unikátnych identifikátorov kontextov spolu s poradovými číslami ľavého a pravého vetného člena
- **l_word/r_word:** originálny tvar ľavého/pravého člena vystupujúceho v danom vzťahu
- **l_xpos/r_xpos:** jazykovo špecifická časť reči ľavého / pravého člena; podtržník ak nie je k dispozícii
- **l_upos/r_upos:** tieto značky označujú základné kategórie reči ľavého / pravého člena
- **l_attrs/r_attrs:** morfológické atribúty ľavého / pravého člena
- **l_head/r_head:** hlava aktuálneho slova, ktorá je buď hodnotou ID ľavého / pravého člena alebo nula (0)
- **l_label/r_label:** špecifické značky označujúce charakteristiky ľavého / pravého člena
- **l_deps/r_deps:** graf závislostí ľavého / pravého člena vo forme zoznamu párov
- **l_space_after/r_space_after:** identifikátor, ktorý indikuje, či po ľavej alebo pravej leme nasleduje medzera



Obr. 3.2: Štruktúra grafových uzlov a hrán (vlastné spracovanie).

Na základe popísanej metódy vytvárania špecifickej štruktúry grafov vytvorených sme získali štruktúrovaný spôsob uchovania vedomostí a závislostí. Oproti takejto štruktúre je následne možné vytvárať rôznorodé sémanticky podložené dopyty na jednotlivé atribúty vrcholov grafu ale aj na atribúty hrán. Rovnako je možné použiť jednotlivé už existujúce algoritmy z problematiky teórie grafov na získavanie zaujímavých poznatkov z určitých hladín kontextu a významu.

Ako príklad použiteľnosti takto navrhovanej údajovej štruktúry uvádzame jednoduchý pohľad na detekciu a určenie zoznamu faktov a poznatkov ktoré vykazujú dôležitosť vzhľadom na kontext analyzovaných odborných textov. Už na prvý pohľad je možné objaviť v spomínanej grafovej štruktúre skutočnosť, že najpodstatnejšie poznatky a pojmy majú v grafe čo najviac hrán vzhľadom na iné vrcholy a vo svojej štruktúre vykazujú napríklad slovný druh podstatné mená. Takýmto spôsobom je teda možné vytvoriť zoradený zoznam vedomostí podľa dôležitosti z určitého konkrétneho kontextu alebo diskurzu. Podobným, rovnako zaujímavým poznatkom, ktorý sa dá dedukovať z grafu, sú napríklad jednotlivé vlastnosti konkrétnej lemy podstatného mena. Vyhľadaním prídavných mien, ktoré zdieľajú hranu s konkrétnym podstatným menom dokážeme získať zoznam vlastností daného podstatného mena. Určovaním štatistických identifikátorov napríklad analýzou ciest medzi jednotlivými vrcholmi dokážeme detekovať zaujímavé závislosti jednotlivých pojmov a podobne.

Navrhnutá grafová štruktúra predstavuje veľmi úrodnú pôdu na ďalší rozvoj a rovnako by uvedená štruktúra nemala byť závislá na konkrétnom jazyku. No toto tvrdenie nedokážeme s určitosťou potvrdiť nakoľko v určitých dialektoch alebo špecifických jazykoch môžu existovať určité skryté špecifické vlastnosti a významy.

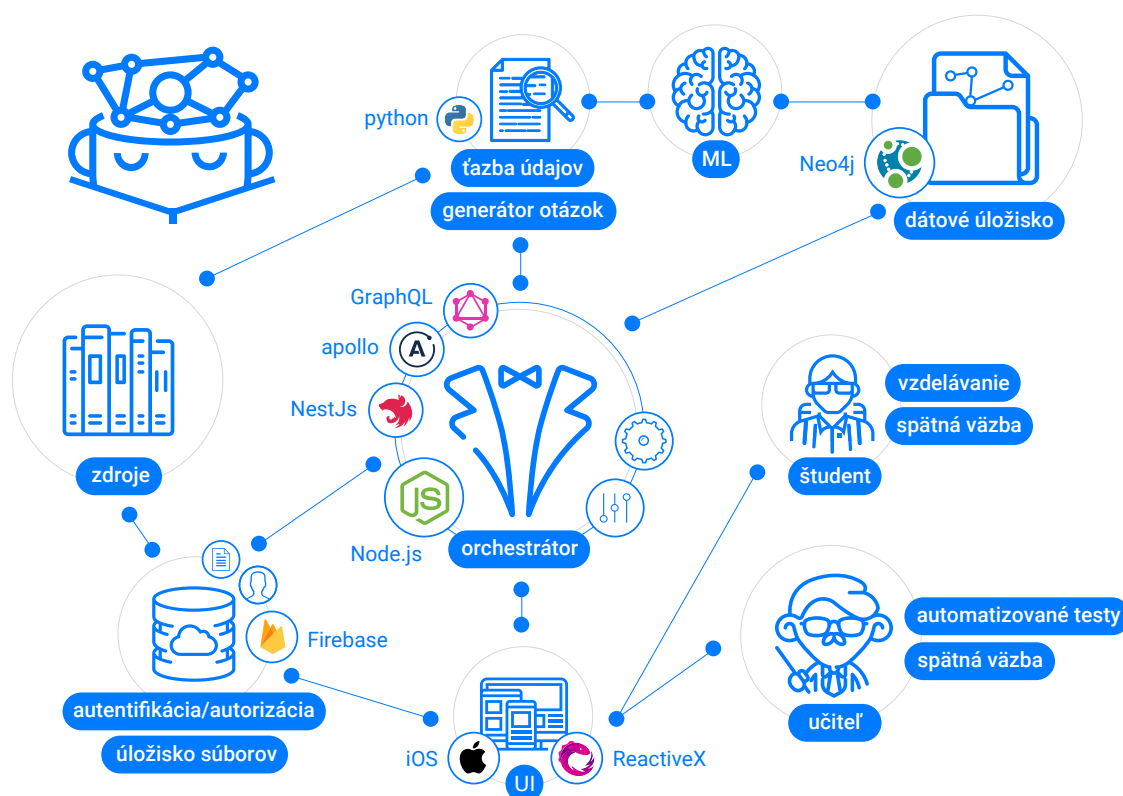
4 Implementácia

V implementačnej fáze výskumu sme sa venovali konkretizovaniu jednotlivých závislostí v navrhnutom konceptuálnom modeli systému. Naša snaha sa opierala o vytvorenie prvotného dôkazu funkčnosti navrhnutých komponentov a celkového systému ako takého. Tento princíp sa nazýva aj "proof of concept"(PoC). Implementácia systému teda pretvára konceptuálny návrh systému prostredníctvom požiadaviek identifikovaných pri analýze problému a pri určovaní správnej metodológie na technické riešenie úloh. Zaoberali sme sa teda technickým zabezpečením funkčnosti metód spracovania Slovenského jazyka a metód určených na spracovanie a štruktúrovanie informácií.

4.1 Architektúra systému

Diagram návrhu konceptuálneho modelu systému (3.1) sme rozšírili o detaily vo forme značenia jednotlivých modulov technologickými značkami reprezentujúcimi použité technológie, pomocou ktorých sme sprostredkovali implementáciu spomínaných metód. Finálna verzia architektúry navrhovaného systému je ilustrovaná na diagrame 4.1. Jednotlivé komponenty sú teda anotované základnými nástrojmi vďaka, ktorým je možné docieľiť implementáciu krokov spracovania textu a umožniť prístup k jednotlivým surovým, či analyzovaným zdrojom. Prepojenia v diagrame vyobrazujú interakciu medzi modulmi, ktorá môže byť reprezentovaná vzdialenými volaniami, volaniami vrámci konkrétneho stroja, alebo akciami inicializovanými používateľmi v používateľskom rozhraní. Pri implementácii takto definovanej architektúry sme sa zamerali hlavne na backendovú časť systému nakoľko táto časť je hlavným cieľom požadovaného systému tejto práce, no zaoberali sme sa aj spôsobom interakcie používateľov voči frontendovej časti systému. Riešenie frontendovej časti teda bonusovo obohacuje riešenie cieľov na-

vrhovaného systému. Popisky znázorňujúce účel komponentu alebo charakterizujúce danú časť architektúry definujú špecifickosť daného subjektu vystupujúceho v systéme. Ako je to z diagramu zjavné, systém dosahuje určitú hladinu komplexnosti a na implementáciu jednotlivých častí boli použité viaceré technológie z rôznych oblastí webových, mobilných, databázových či cloudových technológií.



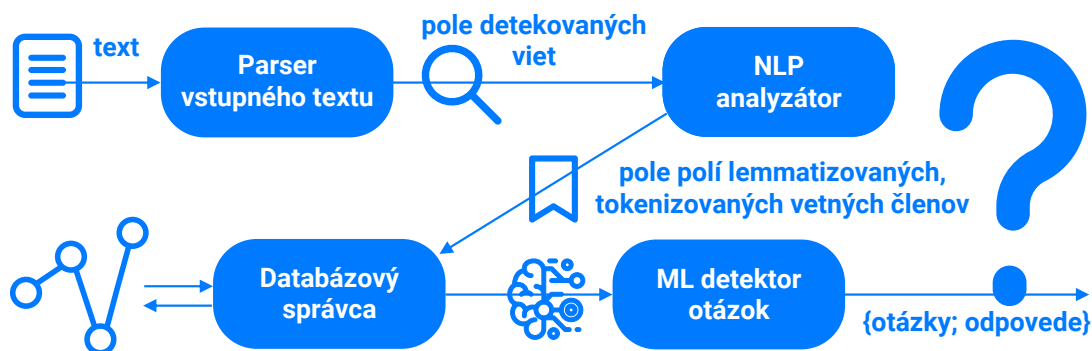
Obr. 4.1: Architektúra riešenia (vlastné spracovanie).

V nasledujúcich podkapitolách sa menovite venujeme popisu implementácie jednotlivých modulov architektúry systému. Detailný popis postupov pri implementácii dokáže priblížiť myšlienky zakotvené v zdrojovom kóde a rovnako pochopiť prepojenie modulov existujúcich v systéme. Postupným odhalením implementačných postupov a technológií v rámci modulov, ktoré integrujú logiku funkcií predstavíme systém ako celok. Postupnosť opisu modulov je zachovaná podľa dôležitosti konkrétnych modulov voči celkovému implementovanému systému.

4.2 Generátor otázok

Generátor otázok sme implementovali v jazyku *python*¹ vo verzii 3.7 a tento modul konkrétne sprostredkúva funkcionalitu generovania faktických prepozičných otázok s príznakom časového príznaku alebo príznaku miesta. Tento modul taktiež obsahuje komunikáciu s grafovou databázou a detekciu otázok prostredníctvom strojého učenia. Diagram 4.2 vizualizuje vnútornú štruktúru tohto nástroja na generovanie faktických otázok.

Vstupom do tohto modulu je text, ktorý vo svojej podstate skrýva určitý kontext a vetné konštrukcie. V prvom kroku sme implementovali *parser*, ktorý detekuje vetné konštrukcie v súvislom texte, túto funkcionalitu nám sprostredkoval *python* rámec s názvom NLP-Cube². Po detekcii viet v súvislom texte uvedený podporný NLP balíček vytvorí analýzu Slovenského jazyka. V tejto fáze analýzy dochádza k tokenizácii, lematizácii, značkovaniu časti reči, analýza závislostí a rozpoznávanie pomenovaných entít. Po týchto úkonoch spracovania textu dochádza k vytváraniu a identifikácii nových alebo existujúcich vrcholov a hrán v grafe pre slovenský jazyk. Všetky detekované a analyzované závislosti sa tak uložia do grafovo štruktúrovanej databázy.



Obr. 4.2: Štruktúra generátora otázok (vlastné spracovanie).

Posledným, najdôležitejším krokom generátora otázok je samozrejme proces detekovania a tvorby otázok a odpovedí, teda transformácia analyzovanej vety na určitú formu faktickej otázky, tak ako sme to popisovali v kapitole metodológia

¹Python: <https://www.python.org/>

²NLP-Cube: <http://opensource.adobe.com/NLP-Cube>

v časti o faktických otázkach (3.2.1). Implementovaný nástroj sa na základe uvedených skutočností stáva srdcom celkového implementovaného systému. Diagram 4.2 ilustruje aj formát vstupného textu v jednotlivých fázach spracovania vo forme pre systém surového textu, pole viet, pole polí analyzovaných lematických konštrukcií, graf až po formulované otázky s odpoveďami.

4.3 NLP analyzátor

Ako sme už uviedli NLP analyzátor je veľmi dôležitou súčasťou systému, nakoľko práve od tohto nástroja sa odvíja presnosť výsledku po spracovaní vstupného textu. V začiatkoch implementácie sme sa zameriavali na použitie morfológického analyzátoru s názvom *MorphoDiTa*³.

MorphoDiTa predstavuje morfológický slovník a open-source značkovač pre morfológickú analýzu textov prirodzeného jazyka. Vo svojej škále funkcionalít ponúka hlavne morfológickú analýzu ale aj morfológické generovanie, značkovanie a tokenizáciu a tento nástroj je navyše distribuovaný ako samostatný nástroj alebo knižnica spolu s natrénovanými lingvistickými modelmi. Napríklad v českom jazyku dosahuje *MorphoDiTa* výkonnosť priepustnosti okolo 10-200 tisíc slov za sekundu. *MorphoDiTa* je bezplatný softvér pod *Mozilla Public License 2.0* a jazykové modely sú sú bezplatné pre nekomerčné použitie. Problém nastáva pri špecifickosti nášho systému, ktorý je naviazaný na slovenský jazyk.

Existencia natrénovaného modelu je podmienená dostupnosťou pre konkrétny jazyk. *MorphoDiTa* takýto model pre podporu slovenského jazyka priamo neponúka a *Jazykovedný ústav Ľudovíta Štúra* sa nám prisľúbil, že nám poskytne takýto natrénovaný model, no doposiaľ sme nedostali priaznivú odpoveď. Vzhľadom na uvedené skutočnosti sme sa teda rozhodli natrénovať si takýto model, nakoľko knižnica *MorphoDiTa* vie sprostredkovať aj takúto možnosť. Natrénovaný testovací model však nedosahoval dostatočnú presnosť pri lematizácii a značkovaní morfológických charakteristík, čo bolo zjavne spôsobené nedostatkom použitých tréningových údajov.

Skúmali sme aj nástroj z našej *alma mater*, priamo z *Katedry elektroniky a multi-mediálnych telekomunikácií*, ktorý nesie názov *Dagger*⁴ (Hladek a kol., 2015). Tento

³MorphoDiTa: <http://ufal.mff.cuni.cz/morphodita>

⁴Dagger: <https://nlp.web.tuke.sk/nlpform>

nástroj je označený ako značkovač reči a morfológický analyzátor, čo je aj v skutočnosti pravda. Nástroj dokáže veľmi rýchlo analyzovať text morfológickou analýzou a označovať jednotlivé vetné členy správnymi morfológickými značkami, no nedokáže lematizovať jednotlivé slová. Nakoľko naše riešenie a návrh podmienili potrebou takéhoto spracovania, nástroj sme nemohli použiť priamo v jadre nášho systému. Bližšie sa tomuto nástroju budeme venovať v podkapitole s názvom *Orchestrátor* (4.5).

Po vyššie uvedených skúsenostiach sme sa dopracovali k inému komplexnejšiemu rámcu, ktorý nesie názov *NLP-Cube*⁵. *NLP-Cube* je voľne dostupný NLP rámec s podporou jazykov, ktoré sú zahrnuté v *UD Treebanks*⁶. *UD* alebo *Universal Dependencies* je rámcom pre krížovo lingvisticky gramatickú anotáciu a sprostredkúva hlavne otvorené komunitné úsilie s viac ako 200 prispievateľmi, ktorý doposiaľ vyprodukovali viac ako 100 lingvistických repozitárov vo viac ako 70-tich jazykoch. *UD* predstavuje projekt, ktorý vo všeobecnosti vyvíja multijazykovú konzistentnú anotáciu pre viacero jazykov, s cieľom uľahčiť viacjazyčný vývoj syntaktického analyzátoru, vzájomné jazykové vzdelávania a analyzovať výskum z perspektívy jazykovej typológie. Anotačná schéma je založená na vývoji (univerzálnych) *Stanfordských závislostí* (De Marneffe; MacCartney a kol., 2006; De Marneffe; Manning, 2008; De Marneffe; Dozat a kol., 2014), univerzálnych značkách spoločnosti *Google* v oblasti reči (Petrov a kol., 2011) a záujme o medzijazykovú analýzu morfológicko-syntaktických značkovačov (Zeman a kol., 2008). Všeobecnou filozofiou je tu poskytnúť univerzálny zoznam kategórií a usmernení, ktoré z určitou mierou uľahčia konzistentnú anotáciu podobných konštrukcií medzi jazykmi, pričom v prípade potreby umožnia rozšírenia špecifické pre konkrétny jazyk.

