VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

STROJOVÉ UČENÍ PRO ODPOVÍDÁNÍ NA OTÁZKY V PŘIROZENÉM JAZYCE

MACHINE LEARNING FOR NATURAL LANGUAGE QUESTION ANSWERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JONÁŠ SASÍN**
AUTHOR

VEDOUCÍ PRÁCE **Doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Sasín Jonáš**
Program: Informační technologie
Název: **Strojové učení pro odpovídání na otázky v přirozeném jazyce**
Machine Learning for Natural Language Question Answering
Kategorie: Databáze

Zadání:

1. Seznamte se s metodami odpovídání na otázky v přirozeném jazyce se zaměřením na techniky strojového učení pro extrakci odpovědí, případně přenos výsledků napříč jazyky
2. Shromážděte data pro testování průběžného vývoje i pro závěrečné vyhodnocení úspěšnosti zvolených metod v českém jazyce.
3. Navrhněte a realizujte systém, který s využitím existujících implementací dokáže odpovídat na otázky nad českou wikipedií, případně nad doplňkovými zdroji informací.
4. Vyhodnoťte výsledky systému na shromážděných datech a diskutujte možná zlepšení.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle domluvy s vedoucím

Pro udělení zápočtu za první semestr je požadováno:

- Funkční prototyp řešení

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Smrž Pavel, doc. RNDr., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Práce se zabývá odpovídáním na otázky v přirozeném jazyce nad českou Wikipedií. Systémy pro odpovídání na otázky získávají rostoucí popularitu, většina jich ale vzniká pro angličtinu. Cílem této práce je prozkoumat dostupné možnosti a datové sady a vytvořit takový systém pro češtinu. V práci jsem se zaměřil na dva přístupy. Jeden z nich využívá pro extrakci odpovědi anglický model ALBERT a strojový překlad pasáží. Druhý využívá vícejazyčný model BERT. V práci je provedeno porovnání několika variant systému. Diskutovány jsou také možnosti získávání relevantních pasáží. Pro všechny varianty testovaných systémů je provedeno vyhodnocení pomocí standardních metrik. Nejlepší varianta systému byla vyhodnocena na datové sadě SQAD v3.0 s úspěšností 0,44 EM a 0,55 F1 skóre, což je v porovnání s existujícími systémy vynikající výsledek. Hlavním přínosem této práce je analýza možností a nasazení laťky pro další vývoj lepších systémů pro češtinu.

Abstract

This thesis deals with natural language question answering using Czech Wikipedia. Question answering systems are experiencing growing popularity, but most of them are developed for English. The main purpose of this work is to explore possibilities and datasets available and create such system for Czech. In the thesis I focused on two approaches. One of them uses English model ALBERT and machine translation of passages. The other one utilizes the multilingual BERT. Several variants of the system are compared in this work. Possibilities of relevant passage retrieval are also discussed. Standard evaluation is provided for every variant of the tested system. The best system version has been evaluated on the SQAD v3.0 dataset, reaching 0.44 EM and 0.55 F1 score, which is an excellent result compared to other existing systems. The main contribution of this work is the analysis of existing possibilities and setting a benchmark for further development of better systems for Czech.

Klíčová slova

zpracování přirozeného jazyka, NLP, čeština, odpovídání na otázky, strojové učení, dolování znalostí, Wikipedie, otevřená doména, SQAD, ALBERT, BERT, BM25

Keywords

natural language processing, NLP, Czech, question answering, machine learning, knowledge mining, Wikipedia, open-domain, SQAD, ALBERT, BERT, BM25

Citace

SASÍN, Jonáš. *Strojové učení pro odpovídání na otázky v přirozeném jazyce*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. RNDr. Pavel Smrž, Ph.D.

Strojové učení pro odpovídání na otázky v přirozeném jazyce

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jonáš Sasín
6. května 2021

Poděkování

Poděkování patří v první řadě mým rodičům za jejich podporu při studiu. Dále samozřejmě děkuji doc. RNDr. Pavlu Smržovi, Ph.D., za odbornou pomoc a cenné konzultace při vedení práce. Poděkovat bych chtěl také Ing. Karlu Ondřejovi a Ing. Janu Doležalovi za jejich pomoc při nasazení webové aplikace.