Vyššie uvedeny analyzátor nám teda sprostredkoval všetky potrebné nástroje na vytvorenie požadovanej a navrhutej analýzy. Pričom tabuľka 4.1 ilustruje testovanú presnosť značkovania na určitom segmentovanom texte, kde jednotlivé stĺpce reprezentujú charakteristiky identifikované na naplnenie atribútov už v metodológii uvádzaného *CoNLL* formátu. Rámec nám ponúkol možnosť výberu z dvoch natrénovaných modelov, z ktorých sme vybrali do našej implementácie práve *sk-1.0* nakoľko tento model podľa štatistík dosahuje vyššiu presnosť v oblasti lematizácie a tokenizácie, čo sú pre nás preferované charakteristiky.

⁵NLP-Cube: <http://opensource.adobe.com/NLP-Cube>

⁶UD Treebanks: <http://universaldependencies.org/>

Tabuľka 4.1: Presnosť značkovania segmentovaného textu

Model	Token	Veta	UPOS	XPOS	Značky	lemy	UAS	LAS
sk-1.0	99,97	86,0	95,82	82,3	78,43	90,35	88,83	85,69
sk-1.1	99,87	86,67	95,33	81,01	76,98	89,87	87,64	83,84

Použitím tohto rámca sme teda dokázali v implementácii zabezpečiť funkcie na spracovanie vstupného textu do nami požadovaných štruktúr a sprostredkovať tak základ pre ďalšiu analýzu s cieľom v generovaní faktických otázok.

Detekovanie a rozhodovanie, či sa daná konkrétna veta dá transformovať na zmysluplnú faktickú otázku s časovým príznakom alebo príznakom miesta sme implementovali prostredníctvom *python* nástroja *scikit-learn*⁷. Prostredníctvom tohto nástroja sme pripravili aj nástroje na trénovanie modelov so špecifickou klasifikáciou slúžiacou našej konkrétnej problematike.

4.4 Grafová platforma

Na uchovávanie a prístup k analyzovaným pojmom a vzťahom sme použili *Neo4j* grafovú platformu⁸, ktorá má za cieľ roširovanie schopností spoločnosti rozpoznať dôležitosť pretvárajúcich sa vzťahov a spojení prostredníctvom prechodov: od myšlienky, cez dizajn v logickom modeli, cez implementáciu vo fyzickom modeli, až po prevádzku pomocou dotazovacieho jazyka a vytrvalosti v škálovateľnom, spoľahlivom grafovom databázovom systéme. Uvedená platforma vo svojom portfóliu nasledujúce moduly, ktoré je možné prostredníctvom rôznych knižníc a prístupov využiť na uchovanie a dodatočnú analýzu údajov:

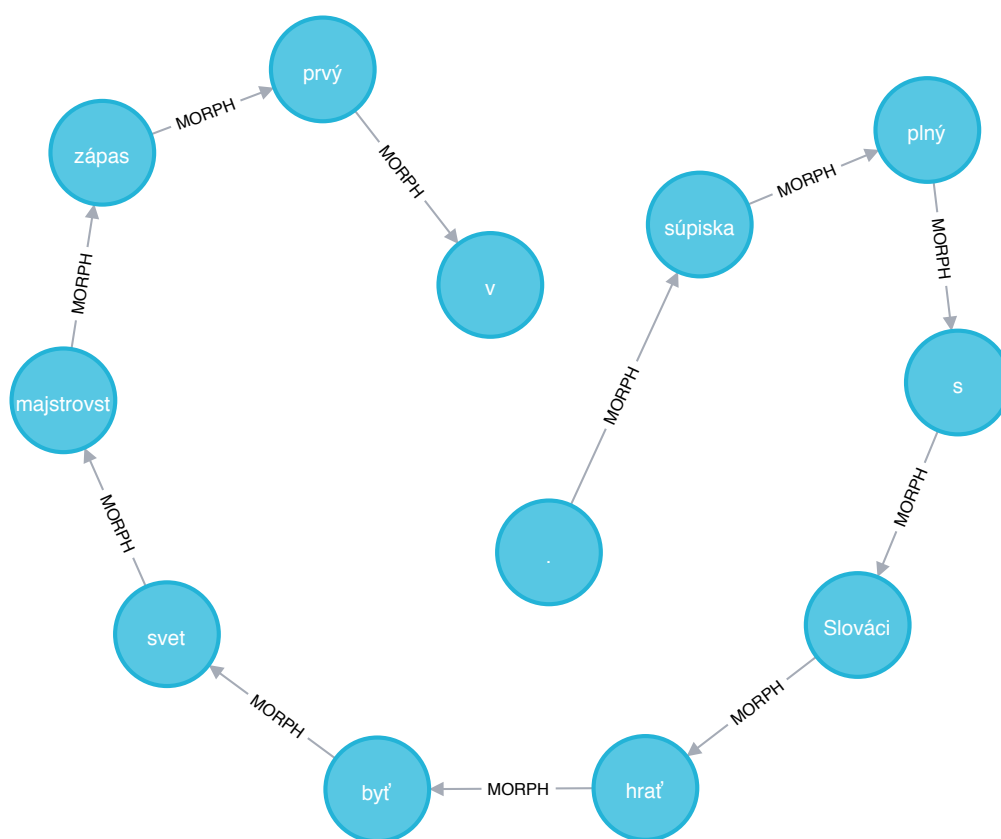
- **Neo4j natívna grafická databáza** - podpora pre transakčné aplikácie a analýzu grafovo.
- **Grafové analýzy** - nástroje na pomoc vedcom pri získavaní nových pohľadov na údaje.
- **Dátová integrácia** - nástroj na urýchlenie generovania tabuľkových údajov a veľkých dát do grafov.

⁷scikit-learn: <https://scikit-learn.org/stable/>

⁸Neo4j: <https://neo4j.com/>

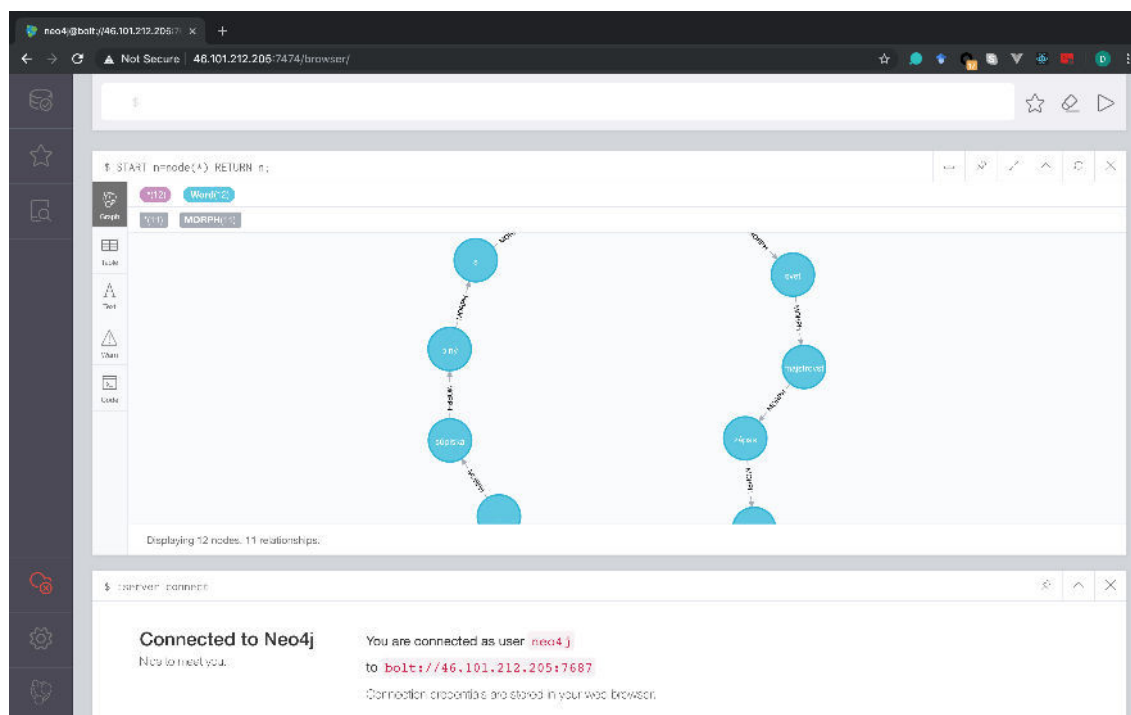
- **Dotazovací jazyk Cypher** - predstavuje most medzi analytckými nástrojmi a údajmi v grafe.
- **Vizualizácia a objavovanie grafov** - nástroj na sprehľadnenie a hľadanie charakteristík pri údajovej analýze.
- **Podniková architektúra** - podpora masívnych grafových štruktúr.

Všetku uvedené charakteristiky tejto platformy vytvárajú silnú podporu na ďalší rozvoj, no pri našej implementácii sme využili hlavne možnosť vytvorenia natívnej grafickej databázy. Nasledujúci diagram (4.3) v jednoduchosti vizualizuje výsledok integrácie grafovej databázy a implementácie funkcií na spracovanie a ukladanie údajov do uvedenej databázy.



Obr. 4.3: Reprezentácia vety z výpisu 3 v grafovej databáze po morfolologickej analýze (vlastné spracovanie).

Implementovaný NLP analyzátor teda vrámci jedného zo svojich modulov obsahuje funkcie na komunikáciu s inštanciou takejto databázy. Na dosiahnutie jednoduchšej interakcie s *Neo4j* databázou sme v jazyku *python* použili balíček *neomodel*⁹, ktorý vo svojej hlavnej podstate prináša funkcie mapovača objektov grafu. Po vytvorení inštancie grafovej databázy sa servri nám *Neo4j* platforma vo svojom inštalačnom balíčku ponúkla množstvo nástrojov v spojení s analýzou alebo spravovaním grafovej databázy. Nasledujúca snímka obrazovky (4.4) ilustruje *Neo4j Browser*, nástroj ktorý sprostredkúva širokú škálu funkcií a nastavení na správu, analýzu a dopytovanie. Použitím dotazovacieho jazyka *Cypher*¹⁰ a aplikácie *Neo4j Browser* je teda možné veľmi rýchlo nazerať a vyhľadávať v grafovej štruktúre.



Obr. 4.4: Neo4j - databázové rozhranie (vlastné spracovanie).

Integráciou vyššie uvedenej platformy naše riešenie získava nový pohľad na uchovávanie analyzovaných údajov s veľkou podporou grafovej analýzy a existenciou širokej základne grafových algoritmov na vytváranie ďalších špecifických analýz. Databáza je jednoducho škálovateľná a dotazovací jazyk *Cypher* prináša hlboký pohľad do uchovávaných údajov, čo pokladáme za veľký prínos pri plnení cieľov riešenia.

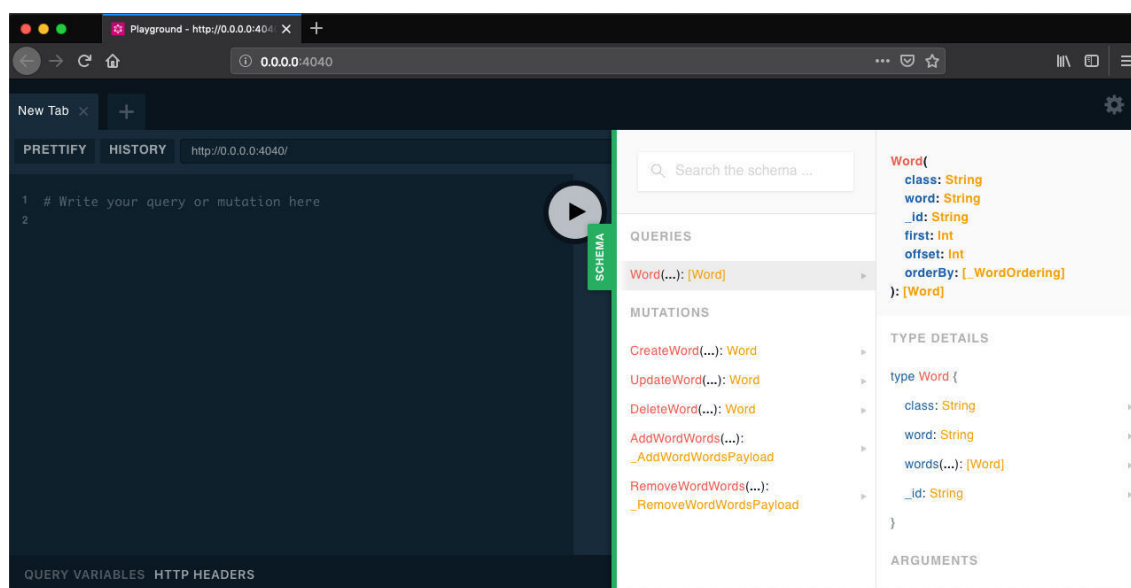
⁹neomodel: <https://github.com/neo4j-contrib/neomodel>

¹⁰Cypher: <https://neo4j.com/developer/cypher-query-language/>

4.5 Orkestrátor

Pri implementácii sme nezabudli ani na navrhovaný *Orkestrátor* modul, ktorý má svojimi funkciami sprostredkovať kontrolu nad ostatnými modulmi a vytvárať tak podporu škálovateľnosti v systéme. Návrh riešenia popisuje tento modulu ako API rozhranie, a tejto skutočnosti sme sa pri implementácii aj držali. Implementovali sme teda jednoduché základné aplikačné rozhranie na komunikáciu s jednotlivými časťami systému na platforme *Node.js*¹¹ prostredníctvom progresívneho rámca *NestJS*¹².

Do implementácie sme integrovali dotazovací jazyk *GraphQL*¹³ pre API a runtime pre plnenie dotazov s existujúcimi údajmi. Výhody, ktoré prinášajú princípy použité v tomto jazyku sú veľmi oceňovaným prídavkom k našej implementácii a rovnako sme sa snažili priniesť určitú technológiu používanú v moderných systémoch. *GraphQL* poskytuje úplný a zrozumiteľný popis údajov v API, dáva tak klientom možnosť dotazovať sa presne na to čo konkrétne potrebujú, uľahčuje vývoj API a sprístupňuje výkonné vývojárske nástroje.



Obr. 4.5: GraphQL API - dokumentačné a dotazovacie rozhranie (vlastné spracovanie).

¹¹Node.js: <https://nodejs.org/en/>

¹²NestJS: <https://nestjs.com/>

¹³GraphQL: <https://graphql.org/>

Príklad takéhoto vývojárskeho nástroja je ilustrovaný na snímke obrazovky 4.5. Tento nástroj sprístupňuje schému dátových modelov a opisne dokumentuje jednotlivé funkcie dostupné v API.

Orchestrátor v implementovanej verzii obsahuje dva základne moduly, a to modul na komunikáciu s *Neo4j* databázou a modul na sprístupnenie *nlp* nástrojov. V poslednom uvedenom module je rovnako možné nájsť funkcie na komunikáciu s už spomínaným značkovačom *Dagger*. Pri implementácii systému tieto funkcie slúžili hlavne k validácii, porovnávaní a overovaniu hodnovernosti jednotlivých značiek konkretizovaných na slovenský jazyk. Ďalšie implementačné rozšírenia tejto časti systému ponechávame na budúcu prácu zameranú na túto oblasť.

4.6 Používateľské rozhranie

Do doplnkového riešenia implementačnej časti spadá vytvorenie jednoduchého a prívetivého používateľského rozhrania. Vybrali sme si teda implementáciu mobilnej aplikácie v jazyku *Swift* na platforme *iOS*. Táto časť implementácie má slúžiť hlavne na vizualizáciu jednotlivých akcií, ktorý môže používateľ v navrhovanom a implementovanom systéme vykonať. Základ tejto mobilnej aplikácie sa ale skrýva v cloudovej platforme *Google Cloud Platform*¹⁴ a konkrétne v produkte *Firebase*¹⁵.

Firebase poskytuje funkcie, ako je napríklad analýza, dokumentačné databázy, zasielanie správ a hlásenia o zlyhaniach, takže sa developer môže veľmi jednoducho sústrediť na svojich používateľov. Z tejto platformy sme pri implementácii použili hlavne autentifikačného a autorizačného sprostredkovateľa na sprostredkovanie prihlasovania a registrácie používateľov do systému cez sociálne siete ako *Facebook*, *Twitter*, *Google* a *Github*. Taktiež sme využili služby súborového úložiska na uchovávanie súborov so vstupným textom na spracovanie od používateľov. Na uchovávanie informácií, s ktorými pracuje používateľské rozhranie sme rovnako použili databázu z uvedenej platformy s názvom *Firestore*¹⁶. Streamovanie živých údajov a úprav v definovanej dokumentačnej štruktúre sme podporili integráciou rámca *RxSwift*¹⁷ do platformy *iOS* na podporu princípov živých údajov a interak-

¹⁴Google Cloud Platform: <https://cloud.google.com/>

¹⁵Firebase: <https://firebase.google.com/>

¹⁶Firestore: <https://firebase.google.com/docs/firestore/>

¹⁷RxSwift: <https://github.com/ReactiveX/RxSwift>

tívneho používateľského rozhrania nástroja *ReactiveX*¹⁸. Takto implementovaná štruktúra mobilnej aplikácie môže v budúcej práci priniesť jednoduché rozšírenie funkcionalít implementovanej aplikácie.

Dizajn používateľského rozhrania rovnako vznikol v našej dielni na základe jednotlivých požiadaviek systému. Jeho vzhľad a funkčnosť bude detailnejšie prezentovaná vo výsledkoch našej práce. Používateľské rozhranie je teda vo svojej implementácii podporené silnými nástrojmi na zabezpečenie vysokej úrovne spokojnosti používateľa, čo tvorí základný kameň úspechu pri produkčnom nasadení aplikácie. Platforma, na ktorej je toto používateľské rozhranie postavené podporuje svojim SDK aj iné aplikačné platformy postavené na podpore webových, desktopových, mobilných, či hybridných aplikácií.

¹⁸ReactiveX: <http://reactivex.io/>

5 Dosiahnuté výsledky

5.1 Analýza a prieskum

Cieľom práce bolo v prvom kroku preskúmať a porovnať existujúce riešenia automatickej tvorby testov zo stromu znalostí. Uvedený cieľ sme sa snažili splniť z iacerých uhlov pohľadu a na základe poznatkov získaných pri skúmaní a porovnávaní existujúcich riešení navrhnúť, čo najoptimálnejší systém, ktorý by dokázal automaticky vytvoriť testovacie otázky z grafu (stromu) znalostí. Analyzovali sme rôzne projekty a výskumy, ktoré sa zaoberali určitou formou automatizácie generovania otázok a rovnako aj projektami, ktoré sa zameriavali na tvorbu grafov znalostí. V práci pojednávame o grafoch znalostí nakoľko stromy sú podmnožinou grafov, a keďže sme pri tvorbe analýzy a definovaní použitej metodológie došli k záveru, že znalostné stromy budú z určitého pohľadu optimálnejšie reprezentované grafom. Dôvodom pre voľbu grafov sa stal poznatok, že pri analýze a vytváraní vzťahov medzi jednotlivými vetnými členmi môže dôjsť aj k cyklickým závislostiam, a nakoľko z teórie grafov vyplýva, že stromy sú acyklické grafy, deklarovali sme ako základnú údajovú štruktúru pre náš systém práve grafy.

K jednotlivým riešeniam sme vždy sumarizovali výhody a nevýhody konkrétneho riešenia. Niektoré riešenia plne spĺňali určitú požiadavku požadovanú zo znenia zadania našej práce, v inej požiadavke však mali určité medzery. Iné zas pojednávali len o konkrétnej vlastnosti zadeného problému. Takouto analýzou sme získali rôznorodé pohľady na problematiku a mohli sme tak vybrať čo najoptimálnejšie metódy spacovania textu v prirodzenom jazyku a metódy uchovávaní vedomostí do údajových štruktúr. Analýza priniesla cenné skúsenosti potrebné pri návrhu systémov riešiacich danú problematiku. Existujúce riešenia predstavujú pre našu prácu základný kameň, a bez tvrdej práce výskumníkov a vývojárov by sme s určitosťou nedosiahli poznatky na spracovanie vlastného návrhu a imple-

mentácie.

Pri prvotných analýzach projektov sme analyzovali už existujúce a fungujúce systémy na automatickú tvorbu otázok, problémom pri nich väčšinou bolo, že otázky nevytvárali priamo z textu ale z nejakých vopred definovaných údajových štruktúr a nespĺňali tak určitú predstavovanú hladinu automatizácie tohto procesu. Ďalšie naopak extrakciu otázok z textu plne automatizovali, no ich údajové štruktúry nespĺňali zadané formy. V určitých systémoch sme zase naopak sledovali len ich údajovú štruktúru, teda akým spôsobom uchovávajú veľké množstvá údajov s dôrazom kladeným práve na vzťahy medzi týmito údajmi. Potvrdilo sa nám tvrdenie, že grafová reprezentácia ma obrovské množstvo výhod vzhľadom na uvádzanú a riešenú problematiku.

5.2 Návrh a implementácia systému

Implementácia navrhovaného systému je do určitej miery obširná a využíva množstvo technológií, ktoré obohacujú systém po funkčnej ale aj vývojárskej stránke. Cieľom implementácie bolo overiť a zvalidovať návrh vytvorený pri definovaní metodológie, ktorá by mala zabezpečiť splnenie jednotlivých cieľov, ktoré sa od systému vyžadujú. Po implementovaní jednotlivých modulov môžeme v krátkosti zhodnotiť, že automatické generovanie faktických otázok je splniteľná úloha vzhľadom na špecifickosť vybraného konkrétneho jazyka a použitie správnych metód spracovania textu. Generovanie faktických otázok vychádzajúce z implementácie však nie je dokonalým riešením a preto ďalšie rozšírenia a úpravu nechávame na budúcu prácu.

Analýza a prieskum existujúcich riešení vniesli do našej práce veľké množstvo poznatkov a spôsobov ako danú problematiku riešiť. Na základe týchto poznatkov sme vypracovali konceptuálny návrh a implementáciu nášho riešenia slúžiaceho na automatickú tvorbu testov z grafu znalostí. Navrhnutý systém predstavuje komplexné riešenie využiteľné vo viacerých oblastiach vzdelávania a do budúcnosti môže predstavovať zjavný prevrat v spôsobe získavania vedomostí z učebných textov a v spôsobe tvorby testovacích otázok pri snahe otestovať vedomosti študentov. Implementácia takto navrhnutého systému nebola vôbec triviálna úloha a zamerali sme sa hlavne na tvorbu znalostných grafov a generovanie faktických otázok s časovým príznakom a príznakom miesta. Pre extrakciu závis-

losti vetných členov vrámci určitých kontextov alebo diskurzov sme volili práve NLP ako princíp lexikálnej, syntaktickej a morfolologickej analýzy prirodzeného jazyka. NLP analyzátor sme konkretizovali na analýzu a detekciu vhodných vetných konštrukcií v slovenskom jazyku. Strojové učenie má v tomto projekte dôležitú úlohu, nakoľko sa používa už pri analýze štruktúr prirodzeného jazyka a potom pri detekcii vzorov viet vhodných na generovanie určitých typov otázok a odpovedí. Počas implementácie sme na údajoch získaných z korpusu slovenského jazyka sprístupneného *Jazykovedným ústavom Ľudovíta Štúra* natrénovali model na detekciu vetkých konštrukcií, ktoré by mohli byť transformované na otázky typu *Kedy* a *Kde*.

Rovnako sme navrhli a implementovali API na komunikáciu s jednotlivými časťami systému a na sprostredkovanie určitej úrovne škálovateľnosti. Taktiež sme vytvorili inštanciu grafovej databázy, do ktorej sa postupne hromadia vedomosti a kontext, v ktorom tieto informácie vystupujú. Ako bonus sme implementovali jednoduché používateľské rozhranie vo forme mobilnej aplikácie na sprístupnenie funkcionality systému.

5.3 Overenie a zhodnotenie generovania otázok

Výsledkom našej práce po implementácii je nástroj, ktorý dokáže tvoriť znalostné grafy, detekovať vetné konštrukcie vhodné na transformáciu na otázky s určitým kontextom a generovať konkrétne zmysluplné faktické otázky uvedeného typu. Implementovaný systém nie je dokonalým riešením uvedenej problematiky no tvorí úrodnú pôdu pre ďalší rozvoj, nakoľko dokazuje, že generovanie faktických otázok z akéhokoľvek textu je dosiahnuteľným cieľom. Funkčnosť systému generovať faktické otázky určeného typu dokážeme na príkladoch takto vygenerovaných testovacích otázok z určitých vzoriek textu pri názornej interpretácii výsledov v ďalšej kapitole.

Systém dokáže detekovať pomocou strojového učenia konkrétne vzory viet, ktoré sú vhodné na transformáciu na otázky s časovým príznakom a príznakom miesta. Dokáže teda vytvoriť otázky z viet, ktoré začínajú časovou predponou alebo predponou miesta a vytvoriť tak otázky začínajúce opytovacím zámenom *Kedy* a *Kde*. Vygenerované otázky systém posunie na výstup ako JSON aj spolu s odpoveďami na tieto otázky. Táto výstupná štruktúra môže byť teda použitá

v ďalších aplikáciách. Systém teda spĺňa požiadavku na generovanie testovacích otázok zo vstupného textu, no na základe aktuálnej implementácie, dokáže generovať otázky len určitého aktuálne detekovateľného typu.

5.4 Dokumentácia

K implementácii systému sme vypracovali aj systémové a používateľské príručky k jednotlivým modulom. Všetky potrebné informácie sú teda prístupné k zhliadnutiu v prílohách tejto práce. Tieto vypracované dokumenty popisujú systém z hľadiska štruktúry a použitých závislostí, ktoré sú reprezentované knižnicami alebo rámcami.

Repozitáre so zdrojovými kódmi obsahujú aj názorné ukážky datasetov, ktoré boli použité v konkrétnych tréningových postupoch. Prístupné sú rovnako aj použité, už natrénované modely použité vo fázach detekcie vzorov v textoch. Postupy opisujú kroky potrebné pri inštalácii, spustení a komunikácii s jednotlivými modulmi systému. Dokumentácia môže poslúžiť pri implementácii rozširujúcich návrhov alebo úprave už existujúcich súčastí návrhu a implementácie.

6 Interpretácia výsledkov

V tejto kapitole sa venujeme interpretácii a konkretizácii jednotlivých výsledkov nášho výskumu a implementácie. Výsledky nášho výskumu majú skôr kvalitatívny charakter než kvantitatívny. Hlavným cieľom tejto interpretácie bude ilustrovať reakcie a výstupy systému na príkladnú vstupnú vzorku textu, ktorú je možné zhliadnuť vo výpise 7. Text uvádzaný ako príklad vstupu reprezentuje určitú časť odborného textu pojednávajúceho o problematike počítačových sietí. Úlohou systému bude teda analyzovať daný text, vytvoriť grafovú štruktúru ako prvý výstup a vygenerovať faktické zmyslupné otázky.

V počítačových sieťach sa používajú rôzne modemy, ktorými sa zabezpečuje prevádzka na komutovaných a pevných linkách. V sieti hviezdicovej topológie pôsobí v centre siete centrálny uzol, ktorým môže byť prepínač (switch), v starších sieťach rozbočovač (hub) alebo, najmä pri bezdrôtových sieťach, opakovač (repeater). K centrálnemu uzlu sú pripojené stanice siete samostatnými linkami, najčastejšie pomocou symetrického kábla (krútená dvojlinka). Priporuche centrálného uzla je celá sieť vyradená z prevádzky.

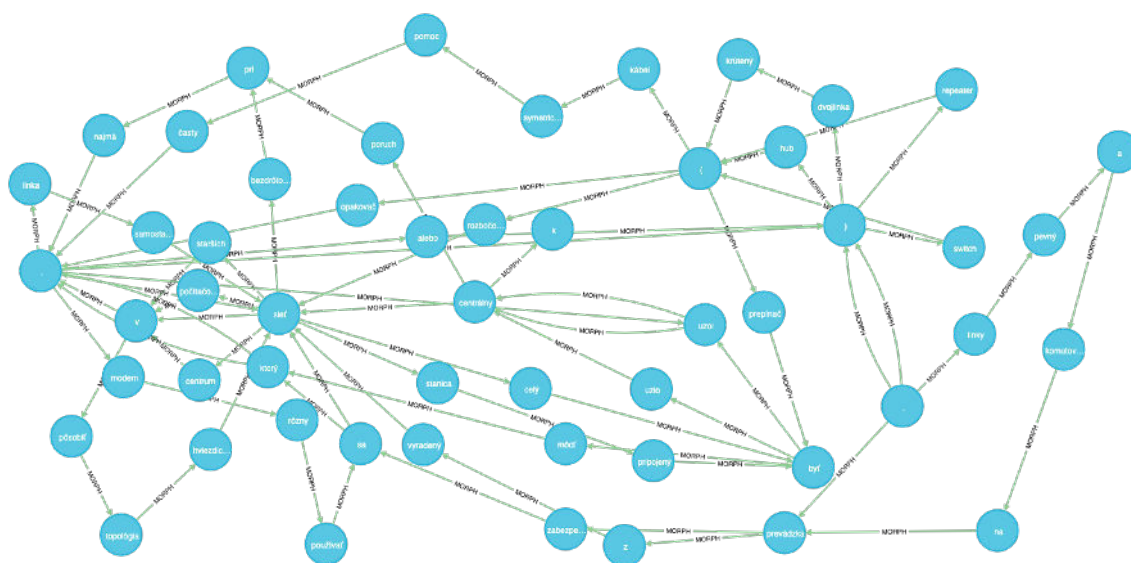
Výpis 7: Príklad odborného vstupného textu. (zdroj textu¹)

6.1 Transformácia vstupného textu na graf

V momente vstupu tohto príkladného textu sa tento text rozčlení na jednotlivé samostatné vety a spustí sa proces morfolologickej analýzy nad vetnými členmi všet-

¹Zdroj textu: <http://www.gymparnr.edu.sk/obsah/predmety/subory/informatika/siete.pdf>

kých detekovaných viet. Vo vnútri tohto procesu dochádza k lematizácii a tokenizácii, kde sa k jednotlivým slovám priradia konkrétne detekované charakteristiky. Na základe zistených vlastností daného slova a na základe jeho postavenia vo význame vety analyzátor kontaktuje grafovú databázu s požiadavkou zápisu alebo overenia existencie daných informácií v už existujúcom grafe. Takýmto spôsobom dôjde k obohateniu grafu o nové poznatky o jednotlivých lemach, nakoľko tie tvoria základnú entitu v nami navrhnutom grafe, následne sa doplnia závislosti danej konkrétnej lemy voči ostatným leмам. Diagram zobrazený na obrázku 6.1 reprezentuje grafovú štruktúru analyzovaného vstupného textu z výpisu 7.



Obr. 6.1: Graf vytvorený analýzou príkladu 7 (vlastné spracovanie).

Vytvorený graf obsahuje všetky unikátne lemy získané zo vstupného textu a rovnako všetky unikátne vzťahy existujúce v uvedenom vstupe. Na existujúcej štruktúre je následne možné vykonávať ďalšiu analýzu prostredníctvom dotazovacieho jazyka *Cypher* a prostredníctvom databázového rozhrania *Neo4j Browser* je možná spustiť aj rôzne grafové analytické algoritmy.