Obsah

Kapitola 1

Úvod

Odpovídání na otázky (QA) je dobře prozkoumaný a populární úkol z oblasti zpracování přirozeného jazyka (NLP). Využití kolem nás, v běžném životě, můžeme vidět třeba u vyhledávacích nástrojů, komunikačních agentů nebo hlasových asistentů. Má také potenciál jako budoucnost získávání informací.

Výzkum se v dané oblasti soustředí na metody strojového učení pro extrakci odpovědi na otázku ze získaného kontextu. Kromě porozumění textu jsou v oblasti důležité také metody pro zhodnocení relevance dokumentů, používané všemi vyhledávacími nástroji, a v neposlední řadě také techniky zpracování textu.

Pro češtinu je množství a velikost dostupných datových sad oproti angličtině menší, a je tak obtížné dosáhnout podobných výsledků, jako u jazyků s mnoha zdroji (angličtina, němčina, francouzština, ...). Obzvlášť v kontextu toho, že pro většinu úkolů na poli porozumění textu se v posledních letech nejlépe osvědčily velké modely předtrénované na velmi rozsáhlých korpusech textu. Ty pro češtinu prozatím bohužel nejsou ve stejné kvalitě dostupné. Čeština na tom ale není tak špatně. Má rozsáhlou Wikipedii, existují pro ni tedy vícejazyčné modely. Výzkum v oblasti její morfologie a strojového překladu je na výborné úrovni. Jsou pro ni dostupné i kvalitní nástroje pro zpracování textu.

Cílem práce je vytvořit QA systém, který by byl s pomocí české Wikipedie úspěšný v odpovídání na otázky, ověřit jej na testovací datové sadě SQAD v3.0, porovnat s existujícími řešeními a položit základ pro další podobné systémy vyvíjené nad otevřenou doménou (open-domain) v češtině.

V této práci experimentuji se dvěma přístupy. První využívá pro extrakci odpovědi vícejazyčný model BERT. Druhý pak anglický model ALBERT, pro který jsou získané pasáže textu s otázkou strojově překládány. Kromě metod pro extrakci odpovědi rozebírám v práci možnosti řazení dokumentů dle jejich relevance k otázce.

Nejprve jsou v kapitole 2 popsány metody pro zpracování a reprezentaci textu. Kapitola 3 se zabývá popisem neuronových sítí pro porozumění textu a extrakci odpovědi. Popsány jsou moderní architektury a modely použité v práci. Kapitola 4 shrnuje metody používané pro řazení relevantních dokumentů. V kapitole 5 jsou poté prozkoumány datové sady vhodné pro trénink a vyhodnocení výsledného systému. Následuje kapitola 6, kde je prezentován návrh systému a přehled již existujících. Kapitola 7 je popisem implementace systému, který je postaven na zmíněných postupech, včetně popisu použitých technologií pro jeho realizaci. Zmíněna je také tvorba demonstrační aplikace – r2d2.fit.vutbr.cz/.cs. V kapitole 8 je výsledný systém vyhodnocen s použitím standardních metrik a porovnán se současným stavem, tedy podobnými systémy, které slouží ke stejnému účelu.

Kapitola 2

Reprezentace slov a zpracování textu

Předtím, než se práce dostane k vysvětlení poměrně složitých jazykových modelů a hodnotících funkcí, je v této kapitole vysvětleno pár základních pojmů a technik pro zpracování a reprezentaci textu.

Počítač lidské řeči na elementární úrovni příliš nerozumí. Proto musíme nejdříve porozumět tomu, s jakou reprezentací slov moderní jazykové modely pracují a do jaké podoby je nutno zdrojový text dostat. Vysvětleny jsou tedy termíny související s reprezentací slov pomocí vektorů, neboli „word embeddings“ a tokenizace textu.

Také je důležité popsat způsoby zpracování textů a slov na jejich morfologické úrovni. Tato úloha je na pomezí informatiky a lingvistiky a používají se pro ni speciální nástroje. Použité techniky jako lemmatizace a odstranění stop-slov zajistí optimální funkčnost některých použitých algoritmů.

Po přečtení kapitoly by měl být čtenář teoreticky vybaven pro pochopení konceptů vysvětlených v kapitolách 3 a 4, které na zde popsaných technologiích staví a částečně se s nimi překrývají.

2.1 Reprezentace slov

Jak už bylo řečeno, je složité pracovat s textem v takové podobě, na jakou jsou lidé zvyklí. Počítač v ní nedokáže vidět důležité sémantické souvislosti, které jsou pro přirozený jazyk tak důležité. Musíme tedy z textu a slov v něm získat nějakou matematickou reprezentaci, která je počítači bližší a se kterou dokáží dále nakládat např. modely umělých neuronových sítí.

Pro takovou reprezentaci jsou nejpoužívanější vektory pevně dané dimenze, známé jako „word embeddings“. Vektory slov ve slovníku nesou důležitou informaci o sémantické příbuznosti jednotlivých slov. Po promítnutí do spojitého prostoru jsou si vektory, které reprezentující sémanticky podobná slova, blízko, jak ukazuje obrázek 2.1.

Pro toto téma jsou důležité články [23] a [24], ze kterých také čerpá tato podkapitola. Díky nim byl zaznamenán výrazný pokrok v úlohách z oblasti zpracování přirozeného jazyka, jako je strojový překlad, porozumění textu, analýza postojů, klasifikace atp. Vektory reprezentující slova jsou většinou získávány analýzou rozsáhlých textových korpusů. Snaží se zachytit sémantickou blízkost slov na základě jejich koexistence v podobných kontextech.

Například slova *pohovka* a *gauč* se nejspíše budou vyskytovat v podobných situacích, protože jsou to synonyma a jejich sémantika je téměř totožná.

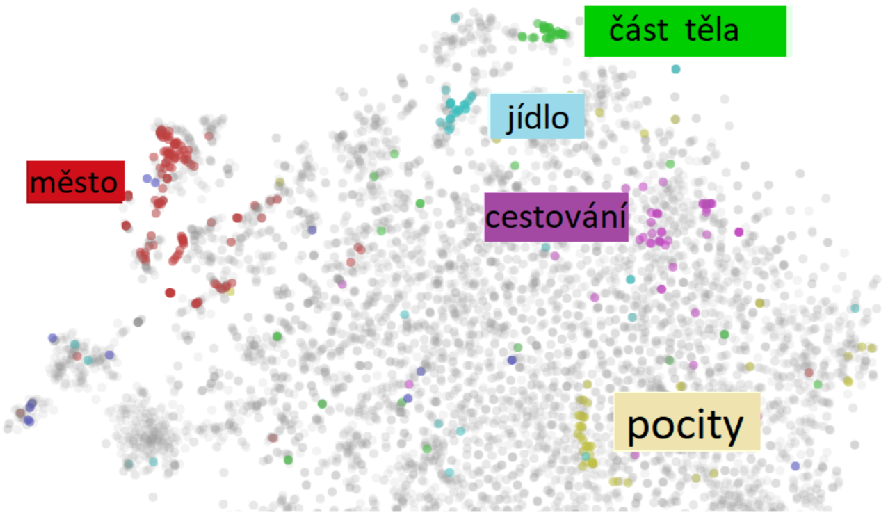
Výsledné vektory nám umožňují provádět se slovy také matematické operace, jako je sčítání a odčítání, a je možné zachytit pomocí nich velmi zajímavé sémantické vztahy, například:

$$[\text{Král}] - [\text{Muž}] + [\text{Žena}] = [\text{Královna}]$$

nebo

$$[\text{Paříž}] - [\text{Francie}] + [\text{Čína}] = [\text{Peking}]$$

V následující části jsou popsány nejznámější metody pro získání vektorových reprezentací a jejich vlastnosti.