6.2 Vygenerované faktické otázky

Po vytvorení grafovej štruktúry sa analyzátor pokúsi detekovať naučené vetné konštrukcie vo vstupných vetách. Úkončenie detekcie vetných vzorov zadefinuje, či daná konkrétna veta obsahuje časovú predložkovú formu alebo predložkovú

formu miesta. Nasledujúci výpis (8) reprezentuje finálny výstup implementovaného systému na automatické generovanie faktických testovacích otázok.

```
{
  "questions": {
    "count": 3,
    "entries": [{
      "answer": "V počítačových sieťach",
      "question": "Kde sa používajú rôzne modemy,
                  ktorými sa zabezpečuje prevádzka
                  na komutovaných a pevných linkách?"
    },
    {
      "answer": "V sieti hviezdicovej topológie",
      "question": "Kde pôsobí v centre siete
                  centrálny uzol, ktorým môže byť prepínač
                  ( switch ), v starších sieťach rozbočovač
                  ( hub ) alebo, najmä pri bezdrôtových sieťach,
                  opakovač ( repeater )?"
    },
    {
      "answer": "Pri poruche centrálného uzla",
      "question": "Kedy je celá sieť vyradená z prevádzky?"
    }
  ]
}
```

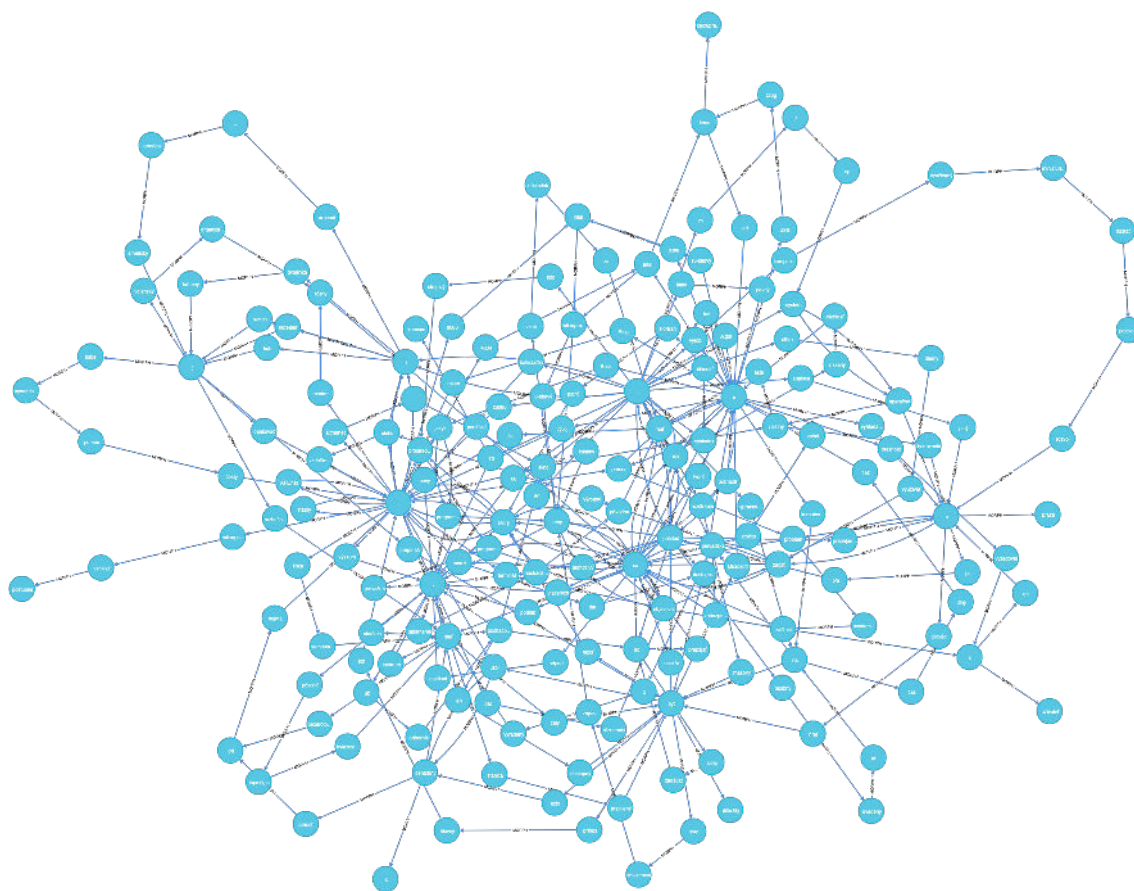
Výpis 8: Príklad vygenerovaných otázok. (vlastné spracovanie)

Generátor otázok teda dokázal z názorného vstupného textu vygenerovať 3 faktické otázky aj spolu s odpoveďami. Dokázali sme funkčnosť našej implementácie, v nasledujúcich podkapitolách sa budeme venovať doplnkovým výsledkom nášho výskumu.

6.3 Koreňový graf slovenského jazyka

Ďalším veľmi pozitívnym výsledkom je aj skutočnosť, že pri dostatočnom obohatení grafu napríklad všetkými možnými slovami a vetnými konštrukciami slovenského jazyka dokážeme za ideálnych podmienok získať skutočne komplexnú štruktúru, ktorá sa dokáže dostať až na úroveň existujúceho korpusu slovenského jazyka v počte základov slov reprezentovaných vrcholmi grafu. Navyše naša štruktúra bude obsahovať aj všetky možné tvary konkrétnych lemm aj s morfológickým popisom týchto tvarov a dokonca zo všetkými možnými prepojeniami a konštrukciami, v ktorých sa daný výraz alebo slovo vyskytuje v priamej reči.

Príklad komplexnejšieho grafu ilustruje obrázok 6.2. Takúto štruktúru sme pomenovali menom *Félix*, keďže takýto ideálny graf má svoju vlastnú existenciu a rozširoval by sa len na základe vstupov od čitateľov, ktorý by mali záujem o analýzu slovenských odborných či literárnych textov.



Obr. 6.2: Graf reprezentujúci komplexnejší diskurz (vlastné spracovanie).

Félix by teda za určitých ideálnych podmienok dokázal nahradiť korpus slovenského jazyka a stal by sa tak prvou voľbou pri analýze prirodzeného jazyka. Za určitých podmienok by dokázal podporovať aj iné jazyky nakoľko jeho štruktúra je vytvorená na základe CoNLL formátu², ktorý reprezentuje štandard pri NLP. *Félix* vo svojej podstate dokáže podporovať akýkoľvek jazyk, ktorý vie byť analyzovaný s rovnakou výstupnou štruktúrou. Keďže nástroj, ktorý používame na analýzu slovenského jazyka *NLP-Cube* v aktuálnej verzii podporuje analýzu 51 svetových jazykov, *Félix* by bol použiteľný pre všetky tieto jazyky. Určite by však stálo v úvahu integrovať všetky tieto jazyky do jednej multijazykovej štruktúry, ktorá by potom dokca dokázala zabezpečiť aj prekladové funkcie.

6.4 Mobilná aplikácia

Implementované používateľské rozhranie vo forme mobilnej aplikácie je rovnako dobrým kvalitatívnym výsledkom našej práce. Snímky obrazoviek z aplikácie znázornené na obrázku 6.3 vizualizujú základnú štruktúru aplikácie.



Obr. 6.3: Obrazovky aplikácie *Fairgrade* (vlastné spracovanie).

V aktuálnej verzii aplikácie sú podporované funkcie ako autentifikácia prostredníctvom e-mailu alebo sociálnych sietí alebo výber vstupných PDF súborov s textom určeným na analýzu a generovanie testovacích otázok tak ako je to znázornené na posledných dvoch snímkach sprava.

²CoNLL formát: <https://universaldependencies.org/format.html>

7 Ďalšia práca

V tejto časti práce chceme konkretizovať a poukázať na možné vylepšenia a rozšírenia našej práce. Na základe nasledujúcich poznatkov podávame budúcim čitateľom podnety k jednoduchšej identifikácii určitých problémov alebo bodov, ktorých riešenie by prispelo k prospešnému vývoju navrhnutého systému.

7.1 Rozšírenie detekcie otázkových vzorov

Najväčším prínosom pre našu prácu by bolo rozšírenie modelu na detekciu vzorov potenciálnych otázkových viet. Pridanie podpory na detekovanie a klasifikáciu aj iných typov otázkových konštrukcií jednoznačne obohatí naše riešenie a svojím spôsobom ho priblíži bližšie k dokonalosti vzhľadom na cieľ, ktorý je jasne stanovený smerom ku automatickému generovaniu faktických otázok z akéhokoľvek odborného alebo bežného textu. V nasledujúcich krokoch implementácie a rozširovania určite odporúčame obohatiť model o detekciu a klasifikáciu otázok kladených na podstatné mená vystupujúce vo vetných konštrukciách s určitým kontextom alebo detekciu viet s potenciálnou transformáciou na otázky cielené na podriadené doložky alebo prístavky.

Vzhľadom na komplexnosť identifikácie možných otázkových transformácií z vetných konštrukcií odporúčame spoluprácu s nejakým skúseným jazykovedcom, ktorý by disponoval obsiahnejšími vedomosťami o vetných konštrukciách v konkrétnom riešenom jazyku. Skúsenosti znalého jazykovedca dokážu byť v tejto konkrétnej problematike na nezaplatenie, nakoľko po technologickej stránke je všetko pripravené na to aby boli do natrénovaného modulu zaintegrované aj ďalšie možné vzory. Myslíme si, že aj plná podpora napríklad zo strany vedcov a technikov z *Jazykovedného ústavu Ľudovíta Štúra* by projektu len a len prospela a riešenie problematiky automatického generovania testovacích otázok by nabralo nové rozmery.

Spomínaní vedci a technici jednoznačne po skúsenostiach s korpusom slovenského jazyka a po skúsenostiach s prácou s morfológickými databázami určite majú čo povedať k riešenej problematike a k navrhovanému riešeniu.

Ďalším z možných rozšírení generátora otázok by bolo vytvorenie databázy alebo určitej obširnejšej údajovej skupiny hodnotených otázok prijateľných a neprijateľných osobitne. Na takýchto tréňovaných dátach by sa dal následne postaviť nástroj na automatické rozhodovanie o prijateľnosti konkrétnych otázok a prípadne by sa to s určitosťou dalo kombinovať s prijateľnosťou odpovedí vo vzťahu ku konkrétnej odpovedi. Týmto spôsobom by bol systém obohatený o ďalší veľmi užitočný proces na určenie či vygenerované otázky sú naozaj zmysluplné a dosahujú určitú hladinu hodnotenia. Na základe takýchto poznatkov by sa dala ilustrovať náročnosť otázok a náročnosť odpovede.

7.2 Obohatenie grafovej analýzy

Grafová analýza nad vytvorenými grafmi je rozľahlá zelená lúka, na ktorej sa dá stavať. Napríklad určovanie dôležitosti určitého pojmu predstavuje veľmi užitočnú štatistiku, ktorá by obohatila systém o novú funkcionálnu, počet prepojených vrcholov vzhľadom na kontextové identifikátory dokáže v jednoduchosti indikovať úroveň dôležitosti konkrétneho pojmu.

Ďalším z možných rozšírení s prepojením na grafovú štruktúru by mohla byť integrácia multijazykovej podpory v prostredí jedného grafu. Takáto integrácia by dokázala priblížiť a v ideálnom prípade aj zjednotiť jazyky v jednej konkrétnej štruktúre, čo by mohlo viesť k extrakcii napríklad rozdielov a porovnaní vo vybraných jazykoch.

Kategorizácia vzťahov podľa kontextu, v ktorom sa vyskyvali by prinieslo možnosť tvorby podgrafov, ktoré by dokázali popisovať konkrétnejšie kategórie. Teda ak by používateľ hľadal otázky na konkrétny pojem v určitej kategórii, dostal by prijateľnejšie výsledky než by sa dopytoval nad celou množinou kontextov. Pri definovaní tejto logiky by mohol vzniknúť aj spôsob alebo proces zjednotenia podgrafov alebo celých grafov do jednej konkrétnej kategórie.

So všetkými uvedenými rozšíreniami sa taktiež spája rozšírenie rozhrania *Orchestrátora* na ucelené sprostredkovanie a monitorovanie, prípadné škálovanie týchto jednotlivých rozšírení, či už na strane generátora otázok alebo obohatenia analýzy

grafovej štruktúry. Tento nástroj by tak naplnil svoje poslanie a enkapsuloval by tak funkcionálnosť hlbších modulov systému.

7.3 Multi-platformové používateľské rozhranie

Z pohľadu používateľa by dobudovanie používateľského rozhrania o všetky podporované funkcie systému prinieslo najväčší ošoh. Nakoľko aj laik by dokázal manipulovať so zložitými štruktúrami a analýzami prirodzeného jazyka. A dokonca ak by dané používateľské rozhranie bolo implementované multi-platformovo, používateľ by sa dočkal ľahkého prístupu a čo je hlavné funkcie systému by boli dostupné v rôznych inštitúciách, ktoré sa zaoberajú vzdelávaním.

Vznik e-learningovej platformy na princípoch automatickej analýzy odborných textov a následnom automatickom generovaní testovacích otázok by úplne zmenil svet vzdelávania. Preto by malo rozšírenie používateľského rozhrania predstavovať dôležitú súčasť ďalšej budúcej práce. Z uvedených dôvodov odporúčame zakomponovať budovanie používateľského rozhrania do budúcej práce, nakoľko práve to môže byť veľkým prínosom v sprístupnení funkcionality a danej problematiky aj bežným ľuďom. Študenti a učitelia by získali jednoduchý prístup k vedomostiam a k hodnoteniu vedomostí na základe analyzovaných vedomostí z už existujúcej literatúry.

8 Záver

Automatické generovanie testovacích otázok zo znalostných grafov je problematika, ktorou sme sa v tejto práci zaoberali. Svoju pozornosť sme venovali vytvoreniu generátora faktických otázok určeného typu a rovnako aj vytvoreniu grafovej štruktúry vedomostných grafov. Všetky potrebné návrhy a implementácia vedú spoločne k jedinému jasnému cieľu, ktorým je získanie zmysluplných faktických otázok z odborných textov. Za účelom splnenia nášho cieľa sme vytvorili prehľad a prieskum existujúcich riešení zaoberajúcich sa problematikou automatického generovania otázok a rovnako aj riešení, ktoré sa zaoberajú uchovávaním vedomostí do grafových štruktúr. Prímosom uvedenej aktivity je skutočnosť, že na základe analýzy riešení v rôznych projektoch a výskumoch sme získali prehľad v danej problematike a nadobudli vedomosti ako uvedený problém optimálne vyriešiť.

Po analýze existujúcich riešení sme spracovali konceptuálny návrh a implementáciu systému na automatické generovanie faktických otázok z odborných textov. Prímosom nášho riešenia je existujúca implementácia navrhnutého systému, ktorá bola otestovaná na určitej vzorke vstupných textov obsahujúcich konkrétne typové vzory vetných konštrukcií. Prínosom v tejto oblasti je aj skutočnosť vytvorenia novej grafovej štruktúry s názvom *Félix*, ktorá slúži na uchovávanie vedomostí spolu s morfológickými a kontextovými vzťahmi, testovanej na lexikálnych vzoroch slovenského jazyka. Implementovaný generátor otázok dokáže detekovať a transformovať vetné konštrukcie, obsahujúce časový predponový vzor alebo predponový vzor miesta, na faktické otázky začínajúce na opytovacie zámená *Kedy ... ?* a *Kde ... ?*. K riešeniu sme vypracovali dokumentáciu štruktúry implementácie a použitých technológií, ktorá sa nachádza roztriedená podľa konkrétnych modulov systému v prílohách. V tomto bode riešenia sme dospeli k splneniu definovaných pokynov na vypracovanie.

Dodatkovým riešením je návrh a implementácia používateľského rozhrania

vo forme mobilnej aplikácie postavenej na platforme *iOS*. Bola vytvorená na sprostredkovanie vizualizácie akcií a primárne slúži ako nástroj na zber vstupných textov od používateľa. Aplikácia má v budúcnosti slúžiť ako súčasť navrhovanej e-learningovej platformy. Vytvorili sme teda jednoduché a pohľadné používateľské rozhranie slúžiace na sprístupnenie funkcionalít navrhovaného systému. Prínos tejto časti riešenia je podložený potrebou existencie nástroja na sprístupnenie zložitých NLP analýz a transformácií textu aj bežným používateľom. Vzhľadom na vyššie uvedené skutočnosti je dokončenie implementácie používateľského rozhrania otvorené ďalším budúcim riešeniam.

Pri návrhu a implementácii sme získali mnoho nových technických a analytických poznatkov z oblastí NLP a NLG, ktoré predstavujú vzácne vedomosti tvoriace základ pre prácu a výskum v tejto oblasti informačných technológií. Riešením uvádzanej problematiky sme sa dostali aj k informáciám, nám doposiaľ neznámym, a vývoj softvéru pre nás otvoril nové dvere v oblasti spracovania prirodzeného jazyka.

Literatúra

- JÚLS, 2019. Morfológická anotácia textov Slovenského národného korpusu. *Návod k používaniu Slovenského národného korpusu*.
- BLUMAUER, Andreas, 2018. *Knowledge Graphs are on the rise*. Ed. VIENNA, Semantics. Dostupné tiež z: <https://2018.semantics.cc/knowledge-graphs-are-rise>. [online: 2-November-2018].
- BOROȘ, Tiberiu; DUMITRESCU, Stefan Daniel; BURTICA, Ruxandra, 2018. NLP-Cube: End-to-end raw text processing with neural networks. In: *NLP-Cube: End-to-end raw text processing with neural networks. Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, s. 171–179.
- L. ŠTÚRA SAV, Bratislava: Jazykovedný ústav, 2018. Slovenský národný korpus – prim-8.0-public-sane. Dostupné tiež z: <http://korpus.juls.savba.sk>.
- RSTUDIO, Support, 2018. *Using Sweave and knitr*. Ed. DOCUMENTATION, RStudio. Dostupné tiež z: <https://support.rstudio.com/hc/en-us/articles/200552056-Using-Sweave-and-knitr>. [online: 4-November-2018].
- UNESCO, 2017. *Six ways to ensure higher education leaves no one behind*. Ed. REPORT, Global Education Monitoring. Dostupné tiež z: <http://unesdoc.unesco.org/images/0024/002478/247862E.pdf>. [online: 2-November-2018].
- EOM, Ju-Seong; LIM, Ui-Kyu; LEE, Jae-Young, 2016. Automatic Generation System of True or False Questions. *Advanced Science and Technology Letters*. Roč. 135, s. 46–49.
- HERNÁNDEZ, Daniel; HOGAN, Aidan; RIVEROS, Cristian; ROJAS, Carlos; ZEREGA, Enzo, 2016. Querying wikidata: Comparing sparql, relational and graph databases. In: *Querying wikidata: Comparing sparql, relational and graph databases. International Semantic Web Conference*, s. 88–103.