Obrázek 2.1: Část vektorového prostoru znázorňující reprezentace slov, která spojuje nějaké téma. Převzato z [31]

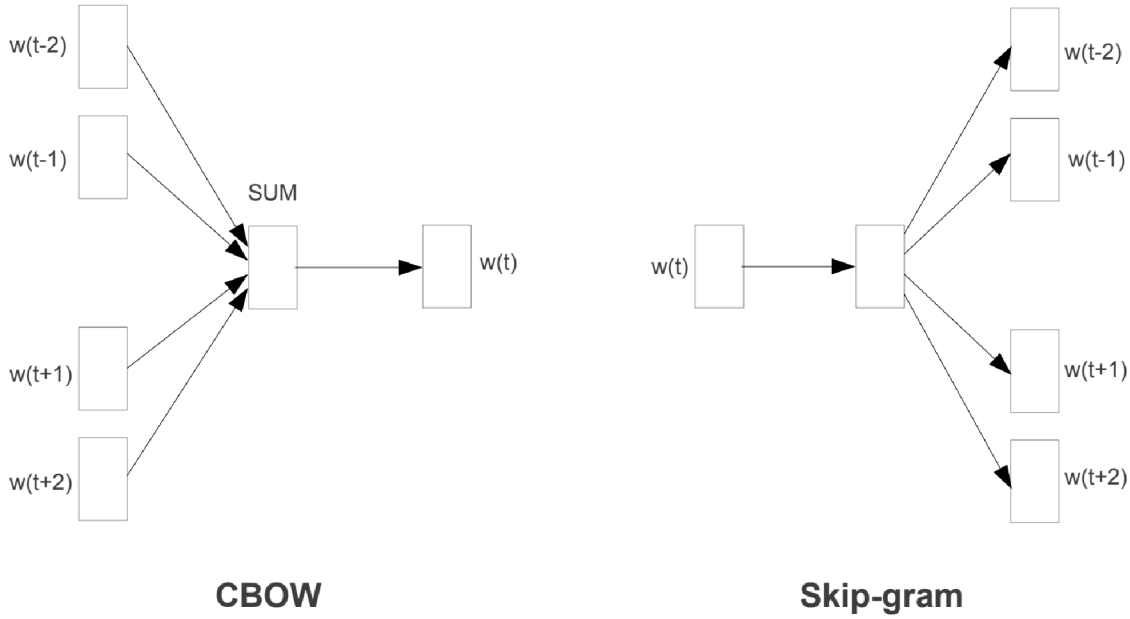
2.1.1 Word2vec

Word2vec je technika prezentovaná v článku [23], schopná naučit se vektorové reprezentace pomocí prediktivního modelu neuronové sítě. Základním konceptem dvou přístupů, které se pro trénink modelu používají, je predikce slov na základě slov sousedních v tzv. kontextovém okně, které se postupně posouvá. Dostáváme tak pro každé slovo pravděpodobnost, se kterou se může vyskytovat v blízkosti slova cílového.

První model používaný pro trénink word2vec vektorů se anglicky označuje jako Continuous Bag of Words, neboli CBOW. Skrytá projekční vrstva neuronové sítě se trénuje hádáním chybějícího slova v kontextu již známých sousedních slov kontextového okna.

Druhá metoda funguje na opačném principu a nazývá se Continuous Skip-gram Model. Na základě současného slova se model snaží předpovědět slova v určité vzdálenosti před i po současném slově, v závislosti na velikosti kontextového okna. Tato metoda se z článku [23] jeví jako o něco úspěšnější.

Na obrázku 2.2 je vidět rozdíl v tom, jak projekční vrstva transformuje vstup na výstup v případě obou architektur.



Obrázek 2.2: Obrázek architektury CBOW, předpovídající slovo na základě kontextu, a Skip-gram, předpovídající kontext na základě slova. Kontextové okno je znázorněno jako $w(t-2)$ až $(t+2)$. Převzato z [23]

2.1.2 GloVe

GloVe, neboli Global Vectors for Word Representation, je technika představená v článku [27]. Výsledkem je vektorový prostor o pevně dané dimenzi, stejně jako u word2vec. Liší se však způsobem získání daných reprezentací.

Zatímco word2vec je prediktivní model, GloVe je spíše modelem statistickým. Klade tedy důraz hlavně na počet výskytů slov a počet jejich výskytů v podobných kontextech. Dostaneme tedy matici společných výskytů slov v korpusu.

Hlavní výhodou tohoto modelu je, že se nesoustředí pouze na slova v těsném kontextu, jak už slovo *Global*, ukryté v GloVe, napovídá. Reprezentace je tedy schopná zachytit sémantické souvislosti v širším kontextu. Nicméně základní idea není konceptu word2vec příliš vzdálená.

2.1.3 Wordpiece embeddings

Wordpiece je prediktivní technika, představená v článku [38], původně určená pro strojový překlad, která je také používána modely popsanými v podkapitole 3.3.

Hlavním záměrem přístupu je lépe se vypořádat s ojedinělými slovy tak, že jsou slova rozdělena do již známých *podslov*. Nabízí tím kompromis mezi *word embeddings* pro celá slova a tzv. *character embeddings* pro jednotlivé znaky.

Slovník je nejprve naplněn všemi znaky v textu a na tomto slovníku je trénován jazykový model. Z výsledného slovníku modelu je poté sestaven slovník nových kombinací jednotlivých znaků tak, aby se zlepšila úspěšnost na trénovací datové sadě. Postup se opakuje, dokud nenarazíme na limit minimální velikosti slovníku, nebo je rozdíl mezi úspěšností jednotlivých iterací modelu příliš malý.

2.2 Tokenizace a předzpracování

Prvotní vyznačení základní struktury a předzpracování textu je důležitou součástí oblasti zpracování přirozeného jazyka. Cílem je transformovat vstupní text do podoby, který je pro automatické zpracování vhodnější.

První část této podkapitoly navazuje na část 2.1 o reprezentaci jednotlivých slov textu. Zabývá se postupem rozdělení textu na tzv. *tokeny*, které mohou být později převedeny na jejich vektorovou reprezentaci. Druhá část je potom věnována postupům pro zpracování a normalizaci textu pro optimální využití hodnotících funkcí. Jak se ukazuje, statistické metody, využívané těmito algoritmy, fungují lépe, pokud jsou z textu odstraněna slova nesoucí pouze malou informaci.

Poznamenejme na tomto místě také, že jednotlivé nástroje pro efektivní zpracování textu, přímo použité v této práci, jsou uvedeny v podkapitole 7.1.

2.2.1 Tokenizace vstupního textu

Tokenizace je převod textu z posloupnosti jednotlivých znaků na posloupnost jednotlivých tokenů. Například rozdělení textu „Adam šel po škole domů.“ na posloupnost tokenů:

$$\_\text{Adam} + \_\text{šel} + \_\text{po} + \_\text{škole} + \_\text{domů} + \_.$$

Primitivním přístupem může být rozdělení textu pomocí bílých znaků a interpunkčních znamének, které však nebere ohled např. na tečky ve zkratkách a špatně se vypořádává se složeninami. Získané tokeny je pak také potřeba normalizovat pro následný převod na word embeddings.

V přístupu uvedeném v podkapitole 2.1.3 jsou slova dělena na tokeny podle toho, která podslova v daném textu jsou slovníku známa. Anglická věta „I like playing.“ by tedy byla převedena na následující tokeny:

$$\_\text{I} + \_\text{like} + \_\text{play} + \_\text{ing} + \_.$$

Na uvedeném případě je stěžejní hlavně převedení posloupnosti znaků „playing“ na dva podřetězce (tokeny) *play* a *ing*. V podkapitole 3.3 budou ještě uvedeny podobné příklady.

2.2.2 Normalizace délky textu a rozdělení do vět

V kontextu práce je normalizací délky textu myšlena nějaká standardní maximální délka dokumentu, ze kterého je později provedena extrakce správné odpovědi. Tento přístup je užitečný z několika důvodů.

Algoritmy pro řazení dokumentů dle relevance sice berou na relativní délku textů ohled, nicméně některé články, např. [14], naznačují, že příliš dlouhé dokumenty mohou být systematicky znevýhodňovány. Citovaný článek sice představuje úpravu klasického algoritmu pro minimalizaci tohoto problému, jsou v něm ale uvedeny i další důvody, proč je vhodné délku jednotlivých odstavců normalizovat.

Model pro vyznačení správné odpovědi v textu je trénován na datové sadě, která příliš dlouhé dokumenty nebere v úvahu[1] a jak je naznačeno v článku [41], není na dlouhých dokumentech příliš úspěšný.

Dalším důvodem je snaha nepřekládat pro účel vyznačení správné odpovědi (experimenty s modelem ALBERT v 8.2.1) příliš dlouhé úseky textu.[2]

[1] Pro každou otázku je jen jeden odstavec kontextu, ne celý článek z Wikipedie.