- MAZIDI, Karen; TARAU, Paul, 2016. Automatic Question Generation: From NLU to NLG. In: MICARELLI, Alessandro; STAMPER, John; PANOURGIA, Kitty (ed.). *Intelligent Tutoring Systems*. Cham: Springer International Publishing, s. 23–33. ISBN 978-3-319-39583-8.
- THOMPSON, Bryan; PERSONICK, Mike; CUTCHER, Martyn, 2016. The Bigdata® RDF graph database. In: *The Bigdata® RDF graph database. Linked Data Management*. Chapman a Hall/CRC, s. 221–266.
- HLADEK, Daniel; STAS, Jan; JUHAR, Josef, 2015. Morphological analysis of the slovak language. *Advances in Electrical and Electronic Engineering*. Roč. 13, č. 4, s. 289–294.
- DE MARNEFFE, Marie-Catherine; DOZAT, Timothy; SILVEIRA, Natalia; HAVE-RINEN, Katri; GINTER, Filip; NIVRE, Joakim; MANNING, Christopher D, 2014. Universal Stanford dependencies: A cross-linguistic typology. In: *Universal Stanford dependencies: A cross-linguistic typology*. LREC. Zv. 14, s. 4585–92.
- DONG, Xin Luna; GABRILOVICH, Evgeniy; HEITZ, Jeremy; HORN, Wilko; LAO, Ni; MURPHY, Kevin; STROHMANN, Thomas; SUN, Shaohua; ZHANG, Wei, 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In: *Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, s. 601–610. Dostupné tiež z: <http://www.cs.cmu.edu/~nlao/publication/2014.kdd.pdf>. Evgeniy Gabrilovich Wilko Horn Ni Lao Kevin Murphy Thomas Strohmann Shaohua Sun Wei Zhang Jeremy Heitz.
- VRANDEČIĆ, Denny; KRÖTZSCH, Markus, 2014. Wikidata: a free collaborative knowledge base.
- PUJARA J., Miao H., 2013. Knowledge Graph Identification. *The Semantic Web – ISWC*. Roč. 8218, s. 542–557. Dostupné tiež z: https://link.springer.com/chapter/10.1007/978-3-642-41335-3_34. [online: 4-November-2018].
- KIMMIG, Angelika; BACH, Stephen H.; BROECHELER, Matthias; HUANG, Bert; GETOOR, Lise, 2012. A Short Introduction to Probabilistic Soft Logic. In: *A Short Introduction to Probabilistic Soft Logic. NIPS Workshop on Probabilistic Programming: Foundations and Applications*. Dostupné tiež z: <https://lincs.soe.ucsc.edu/node/39>. [online: 4-November-2018].

- PIWEK, Paul; BOYER, Kristy Elizabeth, 2012. Varieties of question generation: introduction to this special issue. *Dialogue & Discourse*. Roč. 3, č. 2, s. 1–9.
- RUS V. Wyse B., Piwek P., 2012. A detailed account of the first question generation shared task evaluation challenge. *Dialogue Discourse*. Č. 3, s. 177–204. Dostupné tiež z: <http://dad.uni-bielefeld.de/index.php/dad/article/view/1474>. [online: 4-November-2018].
- HEILMAN, Michael, 2011. Automatic factual question generation from text. *Language Technologies Institute School of Computer Science Carnegie Mellon University*. Roč. 195.
- PETROV, Slav; DAS, Dipanjan; MCDONALD, Ryan, 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.
- BROECHELER M. Mihalkova L., Getoor L., 2010. Probabilistic Similarity Logic. *Uncertainty in Artificial Intelligence*, s. 73–82. Dostupné tiež z: <https://dl.acm.org/citation.cfm?id=3023558>. [online: 4-November-2018].
- FOSTER, Jennifer, 2010. cba to check the spelling investigating parser performance on discussion forum posts. In: *cba to check the spelling investigating parser performance on discussion forum posts*. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, s. 381–384.
- HEILMAN, Michael; SMITH, Noah A., 2010. Rating Computer-generated Questions with Mechanical Turk. In: *Rating Computer-generated Questions with Mechanical Turk*. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Los Angeles, California: Association for Computational Linguistics, s. 35–40. CSLDAMT '10. Dostupné tiež z: <http://dl.acm.org/citation.cfm?id=1866696.1866701>.
- INDURKHIA, Nitin; DAMERAU, Fred J., 2010. *Handbook of Natural Language Processing*. 2nd. Chapman & Hall/CRC. ISBN 1420085921, 9781420085921.
- KOK, Stanley; BROCKETT, Chris, 2010. Hitting the right paraphrases in good time. In: *Hitting the right paraphrases in good time*. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, s. 145–153.

- GRUN B., Zeileis A., 2009. Automatic generation of exams in R. *Journal of Statistical Software*. Č. 29, s. 1–14. Dostupné tiež z: <https://ro.uow.edu.au/cgi/viewcontent.cgi?article=3032&context=commpaper>. [online: 4-November-2018].
- HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome, 2009. *The elements of statistical learning: data mining, inference, and prediction*, Springer Series in Statistics. Springer New York.
- HEILMAN, Michael, 2009. Question Generation via Overgenerating Transformations and Ranking. In: *Question Generation via Overgenerating Transformations and Ranking*.
- MOHLER, Michael; MIHALCEA, Rada, 2009. Text-to-text semantic similarity for automatic short answer grading. In: *Text-to-text semantic similarity for automatic short answer grading. Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, s. 567–575.
- BAILEY, Stacey; MEURERS, Detmar, 2008. Diagnosing meaning errors in short answers to reading comprehension questions. In: *Diagnosing meaning errors in short answers to reading comprehension questions. Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, s. 107–115.
- BOLLACKER, Kurt; EVANS, Colin; PARITOSH, Praveen; STURGE, Tim; TAYLOR, Jamie, 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In: *Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. Vancouver, Canada: ACM, s. 1247–1250. SIGMOD '08. ISBN 978-1-60558-102-6. Dostupné z DOI: 10.1145/1376616.1376746.
- BUNEMAN, Peter; CHENEY, James; TAN, Wang-Chiew; VANSUMMEREN, Stijn, 2008. Curated databases. In: *Curated databases. Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, s. 1–12.
- CALLISON-BURCH, Chris, 2008. Paraphrasing and translation.

- DE MARNEFFE, Marie-Catherine; MANNING, Christopher D, 2008. The Stanford typed dependencies representation. In: *The Stanford typed dependencies representation. Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, s. 1–8.
- GRAESSER, Art; RUS, Vasile; CAI, Zhiqiang, 2008. Question classification schemes. In: *Question classification schemes. Proc. of the Workshop on Question Generation*, s. 10–17.
- NIELSEN, Rodney D; WARD, Wayne; MARTIN, James H; PALMER, Martha, 2008. Extracting a representation from text for semantic analysis. *Proceedings of ACL-08: HLT, Short Papers*, s. 241–244.
- SNOW, Rion; O’CONNOR, Brendan; JURAFSKY, Daniel; NG, Andrew Y., 2008. Cheap and Fast—but is It Good?: Evaluating Non-expert Annotations for Natural Language Tasks. In: *Cheap and Fast—but is It Good?: Evaluating Non-expert Annotations for Natural Language Tasks. Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii: Association for Computational Linguistics, s. 254–263. EMNLP ’08. Dostupné tiež z: [http : //dl.acm.org/citation.cfm?id=1613715.1613751](http://dl.acm.org/citation.cfm?id=1613715.1613751).
- SUKKARIEH, Jana; BOLGE, Eleanor, 2008. Leveraging C-rater’s automated scoring capability for providing instructional feedback for short constructed responses. In: *Leveraging C-rater’s automated scoring capability for providing instructional feedback for short constructed responses. International Conference on Intelligent Tutoring Systems*, s. 779–783.
- ZEMAN, Daniel; RESNIK, Philip, 2008. Cross-language parser adaptation between related languages. In: *Cross-language parser adaptation between related languages. Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.
- AUER, Sören; BIZER, Christian; KOBILAROV, Georgi; LEHMANN, Jens; CYGANIAK, Richard; IVES, Zachary, 2007. Dbpedia: A nucleus for a web of open data. In: *Dbpedia: A nucleus for a web of open data. The semantic web*. Springer, s. 722–735.
- BOLLACKER, Kurt; TUFTS, Patrick; PIERCE, Tomi; COOK, Robert, 2007. A platform for scalable, collaborative, structured information integration. In: *A plat-*

form for scalable, collaborative, structured information integration. Intl. Workshop on Information Integration on the Web (IIWeb'07).

GABRILOVICH, Evgeniy; MARKOVITCH, Shaul a kol., 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: *Computing semantic relatedness using wikipedia-based explicit semantic analysis. IJcAI. Zv. 7*, s. 1606–1611.

DE MARNEFFE, Marie-Catherine; MACCARTNEY, Bill; MANNING, Christopher D a kol., 2006. Generating typed dependency parses from phrase structure parses. In: *Generating typed dependency parses from phrase structure parses. Lrec. Zv. 6*, s. 449–454.

HALEVY, Alon; FRANKLIN, Michael; MAIER, David, 2006. Principles of dataspace systems. In: *Principles of dataspace systems. Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, s. 1–9.

NENKOVA, Ani; MCKEOWN, Kathleen, 2006. *Understanding the process of multi-document summarization: content selection, rewriting and evaluation*. Columbia University.

STRUBE, Michael; PONZETTO, Simone Paolo, 2006. WikiRelate! Computing semantic relatedness using Wikipedia. In: *WikiRelate! Computing semantic relatedness using Wikipedia. AAI. Zv. 6*, s. 1419–1424.

FINKEL, Jenny Rose; GRENAGER, Trond; MANNING, Christopher, 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In: *Incorporating non-local information into information extraction systems by gibbs sampling. Proceedings of the 43rd annual meeting on association for computational linguistics*, s. 363–370.

KRÖTZSCH, Markus; DENNY VRANDECIC, Denny Vr; VÖLKEL, Max, 2005. Wikipedia and the semantic web-the missing links. In: *Wikipedia and the semantic web-the missing links. Proceedings of Wikimania 2005*.

ERKAN, Günes; RADEV, Dragomir R, 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*. Roč. 22, s. 457–479.

- DORR, Bonnie; ZAJIC, David; SCHWARTZ, Richard, 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In: *Hedge trimmer: A parse-and-trim approach to headline generation. Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*, s. 1–8.
- MITKOV, Ruslan a kol., 2003. Computer-aided generation of multiple-choice tests. In: *Computer-aided generation of multiple-choice tests. Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing*.
- BERNERS-LEE, Tim; HENDLER, James; LASSILA, Ora a kol., 2001. The semantic web. *Scientific american*. Roč. 284, č. 5, s. 28–37.
- (US), National Reading Panel; CHILD HEALTH, National Institute of; (US), Human Development, 2000. *Report of the National Reading Panel: Teaching children to read: An evidence-based assessment of the scientific research literature on reading and its implications for reading instruction: Reports of the subgroups*. National Institute of Child Health a Human Development, National ...
- KNIGHT, Kevin; MARCU, Daniel, 2000. Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI*. Roč. 2000, s. 703–710.
- ALLEN, James, 1995. *Natural Language Understanding (2Nd Ed.)* Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc. ISBN 0-8053-0334-0.
- MILLER, George A; BECKWITH, Richard; FELLBAUM, Christiane; GROSS, Derek; MILLER, Katherine J, 1990. Introduction to WordNet: An on-line lexical database. *International journal of lexicography*. Roč. 3, č. 4, s. 235–244.
- BONDY, John Adrian; MURTY, Uppaluri Siva Ramachandra a kol., 1976. *Graph theory with applications*. Citeseer.

Zoznam skratiek

AJAX Asynchronous JavaScript and XML.

API Application Programming Interface.

AUC Area Under Curve.

CoNLL Computational Natural Language Learning.

DSSP Dialog System for Structured Programming.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

JDK Java Development Kit.

JRE Java SE Runtime Environment.

JSON JavaScript Object Notation.

LOB Large Object.

ML Machine learning.

MLN Markov Logic Networks.

SQL Metaweb Query Language.

NELL Never-Ending Language Learning.

NLG Natural Language Generation.

NLP Natural Language Processing.

NLU Natural Language Understanding.

PDF Portable Document Format.

PoC Proof of Concept.

PSL Probabilistic Soft Logic.

QG Question Generator.

RDF Resource Description Framework.

SDK Software Development Kit.

UI User interface.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

Slovník

Algoritmus je konečná postupnosť presne definovaných inštrukcií na splnenie určitej úlohy.

Anafora je jazykový prostriedok, ktorý sa vyskytuje spravidla v začiatočnom postavení (vety, odseku, prejavu a podobne).

Analýzátor je zariadenie analyzujúce nejaký proces alebo jav, najmä zložky meraných veličín, napr. analyzátor webových stránok, analyzátor golfového švih, analyzátor čistoty vody, amplitúdový analyzátor.

Analýza je metóda práce používajúca rozbor a rozkladanie.

Antonymum alebo opozitum je slovo s opačným lexikálnym významom.

Apozícia je v jazykovede: prístavok.

Asociácia je združovanie; spájanie.

Aspekt je stanovisko, hľadisko uplatňované pri posudzovaní určitej veci, javu a podobne.

Automatizácia je etapa rozvoja techniky, pre ktorú je charakteristické uskutočňovanie výrobných, riadiacich a iných procesov bez priameho zásahu človeka.

Binárny je dvojčlenný, podvojný, zložený z dvoch zložiek.

Databáza je množina štruktúrovaných dát alebo informácií uložených v počítačovom systéme, takým spôsobom, že počítačový program alebo človek môže použiť dopytovací jazyk na získavanie týchto informácií.

Dedukcia alebo odvodzovanie je proces postupných transformácií výrazov pomocou odvodzovacích pravidiel.

Definícia je vymedzenie významu nejakého výrazu.

Disambiguácia je časť (väčšinou automatického) procesu anotácie jazykových dát, ktoré vstupujú do korpusu.

Diskurz je jednotka alebo výsek súvislej reči alebo písaného textu, ktorá je dlhšia ako konvenčná veta.

Entita je položka, s ktorou možno manipulovať ako s celkom; často je členom určitej kategórie alebo druhu.

Implementácia je proces uskutočňovania teoreticky stanovenej myšlienky alebo projektu za účelom jej ďalšieho použitia.

Inverzia je štylisticko-gramatický jav, ktorý spočíva v porušení obvyklého slovosledu.

Klasifikácia alebo triedenie je osobitný prípad uplatnenia logickej operácie delenia rozsahu pojmu.

Klauzula je určitá časť vety.

Koexistencia alebo koexistovanie (lat.) je časopriestorové spolubytie, súčasný výskyt viacerých vecí, javov, vlastností.

Kognícia je poznávanie, poznávacia schopnosť.

Kompresia je stláčanie, zmenšovanie objemu.

Kontrapunkt je súčasné vedenie niekoľkých samostatných melódií (hlasov) na harmonickom základe.

Korpus je ohraničený súbor jazykových výpovedí zaznamenaných písmom alebo na zvukovom nosiči, ktorý sa spracováva na vedecko-výskumné a učebné ciele.

Kvantifikovať je určiť množstvo niečoho, jeho mieru, zavádzať, zaviesť hľadiská kvantitatívnych vzťahov pri skúmaní javov.

Lexikológia je náuka o slove, slovnej zásobe jazyka a lexikálnej rovine (pláne, podsystéme) jazyka.

Lineárna regresia je najjednoduchší algoritmus regresnej analýzy v strojovom učení.

Lingvistika je veda skúmajúca (spravidla prirodzený) jazyk.

Negácia je logické prevrátenie hodnoty alebo výroku.

Nominalizácia je nadobudnutie nominálneho charakteru.

Ontológia je filozofické učenie o bytí, filozofická disciplína, ktorá sa zaoberá bytím.

Parafráza je lingvistická operácia, ktorá spočíva vo vyjadrení toho istého obsahu, ako má východiskový výraz, konštrukcia alebo výpoveď, inými výrazovými prostriedkami.

Platforma je v informatike pracovné prostredie, ako po hadvérovej stránke, tak aj softvérovej.

Pojem je kognitívny významový útvar alebo kognitívne významové dianie, ktoré je výsledkom poznávania vlastností alebo vzťahov entít a je súčasťou súdov.

Predikcia je jednoznačná predpoveď objektívne možného, doposiaľ nepozorovaného stavu vecí.

Predikát je: v tradičnej logike súčasť kategorického súdu, to, čo sa v logickom súde hovorí o predmete, subjekte súdu; vlastnosť predmetu v modernej logike výroková funkcia s individuovanými premennými alebo meno tejto funkcie.

Server je softvér alebo počítač v počítačovej sieti, ktorý v rámci modelu klient-server spracúva požiadavky klientov.