[2] Hlavně kvůli určitým omezením dostupných nástrojů pro překlad.

Proč rozdělit text do vět? Předpokládejme, že je text již rozdělen do jednotlivých odstavců. Pro každý odstavec je kontrolována jeho délka, jestli nepřesahuje délku maximální. Pokud je maximální délka odstavce překročena, může být sice rozdělen jednoduše na poloviny, na třetiny a tak dále podle potřeby, bude tím ale ztracena velká část kontextu při rozdělení některých vět.

Pro takový případ je vhodné vědět, kde jednotlivé věty začínají a končí, aby bylo možné jednotlivé odstavce případně smysluplně rozdělit. Vhodné je také zařídit, aby se jednotlivé rozdělené „pododstavce" částečně překrývaly a obsahovaly třeba 3 poslední věty odstavce předchozího – pro maximální zachování původního kontextu.

2.2.3 Převod na malá písmena

Převod na malá písmena je sice velmi jednoduchý, ale přesto užitečný krok. Při psaní otázky např. na virtuální klávesnici mobilního telefonu je velmi pravděpodobně, že uživatel napíše „cd-rom" místo „CD-ROM", ale je nutné, aby byly nalezeny dokumenty obsahující obě varianty. Stejně tak může dojít k nestandardnímu užití malých/velkých písmen v některém z dokumentů.

Pro některé případy může převod na malá písmeny znamenat ztrátu části kontextu. Například *Malá Strana* X *malá strana*, kde velké písmeno indikuje vlastní jméno (městskou část). Vhodné je to tedy pouze v některých případech.

2.2.4 Odstranění stop slov

Ve zpracování přirozeného jazyka jsou *stop slova* typicky souborem nejčastěji používaných slov v daném jazyce [19]. Většinou jsou odstraňována, protože kvůli jejich vysokému výskytu nesou menší informační hodnotu a jejich odstranění vede k lepší funkci vyhledávacích algoritmů a hodnotících funkcí [35].

Nejužívanější slova daného jazyka (například češtiny) je možno najít ve frekvenčních seznamech. Získaný seznam je vhodné kriticky zhodnotit a vyčlenit z něj některá slova, která by pro vyhledávání mohla mít kritický význam.[3] Většinou je zanedbatelná většina funkčních slov – spojek, předložek či zájmen. Typickými příklady stop slov pro češtinu jsou: být, a, se, v, že ...

2.2.5 Získání lemmat

Lemmatizace je druh zpracování textu, při kterém je pro každé slovo nalezen základní tvar, tzv. *lemma*.[4]

ulicí běžely děti ⟶ ulice běžet dítě
ostrovy Středozemního moře ⟶ ostrov Středozemní moře

Pro získání lemmat se používají speciální nástroje pro zpracování přirozeného jazyka. *Lemmatizátory* mohou poskytovat také některé doplňkové informace o mluvnických kategoriích jako rod, pád nebo slovní druh. Hlavním úskalím lemmatizátorů je mnohoznačnost jazyků jako je čeština, kdy může být více základních tvarů daného slova dle kontextu [19]. Potom přichází na řadu zjednoznačňování (desambiguace) lemmat na základě kontextu.

[3] Příklad: „první", „země", „člověk", ...

[4] Podobným postupem je *stematizace*, která ale algoritmicky pouze odstraní koncovky a předpony pro nalezení slovního kmene [19].

Využití lemmatizace je různé, ale v případě této práce je podobné, jako účel postupu uvedeného v podkapitole 2.2.3 (převod na malá písmena). Lemmatizace je také užitečným nástrojem při odstranění stop slov. Mohou být odstraněny například všechny tvary slovesa „být“, zatímco seznam stop slov může obsahovat pouze jeho základní tvar.

Kapitola 3

Strojové učení pro extrakci odpovědi

Rozmach strojového učení umožnil obrovský pokrok v oblastech NLP, jako je strojový překlad, rozpoznání entit, generování textu nebo odpovídání na otázky. Proto na tomto přístupu staví všechny nejmodernější metody a tímto směrem se také ubírá aktuální výzkum.

Účelem této kapitoly je vysvětlit čtenáři, jak fungují neuronové sítě a jakým způsobem dokáže počítač porozumět textu a naučit se vyznačit v textu odpověď na danou otázku.

Nejprve jsou ve zkratce vysvětleny principy fungování a učení neuronových sítí. Poté jsou rozebrány nejznámější architektury neuronových sítí, používané pro vypořádání se s úkoly na poli zpracování přirozeného jazyka, a popsány jsou také nejmodernější modely, které na nich staví a jsou vhodné pro extrakci odpovědi z textu.

3.1 Neuronové sítě

V této podkapitole je vysvětlen princip fungování neuronových sítí. Informace byly převzaty z článku [39] a z prezentací stanfordského kurzu NLP [17].

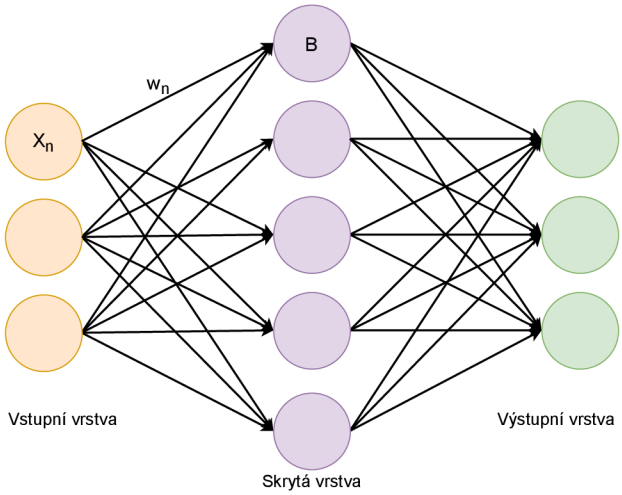
Neuronovou síť si můžeme zjednodušeně představit jako velkou funkci s obrovským množstvím parametrů, která přetváří vstupy na výstupy. V průběhu tréninku neuronové sítě se postupně všechny parametry optimalizují tak, aby síť přetvářela vstupy na výstupy co nejlépe.

Základem jsou tři jednoduché vrstvy (obrázek 3.1). Vstupní, skrytá (zde probíhá proces přetváření vstupů) a výstupní, kde se přetvořený vstup objeví. Skrytá vrstva se navíc běžně skládá z mnoha dalších vrstev.

Každá z těchto vrstev, ať už je vstupní, výstupní nebo skrytá, se skládá z neuronů. Každý neuron v určité vrstvě je propojen s každým neuronem vrstvy další, a tak je propojena celá síť.

3.1.1 Umělý neuron a aktivační funkce

Umělý neuron je primitivní jednotka, inspirovaná neuronem skutečným. Ten má tělo, do kterého přichází krátkými výběžky, tzv. *dendrity*, elektrické signály z ostatním neuronů. Z těla neuronu pak vychází jediný dlouhý výběžek, tzv. *axon*. Neuron se tedy na základě signálů, které do něj přicházejí dendrity, rozhodne (na základě nějaké minimální hranice přijatého signálu), zda bude, nebo nebude vysílat signál dál.



Obrázek 3.1: Schéma neuronové sítě s jedinou skrytou vrstvou. Znázorněn je také jeden vstup neuronu B z předchozí vrstvy x_n a váha jeho propojení w_n

Funkce umělého neuronu je analogická. Jak už bylo popsáno v úvodu podkapitoly 3.1, každý neuron je propojen s každým neuronem vrstvy předcházející i vrstvy další. Na rozdíl od skutečného neuronu má tedy více „axonů“, kterými je propojen s další vrstvou.