Softvér je programové vybavenie počítača.

Strojové učenie je podoblasť umelej inteligencie, zaoberajúca sa metódami a algoritmami, ktoré umožňujú programu učiť sa a následne adekvátne reagovať na rôzne vstupné hodnoty bez toho, aby bol na ne explicitne naprogramovaný, iba na základe informácií, ktoré sa naučil.

Syntax je spájanie slov do viet, tvorba viet a syntaktických celkov.

Sémantika je náuka o význame jazykových jednotiek.

Taxonómia je porovnávanie rovnakých javov v jednom alebo viacerých jazykoch ako základ vedeckého triedenia jazykových faktov a jazykov.

Terminológia je súbor termínov, odborných názvov z určitého vedného alebo praktického odboru ľudskej činnosti.

Transformácia je premena vstupov na výstup(y).

Téza je stručne vyjadrené tvrdenie, najmä také, ktorého pravdivosť sa obhajuje, alebo také, ktoré sa podrobnejšie vykladá v referáte, stati či publikácii.

Umelá inteligencia je inteligencia demonštrovaná strojmi, na rozdiel od tej, ktorú prejavuje človek.

Vektor je prvok vektorového priestoru (na rozdiel od skaláru je okrem svojej absolútnej hodnoty určený aj smerom a orientáciou).

Výrok je jazykové vyjadrenie nejakého faktu, názoru, myšlienky, výpovede alebo vyjadrenia.

Znalostný graf je vedomostná báza, ktorú používa spoločnosť Google a jej služby na zlepšenie výsledkov vyhľadávania pomocou informácií získaných z rôznych zdrojov.

Špecifikácia je detailný opis; jasné a presné vymedzenie kvality, rozmerov a vlastností surovín, tovaru, výrobkov, materiálov, súčiastok a pod., prípadne vymedzenie pracovných úkonov alebo postupov, výrobných procesov, skúšobných postupov, organizačných podmienok, pojmov, jednotiek, veličín a pod.; podrobný zoznam.

Štruktúra je usporiadanie entít a ich väzieb v nejakej inej entite.

Zoznam príloh

Príloha A Systémová príručka Generátora otázok

Príloha B Používateľská príručka Generátora otázok

Príloha C Systémová príručka Orchestrátora

Príloha D Používateľská príručka Orchestrátora

Príloha E Systémová príručka mobilnej aplikácie

Príloha F Používateľská príručka mobilnej aplikácie

Príloha G CD médium – záverečná práca v elektronickej podobe

Príloha H Životopis autora

A Systémová príručka Generátora otázok

A.1 Základné informácie o systéme

Generátor otázok bol vytváraný vo vývojovom prostredí *PyCharm*¹ od spoločnosti *JetBrains*² v programovacom jazyku *python*³. V implementácii sme použili už existujúce knižnice a balíčky na podporu strojového rozhodovania, NLP a NLG. Aplikácia dokáže sprostredkovať svoje funkcie pomocou štandardného vstupu alebo spustení servrovej inštancie sediacej na určitom definovanom porte.

A.2 Použité knižnice a balíčky

Nasledujúci zoznam softvérových balíčkov stručne opisuje jednotlivé knižnice a balíčky použité pri implementácii navrhnutého generátora otázok. K jednotlivým knižniciam sú nalinkované aj k nim príslušné voľne dostupné repozitáre, pripravené na zhliadnutie. V repozitároch sú konkrétne opísané postupy na použitie a integrovanie požadovaných funkcionalít do projektu, ktorými sme sa riadili s malými obmenami aj my pri implementácii aplikácie.

- **cube**⁴ - spracovanie prirodzeného jazyka - rozdeľovanie viet, tokenizácia, lematizácia, značkovanie častí reči a analýza závislostí
- **joblib**⁵ - sada nástrojov na poskytovanie pipeliningu v *pythone*

¹PyCharm: <https://www.jetbrains.com/pycharm/>

²JetBrains: <https://www.jetbrains.com/>

³python: <https://www.python.org/>

⁴cube: <https://github.com/adobe/NLP-Cube>

⁵joblib: <https://github.com/joblib/joblib>

- **numpy**⁶ - základný balík pre vedecké výpočty v *pythone*
- **pandas**⁷ - flexibilná a výkonná knižnica pre analýzu manipuláciu s dátami pre *python*, poskytuje označené dátové štruktúry podobné objektom *R data.frame*, štatistické funkcie a iné ...
- **sklearn**⁸ - strojové učenie v *pythone*
- **flask**⁹ - mikro rámec *pythonu* na vytváranie webových aplikácií
- **click**¹⁰ - komponovateľná sada nástrojov príkazového riadku
- **neomodel**¹¹ - mapovač grafu objektov - Object Graph Mapper (OGM) pre grafovú databázu *Neo4j*

A.3 Štruktúra projektu

Nasledujúca stromová štruktúra reprezentuje najzákladnejšie rozloženie zdrojových kódov generátora otázok.

```

Fairgrade_QG
├── classify.py
├── dataset_predict.csv
├── README.md
├── dataset.py
├── setup.py
├── tagger.py
├── pipeline.pkl
├── dataset.csv
├── dataset_raw.txt
├── core
│   ├── question.py
│   ├── __init__.py
│   ├── fairgrade.py
│   └── helpers.py
└── database
    └── __init__.py

```

⁶numpy: <https://github.com/numpy/numpy>

⁷pandas: <https://github.com/pandas-dev/pandas>

⁸sklearn: <https://github.com/scikit-learn/scikit-learn>

⁹flask: <https://github.com/pallets/flask>

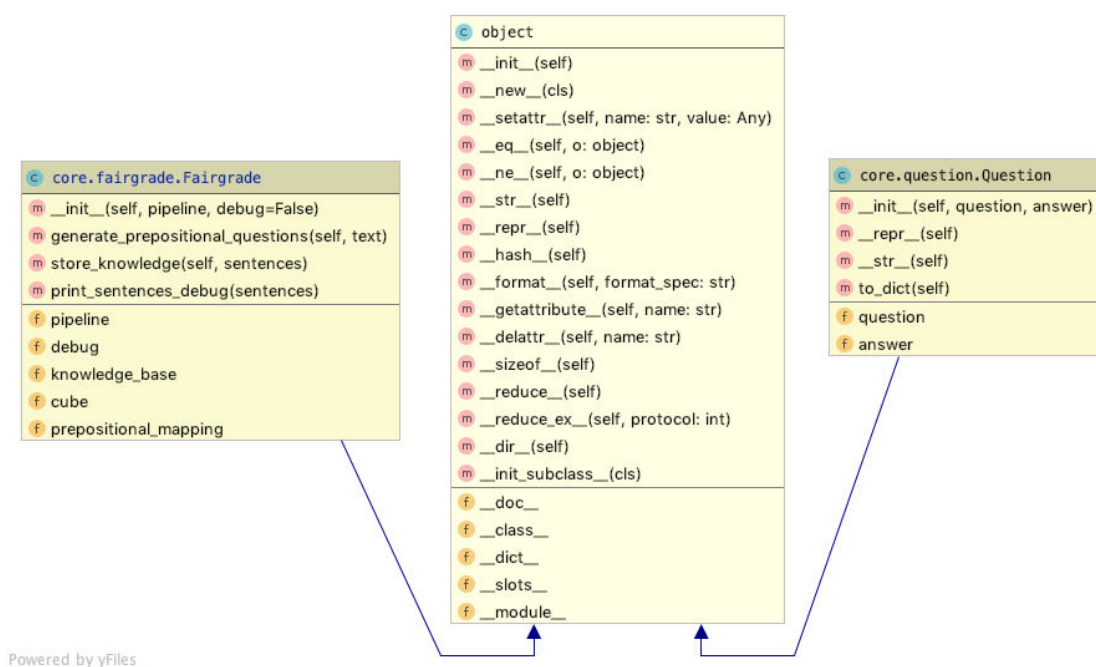
¹⁰click: <https://github.com/pallets/click>

¹¹neomodel: <https://github.com/neo4j-contrib/neomodel>

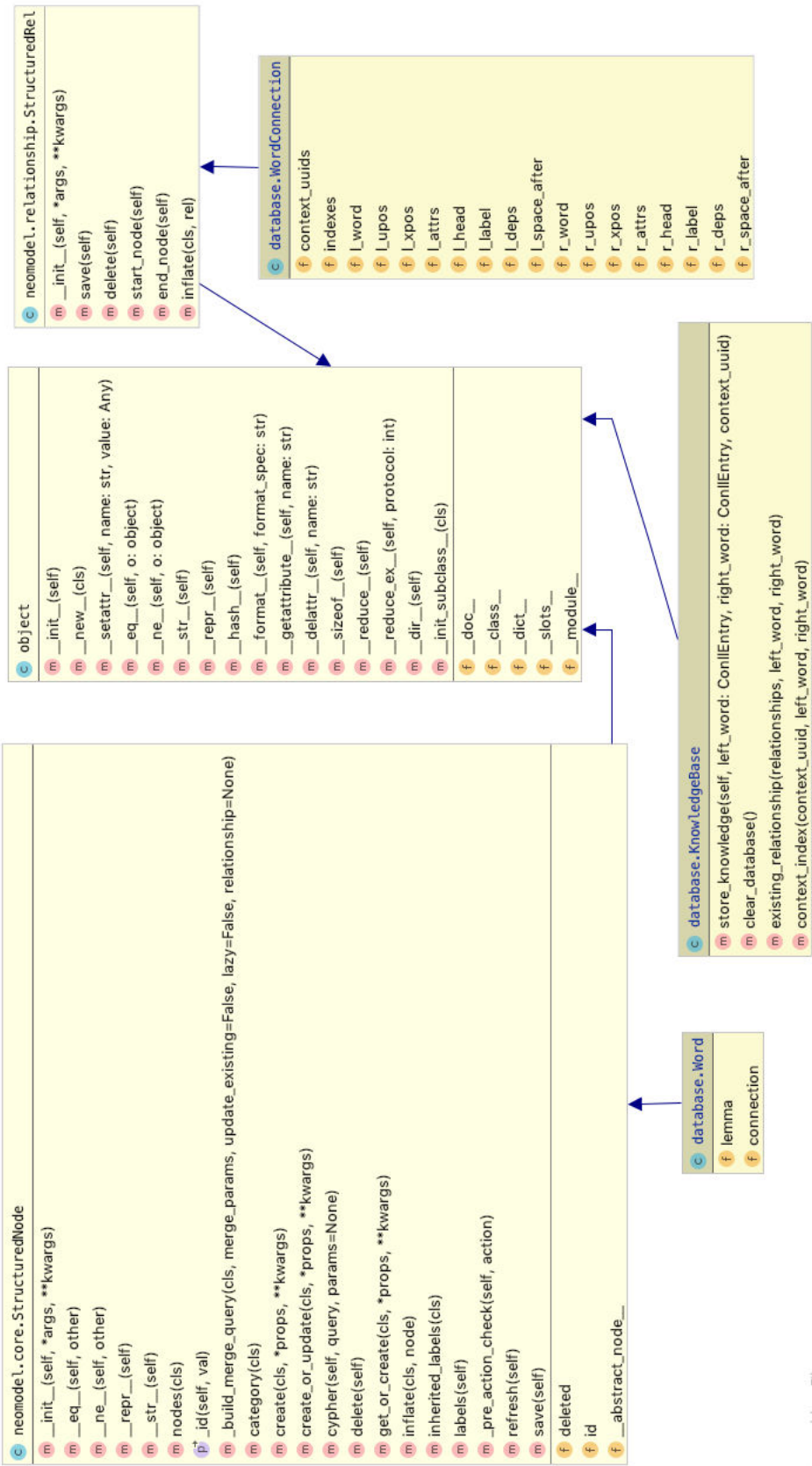
Z hľadiska štruktúry kódu sme sa snažili zachytiť najpodstatnejšie moduly celkovej aplikácie. Hlavná časť aplikácie sa delí na dva moduly a to konkrétne na *core* a *database*. Diagram tried ilustrovaný na obrázku A.1 reprezentuje štruktúru modulu *core* a diagram tried na obrázku reprezentuje štruktúru modulu *database*. V súborovej štruktúre je taktiež možné nájsť aj pomocné nástroje na spracovanie datasetov menovite:

- *classify.py* - nástroj na vygenerovanie modelu *pipeline.pkl* za účelom detekovania vetných vzorov,
- *dataset.py* - nástroj, ktorý sprostredkúva tokenizáciu nad zoznamom viet,
- *tagger.py* - nástroj na spracovanie datasetov s anotáciou konkrétnej triedy.

Detailná dokumentácia aplikácie sa nachádza v prílohe G na priloženom CD vo forme L^AT_EXPDF a interaktívnej HTML verzii.



Obr. A.1: Diagram tried *core* modulu (vygenerované v prostredí *PyCharm*).



Obr. A.2: Diagram tried *database* modulu (vygenerované v prostredí PyCharm).

B Používateľská príručka Generátora otázok

B.1 Inštalácia *Neo4j* databázy

Nasledujúca postupnosť krokov názorne ukazuje spôsob inštalácie *Neo4j* databázy na *Ubuntu* server, nakoľko táto databáza je potrebná pri behu generátora:

1. Inštalácia Java SE Runtime Environment (JRE):

```
$ sudo apt install default-jre default-jre-headless
```

2. Pridanie kľúča k *Neo4j* úložisku do kľúčenky:

```
$ wget --no-check-certificate -O \
- https://debian.neo4j.org/neotechnology.gpg.key | \
sudo apt-key add -
```

3. Pridanie repozitára do zoznamu *apt* zdrojov:

```
$ echo 'deb http://debian.neo4j.org/repo stable/' | \
sudo tee /etc/apt/sources.list.d/neo4j.list
```

V tomto momente je pripravené všetko potrebné na inštaláciu a prvotné spustenie databázy.

4. Nakoniec aktualizácia informácií o pridanom prepozitári a inštalácia *Neo4j*:

```
$ sudo apt update  
$ sudo apt install neo4j
```

5. Server by sa mal spustiť automaticky po inštalácii a mal by byť registrovaný medzi službami systému, tak by sa mal samostatne reštartovať pri štarte fyzických zariadení. V prípade potreby je možné server zastaviť:

```
$ sudo service neo4j stop
```

6. a znovu naštartovať príkazom:

```
$ sudo service neo4j start
```

7. V prípade potreby je možné zhladať aj status služby príkazom:

```
$ sudo service neo4j status
```

Po úspešnej inštalácii je potrebné otvoriť port na danom virtuálnom stroji a pripojiť tak proces vzdialenej komunikácii.

B.2 Inštalácia aplikácie

Na inštaláciu aplikácie do lokálneho alebo virtuálneho prostredia je potrebný len jeden nasledujúci príkaz:

```
$ python setup.py install
```

Tokenizácia tréningového datasetu

Na tokenizáciu vstupného tréningového datasetu je potrebné spustiť nasledujúci príkaz:

```
$ python dataset.py <input_file_path> <output_filename>
```

pričom vstupný dataset by mal zachovávať nasledujúci formát:

```
<sentence> \n
```

a výstupný dataset by mal obsahovať formát:

```
<sentence> \t <(lemma|tag )+> \n
```

Klasifikácia tréningového datasetu

Na klasifikáciu tréningového datasetu je možné použiť nasledujúci príkaz:

```
$ python tagger.py <input_file_path> <output_file_path>
```

pričom vstupný dataset by mal zachovávať nasledujúci formát:

```
<sentence> \t <(lemma|tag )+> \n
```

a výstupný dataset by mal obsahovať formát:

```
<sentence> \t <(lemma|tag )+> \t (P|T|I) \n
```

Trénovanie nového modelu na detekciu vetných vzorov

Na trénovanie modelu je možné použiť nasledujúci príkaz:

```
$ python classify.py <input_csv_path>
```

pričom vstupný dataset vo forme *.csv* by mal zachovávať nasledujúci formát:

```
<sentence> \t <(lemma|tag )+> \t (P|T|I) \n
```

B.3 Spustenie aplikácie

Po spracovaní jednotlivých krokov je následne možné aplikáciu spustiť nasledovným spôsobom priamo zo vstupu z príkazového riadku:

```
$ echo "Pri inštalácii si užívateľ môže vybrať jeden \z  
troch možností inštalovania: úplná, typická a vlastná inštalácia." | \  
fairgrade --pipeline pipeline.pkl
```

alebo je možné aplikáciu spustiť vo forme webovej služby nasledujúcim príkazom:

```
$ fairgrade --pipeline pipeline.pkl --server
```

```
Core init started...  
[dynet] random seed: 2691837243  
[dynet] allocating memory: 512MB  
[dynet] memory allocation done.  
Loading latest local model: sk-1.0  
    Loading embeddings ...  
    Loading tokenization model ...  
    Loading lemmatization model ...  
    Loading tagger model ...  
    Loading parser model ...  
Model loading complete.  
  
Core init done  
* Serving Flask app "fairgrade" (lazy loading)  
* Environment: production  
* Debug mode: off  
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

V prípade potreby je možné zhliadnuť nápovedu k aplikácii:

```
$ fairgrade --help
```

C Systémová príručka Orchestrátora

C.1 Základné informácie o systéme

Orchestrátor bol vytváraný vo vývojovom prostredí *WebStorm*¹ od spoločnosti *JetBrains*² v programovacom jazyku *TypeScript*³. Aplikácia ma formu API s podporou *GraphQL* princípov a sprostredkúva komunikáciu s jednotlivými modulmi v rôznych varziách. Pri výskume slúžila táto aplikácia ako laboratórium.

C.2 Použité knižnice a balíčky

Nasledujúci zoznam softvérových balíčkov stručne opisuje jednotlivé knižnice a balíčky použité pri implementácii navrhnutého *Orchestrátora*. K jednotlivým knižniciam sú taktiež nalinkované aj príslušné voľne dostupné repozitáre. V konkrétnych repozitároch sú opísané postupy na použitie a integrovanie knižníc do projektu.

- **apollo-boost**⁴ (verzia 0.3.1) - bezkonfiguračný spôsob, ako začať používať Apollo klienta,
- **apollo-server**⁵ (verzia 2.4.8) - komunitne udržiavaný open-source GraphQL server,
- **async-parallel**⁶ (verzia 1.2.3) - umožňuje používanie `each()`, `map()`, `filter()` funkcií ako ich štandardné náprotivky, no môžu byť použité s `async/await` a zároveň sprostredkujú paralelizmus,

¹WebStorm: <https://www.jetbrains.com/webstorm/specials/webstorm/webstorm.html>

²JetBrains: <https://www.jetbrains.com/>

³TypeScript: <https://www.typescriptlang.org/>

⁴apollo-boost: <https://www.npmjs.com/package/apollo-boost>

⁵apollo-server: <https://www.npmjs.com/package/apollo-server>

⁶async-parallel: <https://www.npmjs.com/package/async-parallel>

- **class-transformer**⁷ (verzia 0.2.0) - umožňuje transformovať obyčajný objekt na nejakú inštanciu triedy a naopak,
- **class-validator**⁸ (verzia 0.9.1) - umožňuje použitie overovania na základe vzoru dekorátor a bez dekorátora,
- **connect-timeout**⁹ (verzia 1.9.0) - umožňuje správu request time outov v rámci rámca *Express*,
- **cross-fetch**¹⁰ (verzia 3.0.1) - univerzálne WHATWG fetch API pre *Node*, prehliadače a *React Native*,
- **dotenv**¹¹ (verzia 7.0.0) - modul s nulovou závislosťou, ktorý načítava premenné prostredia zo súboru *.env* do *process.env*,
- **graphql**¹² (verzia 14.1.1) - referenčná implementácia GraphQL do *JavaScriptu*, dotazovací jazyk pre API vytvorené Facebook-om,
- **graphql-tag**¹³ (verzia 2.10.1) - užitočné nástroje na analýzu dotazov GraphQL,
- **jest**¹⁴ (verzia 24.1.0) - balíček s funkciami na testovanie *JavaScriptu*,
- **neo4js**¹⁵ (verzia 3.3.2) - *Neo4j-js* je mapovač objektových grafov pre *JavaScript* a *Neo4j* s plnou podporou jazyka *TypeScript*,
- **node-fetch**¹⁶ (verzia 2.3.0) - jednoduchý modul, ktorý prináša *window.fetch* do *Node.js*,
- **nodemon**¹⁷ (verzia 1.18.3) - nástroj, ktorý pomáha vyvíjať aplikácie založené na *Node.js* automatickým reštartovaním aplikačných uzlov, v prípade zistenia zmeny v súborovom adresári,

⁷class-transformer: <https://www.npmjs.com/package/class-transformer>

⁸class-validator: <https://www.npmjs.com/package/class-validator>

⁹connect-timeout: <https://www.npmjs.com/package/connect-timeout>

¹⁰cross-fetch: <https://www.npmjs.com/package/cross-fetch>

¹¹dotenv: <https://www.npmjs.com/package/dotenv>

¹²graphql: <https://www.npmjs.com/package/graphql>

¹³graphql-tag: <https://www.npmjs.com/package/graphql-tag>

¹⁴jest: <https://www.npmjs.com/package/jest>

¹⁵neo4js: <https://www.npmjs.com/package/neo4js>

¹⁶node-fetch: <https://www.npmjs.com/package/node-fetch>

¹⁷nodemon: <https://www.npmjs.com/package/nodemon>

- **prettier**¹⁸ (verzia 1.14.2) - nástroj slúžiaci na formátovanie kódu,
- **supertest**¹⁹ (verzia 3.1.0) - zjednodušené dotazy na textovanie HTTP,
- **ts-jest**²⁰ (verzia 23.1.3) - je *TypeScript* preprocesor s podporou zdrojovej mapy, ktorý umožňuje používať *Jest* na testovanie projektov písaných v jazyku *TypeScript*,
- **ts-loader**²¹ (verzia 4.4.2) - *TypeScript* loader pre *Webpack*,
- **ts-node**²² (verzia 7.0.1) - balíček na spúšťanie *REPL* pre *Node.js* s podporou zdrojovej mapy,
- **tsconfig-paths**²³ (verzia 3.5.0) - načítanie modulov, ktorých umiestnenie je zadané v *tsconfig.json*,
- **tslint**²⁴ (verzia 5.11.0) - rozšírený nástroj na statickú analýzu, ktorý kontrolu typový kód pre zaručenie čitateľnosti a udržiavateľnosti,
- **@nestjs/common**²⁵ (verzia 5.4.0) - progresívny *Node.js* rámec pre budovanie efektívnych a škálovateľných serverových aplikácií,
- **@nestjs/core**²⁶ (verzia 5.4.0) - progresívny *Node.js* rámec pre budovanie efektívnych a škálovateľných serverových aplikácií,
- **@nestjs/graphql**²⁷ (verzia 6.0.2) - *GraphQL* modul pre *Nest.js*,
- **apollo-server-express**²⁸ (verzia 2.4.8) - integrácia *GraphQL Express* servra,
- **graphql-tools**²⁹ (verzia 4.0.4) - balík, ktorý poskytuje niekoľko užitočných spôsobov ako vytvárať *GraphQL* schému,

¹⁸prettier: <https://www.npmjs.com/package/prettier>

¹⁹supertest: <https://www.npmjs.com/package/supertest>

²⁰ts-jest: <https://www.npmjs.com/package/ts-jest>

²¹ts-loader: <https://www.npmjs.com/package/ts-loader>

²²ts-node: <https://www.npmjs.com/package/ts-node>

²³tsconfigpaths: <https://www.npmjs.com/package/tsconfig-paths>

²⁴tslint: <https://www.npmjs.com/package/tslint>

²⁵@nestjs/common: <https://www.npmjs.com/package/@nestjs/common>

²⁶@nestjs/core: <https://www.npmjs.com/package/@nestjs/core>

²⁷@nestjs/graphql: <https://www.npmjs.com/package/@nestjs/graphql>

²⁸apollo-server-express: <https://www.npmjs.com/package/apollo-server-express>

²⁹graphql-tools: <https://www.npmjs.com/package/graphql-tools>

- **neo4j-driver**³⁰ (verzia 1.7.2) - databázový ovládač pre *Neo4j*,
- **neo4j-graphql-js**³¹ (verzia 2.4.2) - vrstva na vykonávanie *GrapQL* dotazov v jazyku *Cypher* s účelom vytvárania dotazov na *Neo4j* databázu,
- **reflect-metadata**³² (verzia 0.1.12) - pridanie metadát do *ECMAScript-u*,
- **rimraf**³³ (verzia 2.6.2) - *UNIX* príkaz *rm -rf* pre *Node.js*,
- **rxjs**³⁴ (verzia 6.2.2) - reaktívne rozšírenia pre *JavaScript*,
- **typescript**³⁵ (verzia 3.0.1) - jazyk v aplikačnom rozsahu *JavaScript-u*,

C.3 Štruktúra projektu

Nasledujúca stromová štruktúra reprezentuje základnú štruktúru projektu a ilustruje rozloženie zdrojových kódov v rámci repozitára.

```
Fairgrade_Orchestrator
├── models
│   └── slovakweb.dagger
├── src
├── test
├── .env
├── .prettierrc
├── configure_dagger.sh
├── nest-cli.json
├── nodemon.json
├── nodemon-debug.json
├── package.json
├── README.md
├── tsconfig.build.json
├── tsconfig.json
└── tslint.json
```

Stromová štruktúra ilustrovaná na ďalšej strane detailnejšie popisuje vnútornú štruktúru projektu a bola vytvorená za účelom sprostredkovania jednoduchého prehľadu v implementovaných zdrojových súboroch.

³⁰neo4j-driver: <https://www.npmjs.com/package/neo4j-driver>

³¹neo4j-graphql-js: <https://www.npmjs.com/package/neo4j-graphql-js>

³²reflect-metadata: <https://www.npmjs.com/package/reflect-metadata>

³³rimraf: <https://www.npmjs.com/package/rimraf>

³⁴rxjs: <https://www.npmjs.com/package/rxjs>

³⁵typescript: <https://www.npmjs.com/package/typescript>

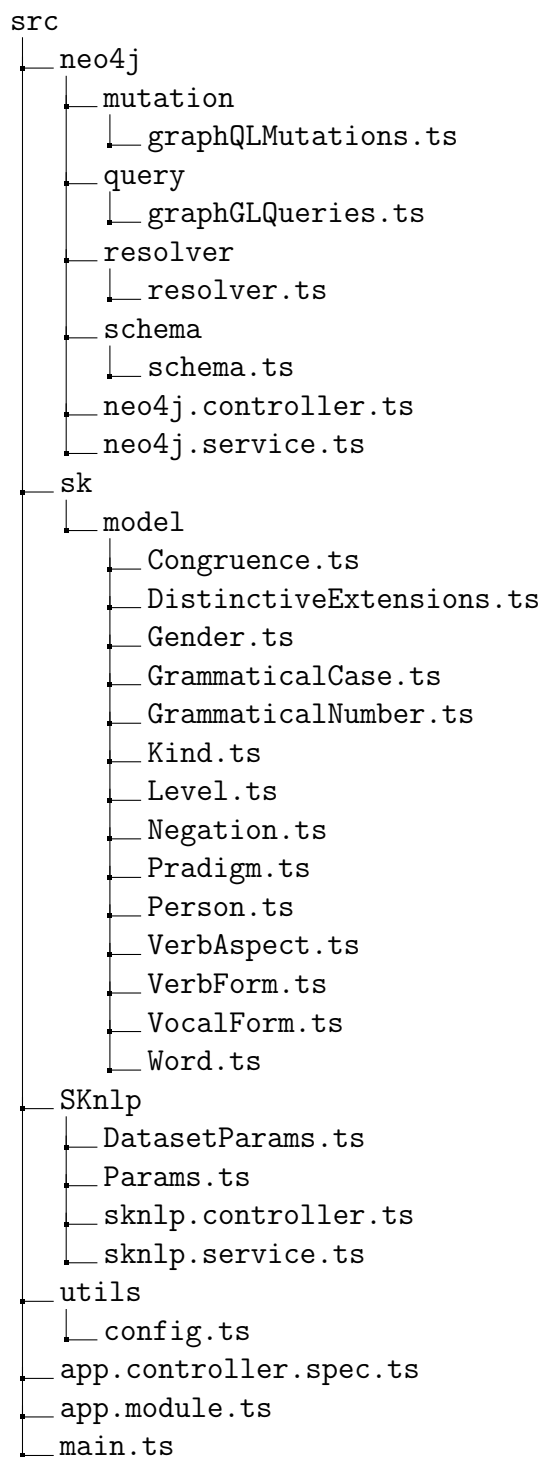
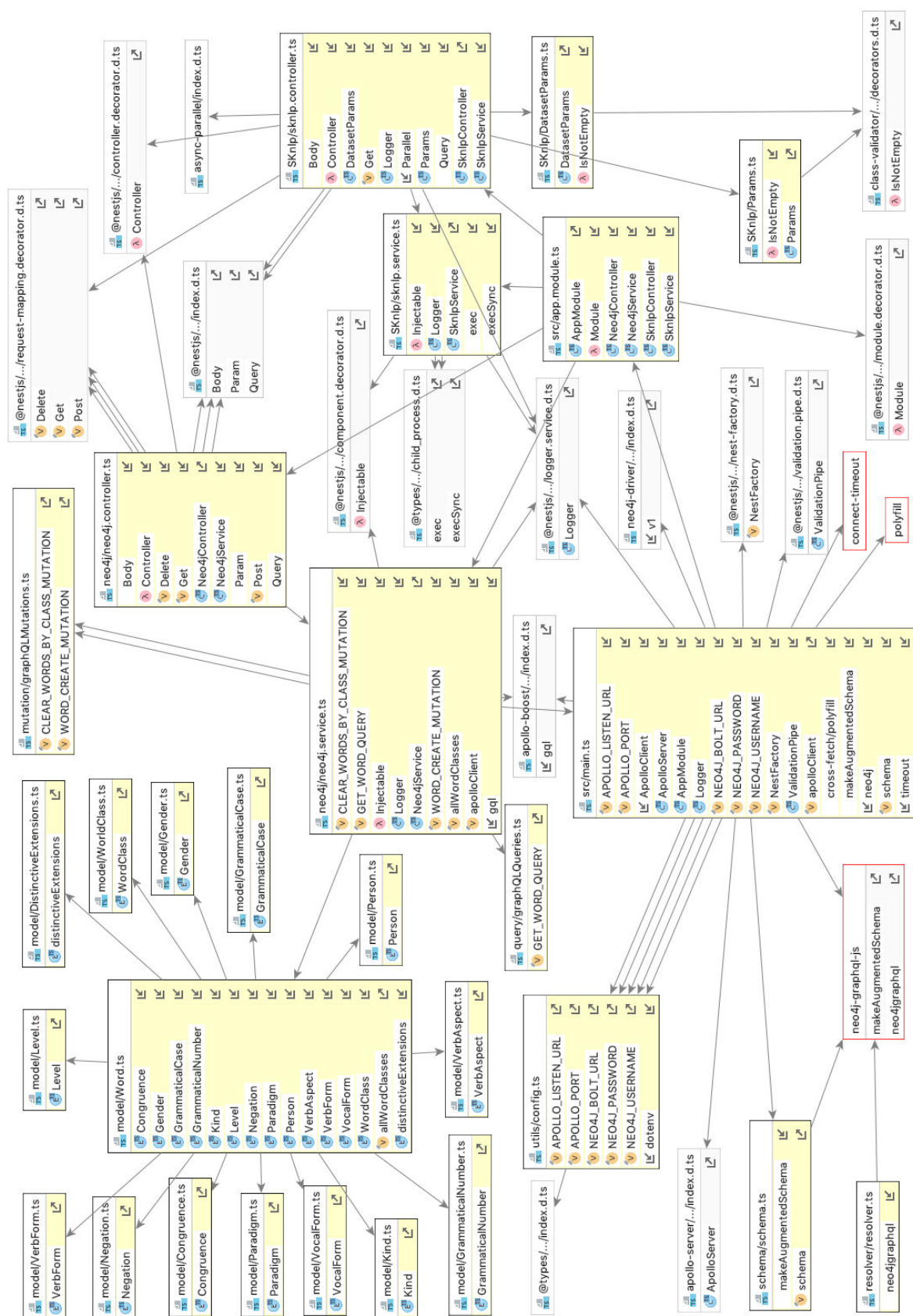


Diagram uvedený na obrázku C.1 znázorňuje diagram tried *Orchestrátora* a spre-
hľadňuje závislosti jednotlivých modulov a tried v projekte.



Obr. C.1: Diagram tried *Orchestrátora* (vygenerované v prostředí *WebStorm*).

D Používateľská príručka Orchestrátora

D.1 Inštalácia závislostí

Inštaláciu závislostí v projekte je možné spustiť príkazom:

```
$ npm install
```

Po inštalácii je možné aplikáciu spustiť v rôznych prostrediach s rôznym zámerom. Jednotlivé definované *npm* skripty popíšeme v nasledujúcej kapitole.

D.2 Spustenie aplikácie

Nasledujúci výpis obsahuje skripty definované na spustenie aplikácie a testov v rôznych prostrediach s rôznymi príznakmi. Aplikáciu je teda možné spustiť vo vývojárskom, produkčnom a debugovacom móde. Taktiež sme integrovali testovacie prostredie jednotkových a end-to-end testov.

- **build** - kompilácia projektu
- **format** - formátovanie kódu
- **start** - spustenie projektu pomocou *ts-node* - bez sledovania zmien v adresároch
- **start:dev** - spustenie projektu vo vývojárskom prostredí
- **start:debug** - spustenie aplikácie v debugovacom móde

- **prestart:prod** - vybudovanie projektu na produkčnú distribúciu
- **start:prod** - spustenie aplikácie v produkčnom prostredí
- **lint** - spustenie statickej kontroly nad zdrojovými kódmi
- **test** - spustenie jednotkových testov
- **test:watch** - spustenie jednotkových testov so sledovaním zmien v adresároch
- **test:cov** - spustenie analýzy pokrytia kódu testami
- **test:debug** - spustenie jednotkových testov v debugovacom móde
- **test:e2e** - spustenie end-to-end testov

```
"scripts": {  
  "build": "tsc -p tsconfig.build.json",  
  "format": "prettier --write \"src/**/*.ts\"",  
  "start": "ts-node -r tsconfig-paths/register src/main.ts",  
  "start:dev": "nodemon",  
  "start:debug": "nodemon --config nodemon-debug.json",  
  "prestart:prod": "rimraf dist && npm run build",  
  "start:prod": "node dist/main.js",  
  "lint": "tslint -p tsconfig.json -c tslint.json",  
  "test": "jest",  
  "test:watch": "jest --watch",  
  "test:cov": "jest --coverage",  
  "test:debug": "node --inspect-brk -r tsconfig-paths/register -r  
    ts-node/register node_modules/.bin/jest --runInBand",  
  "test:e2e": "jest --config ./test/jest-e2e.json"  
}
```

E Systémová príručka mobilnej aplikácie

E.1 Základné informácie o systéme

Mobilná aplikácia bola vyvíjaná vo vývojovom prostredí *Xcode*¹ od spoločnosti *Apple*² v programovacom jazyku *Swift*³. Aplikácia predstavuje jednoduché používateľské rozhranie pre navrhovaný systém a má slúžiť na interakciu s nástrojmi vytvárajúcimi NLP a NLG analýzu.

E.2 Použité knižnice a balíčky

Nasledujúci zoznam softvérových knižníc stručne opisuje jednotlivé závislosti, na ktorých je postavená aplikácia. K jednotlivým knižniciam sú priložené aj príslušné voľne dostupné repozitáre. Vo vnútri týchto repozitárov sa nachádzajú detailné popisy k integrácii týchto nástrojov.

- **SwiftyBeaver**⁴ - farebné, flexibilné a jednoduché logovanie v jazyku *Swift*
- **Firestore/Auth**⁵ - knižnica na komunikáciu s *Firestore* autentifikačným prostredkovateľom
- **Firestore/Core**⁶ - jadro *Firestore* SDK v *Swift*-e

¹Xcode: <https://developer.apple.com/xcode/>

²Apple: <https://www.apple.com>

³Swift: <https://developer.apple.com/swift/>

⁴SwiftyBeaver: <https://cocoapods.org/pods/SwiftyBeaver>

⁵Firestore/Auth: <https://cocoapods.org/pods/FirebaseAuth>

⁶Firestore/Core: <https://cocoapods.org/pods/FirebaseCore>

- **Firebase/Storage**⁷ - knižnica na komunikáciu s *Firebase* cloudovým súborovým úložiskom
- **Firebase/Firestore**⁸ - knižnica na komunikáciu s *Firestore* databázou
- **SwiftLint**⁹ - nástroj slúžiaci na statickú kontrolu *Swift* zdrojových kódov
- **Fabric**¹⁰ - knižnica poskytujúca nástroje potrebné na rýchle vytváranie mobilných aplikácií
- **Crashlytics**¹¹ - nástroj na vytváranie chybových hlásení
- **FBSDKLoginKit**¹² - open-source knižnica na integráciu *Facebook*-u do *iOS* aplikácie
- **GoogleSignIn**¹³ - SDK na prihlasovanie pomocou *Google* účtov
- **TwitterKit**¹⁴ - SDK na podporu komunikácie so sociálnou sieťou *Twitter*
- **OAuthSwift**¹⁵ - *Swift* knižnica podporujúca *OAuth* princípy pre *iOS* a *macOS*
- **lottie-ios**¹⁶ - je mobilná knižnica pre *Android* a *iOS*, ktorá natívne vykresľuje vektorové animácie a umenie v reálnom čase s minimálnou integráciou
- **RxSwift**¹⁷ - *Swift* verzia reaktívnej platformy *ReactiveX*
- **CodableFirebase**¹⁸ - knižnica podporujúca používanie špecifických typov, ktoré podporujú protokol *Codable* voči *Firebase* SDK

⁷Firebase/Storage: <https://cocoapods.org/pods/FirebaseStorage>

⁸Firebase/Firestore: <https://cocoapods.org/pods/FirebaseFirestore>

⁹SwiftLint: <https://cocoapods.org/pods/SwiftLint>

¹⁰Fabric: <https://cocoapods.org/pods/Fabric>

¹¹Crashlytics: <https://cocoapods.org/pods/Crashlytics>

¹²FBSDKLoginKit: <https://cocoapods.org/pods/FBSDKLoginKit>

¹³GoogleSignIn: <https://cocoapods.org/pods/GoogleSignIn>

¹⁴TwitterKit: <https://cocoapods.org/pods/TwitterKit>

¹⁵OAuthSwift: <https://cocoapods.org/pods/OAuthSwift>

¹⁶lottie-ios: <https://cocoapods.org/pods/lottie-ios>

¹⁷RxSwift: <https://cocoapods.org/pods/RxSwift>

¹⁸CodableFirebase: <https://cocoapods.org/pods/CodableFirebase>

E.3 Štruktúra projektu

Dokumentáciu mobilnej aplikácie rovnako doľňame o stromovú štruktúru repozitáru, ktorá má slúžiť k jednoduchšej orientácii medzi zdrojovými súbormi.

```
Fairgrade_iOS
├── Reactive+Extensions
│   ├── ReactiveFirestore.swift
│   └── String+Extensions.swift
├── Protocols
│   └── DatabaseCollenction.swift
├── Model
│   ├── Category.swift
│   ├── Library.swift
│   ├── Test.swift
│   └── User.swift
├── Managers
├── Foundation+Extensions
│   └── DateExtension.swift
├── UIKit+Extensions
│   └── UIImageExtension.swift
├── Lottie
│   ├── done.json
│   ├── upload.json
│   ├── search_knowledge_grey.json
│   ├── search_knowledge.json
│   ├── books_animation.json
│   └── login_loader.json
├── Helper
│   ├── AlertHelper.swift
│   ├── ModalHelper.swift
│   └── FairgradeError.swift
├── Designable
│   ├── CardView.swift
│   └── RoundedButton.swift
├── Fairgrade.entitlements
├── Storyboard
│   ├── Main.storyboard
│   └── LaunchScreen.storyboard
├── Controller
│   ├── Modal
│   │   └── AuthorizationModalViewController.swift
│   ├── Library
│   │   └── LibraryViewController.swift
```

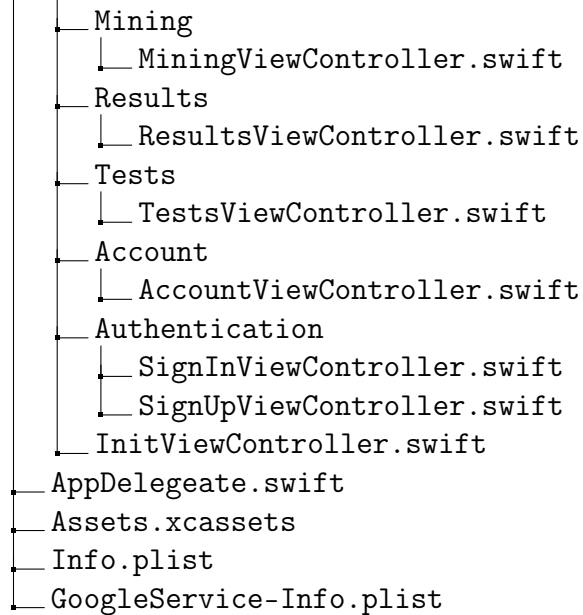
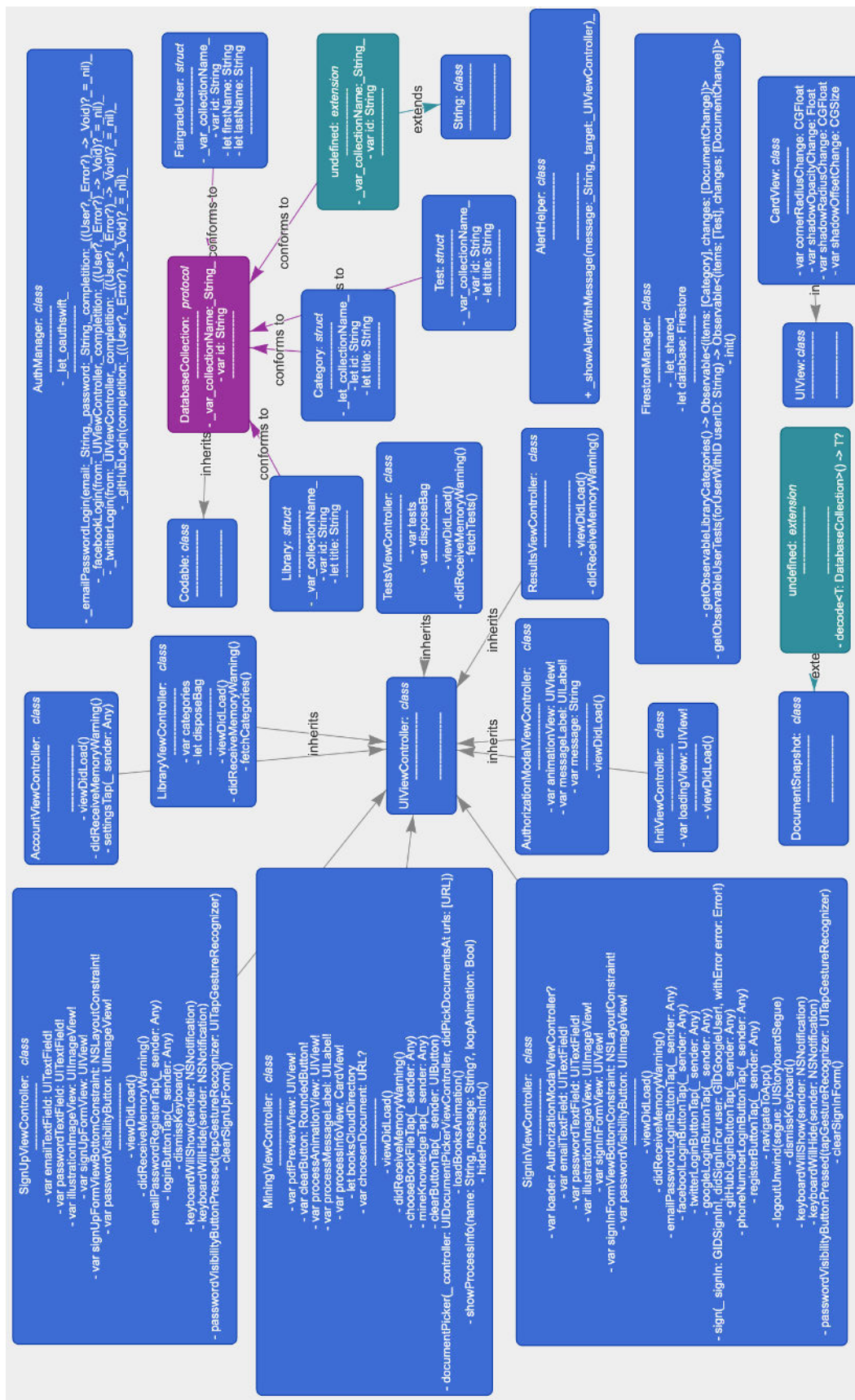


Diagram ilustrovaný na obrázku E.1 je názornou ukážkou diagramu tried implementovanej aplikácie. Je v ňom možné identifikovať jednotlivé UI ovládače no i dátové modely slúžiace ako rozhranie medzi *Firebase* databázou a používateľským rozhraním. Rovnako je možné zhliadnuť vnútornú štruktúru ovládačova a získať tak vedomosti o funkciách a atribútoch existujúcich v konkrétnych triedach riešenia.



Obr. E.1: Diagram tried *Orchestrátora* (vygenerované pomocou nástroja *WebStorm*).

F Používateľská príručka mobilnej aplikácie

F.1 Inštalácia závislostí

Na spustenie projektu je potrebné v prvom kroku nainštalovať manažéra závislostí *CocoaPods*¹. Túto akciu je možné vykonať na strojoch s operačným systémom *macOS* nasledujúcim príkazom:

```
$ sudo gem install cocoapods
```

Následne v priečinku projektu je potrebné inicializovať inštaláciu závislostí príkazom:

```
$ pod install
```

Po úspešnej inštanácii je možné prejsť ku kroku spustenia aplikácie.

F.2 Spustenie aplikácie

Aplikáciu je možné spustiť priamo v prostredí *Xcode* s použitím osobného alebo firemného certifikátu. Aplikáciu je možné spustiť v simulátore alebo priamo na fyzickom zariadení.

¹CocoaPods: <https://cocoapods.org>

G CD médium – záverečná práca v elektronickej podobe

Súborová štruktúra uložená na CD médiu:

```
CD
├── doc ..... záverečná práca v PDF formáte
├── lib ..... dokumentácia k zdrojovým kódom
│   ├── Fairgrade_QG_lib
│   ├── Fairgrade_Orchestrator_lib
│   └── Fairgrade_iOS_lib
├── src ..... zdrojové kódy
│   ├── Fairgrade_QG_src
│   ├── Fairgrade_Orchestrator_src
│   └── Fairgrade_iOS_src
└── tex ..... zdrojové súbory záverečnej práce
```



Bc. Dominik Sečka [Programmer , Graphic designer]



PERSONAL INFORMATION
family status: married



ADDRESS
Komenského 501/12
Markušovce 053 21



CONTACT
secka.dominik@gmail.com
+421 919 394 580



PERSONAL STATEMENT

I am 24 years old mid programmer, currently studying Computer Science at Technical University of Košice. My biggest interest is in development of various kinds of software and applications. My speciality is to discover new ideas and overhead boundaries of my thinking to implement my knowledge and skills to the code. The second place in my list of interests belongs to graphic design and portrait drawing.



EDUCATION

2014 - ...

2010 - 2014

2001 - 2014

Technical University of Košice
Faculty of Electrical Engineering and Informatics
Computer Science
Letná 9, Košice 042 00

Catholic secondary school of Štefan Mišík
Radničné námestie 271/8,
Spišská Nová Ves, 052 01

Art school
J. Fabiniho 1,
Spišská Nová Ves, 052 01



PROGRAMMING SKILLS

Java	●●●●●●●●○○○○
C#	●●●●●○○○○○
C	●●●●●○○○○○
HTML	●●●●●●●○○○
CSS	●●●●●●●○○○
JavaScript	●●●●●○○○○○
meteor.js	●●●●○○○○○○
aurelia.js	●●○○○○○○○○
PHP	●●●●●●○○○○
Android	●●●●●●○○○○
iOS	●●●●●●●○○○
Embedded systems	●●●●○○○○○○
SQL	●●●●●●●○○○
noSQL	●●●●●●●○○○
node.js	●●●●●○○○○○
Angular	●●●●●○○○○○
Ionic	●●●●●○○○○○
Bootstrap	●●●●●○○○○○
Ethereum	●●●●●●●○○○
Haskell	●●●○○○○○○○
Visual Basic	●●●●○○○○○○
ReactiveX	●●●●●○○○○○
React	●●●●●●●○○○
Wicket	●●●●●○○○○○
Laravel	●●●●○○○○○○
Google cloud platform	●●●●●○○○○○



LANGUAGE SKILLS



SOFTWARE SKILLS

Ps	Adobe Photoshop	●●●●●●○○○○
Ai	Adobe Illustrator	●●●●●●○○○○
Id	Adobe InDesign	●●○○○○○○○○
CDR	CorelDRAW	●●●●●●○○○○
DOCX	Microsoft Office	●●●●●●●○○○
Linux	Linux	●●●●●○○○○○
Windows	Windows	●●○○○○○○○○
MacOS	MacOS	●●●●●●●○○○
Unity	Unity	●●●●●○○○○○
Blender	Blender	●●●●●●○○○○
Sketch	Sketch	●●●●●●●○○○



ORGANIZATIONS

IAESTE SLOVAKIA	The International Association for the Exchange of Students for Technical Experience
avse	Association of Ving Tsun, Security System and Escrima
OZ FOYA	Association to promote sports events



DRIVING LICENCE

AM AM A1 A2 B1 B



KNOWLEDGE, SKILLS AND INTERESTS



SPECIFIC CHARACTERISTICS



team player



hard worker



open mind



communicator



ethical



Art



Martial arts



Cycling



Vintage cars



Books