Každé propojení mezi dvěma neurony má svoji váhu w, která ovlivňuje důležitost daného vstupu do neuronu. Každý neuron má také svůj práh B a aktivační funkci. Výstupem y neuronu i je tedy vážená suma vstupů do neuronu s přičteným prahem a aplikovanou aktivační funkcí, neboli

$$y_i = F(\sum_{n=1}^{k}(x_n w_n) + B_i) \tag{3.1}$$

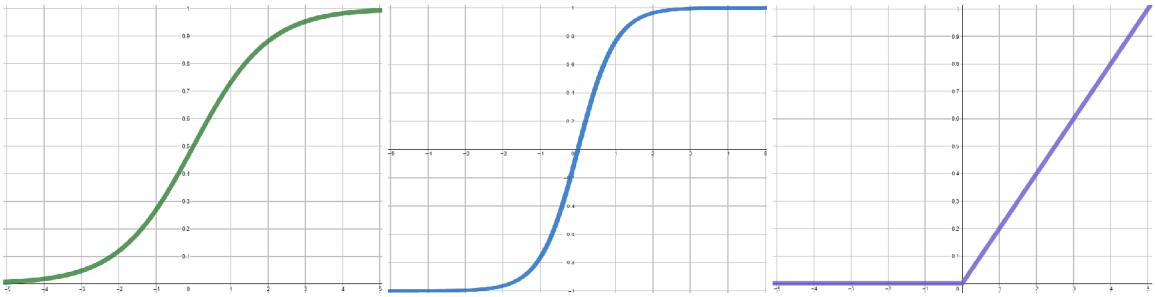
kde x_n je neuron předcházející vrstvy, w_n je váha jeho propojení, B_i je práh aktuálního neuronu i a F je aktivační funkce. Znázornění vah propojení a jednotlivých neuronů je vyznačeno na obrázku 3.1.

Všechna propojení jednotlivých neuronů s jejich prahy dávají dohromady parametry neuronové sítě, které jsou v průběhu tréninku optimalizovány pro každý neuron zvlášť. Pro síť, jako je na obrázku 3.1, je to třeba 30 jednotlivých vah propojení a 8 prahů jednotlivých neuronů (skryté a výstupní vrstvy), tedy 38 parametrů v relativně jednoduché síti.

Aktivační (přenosová) funkce rozhoduje, jestli bude daný neuron aktivován. Skutečný neuron má, zdá se, funkci skokovou. Umělé neurony však používají funkce s postupnou aktivací. Běžné jsou tři základní varianty aktivačních funkcí:

- Sigmoidální
- Hyporbolické tangenty
- ReLU (Rectified Linear Unit)

Funkce jsou znázorněny na obrázku 3.2. Vhodnost jejich použití záleží na kontextu. Výhodou sigmoidální funkce je normalizace výstupu do rozmezí $(0, 1)$. Nejpoužívanější funkcí pro skrytou vrstvou je však funkce ReLU. Kvůli její charakteristice je aktivováno méně neuronů, což vede k rychlejšímu tréninku a konvergenci. Použití ReLU funkce je také výpočetně méně náročné [33].



Obrázek 3.2: Jednotlivé aktivační funkce. Zleva: sigmoidální ($H(f) = \langle 0, 1 \rangle$), hyperbolické tangenty (tanh $H(f) = \langle -1, 1 \rangle$) a ReLU ($H(f) = \langle 0, \text{inf} \rangle$)

3.1.2 Dopředná a zpětná propagace

Proces, kdy se opakovaně ze vstupů počítá výstup každého neuronu v dané vrstvě, a stejně tak pro každou následující vrstvu, se nazývá dopředná propagace. Hodnoty vstupní vrstvy se postupně přetvoří na hodnoty ve vrstvě výstupní a může se určit, jak byla síť v dané iteraci úspěšná.

Pro určení úspěšnosti každé takové iterace neuronové sítě se používá nákladová funkce (anglicky *cost function* pro jednu iteraci, *loss function* pro průměr všech), která udává jediné číslo, charakterizující úspěšnost modelu. Příkladem nákladové funkce je střední kvadratická chyba (anglicky *Mean Squared Error* – MSE). Cílem je hodnotu této nákladové funkce postupně minimalizovat.

Postup optimalizace jednotlivých parametrů neuronové sítě pro zajištění minimalizace nákladové funkce se nazývá **zpětná propagace**. Je to v podstatě opačný průchod neuronovou sítí, který má za úkol zjistit, které neurony se největší mírou podílely na výsledné chybě. Snížením významu těchto neuronů můžeme dosáhnout lepší úspěšnosti celé sítě. Naopak můžeme zvýšit význam propojení s neurony, které výstup ovlivnily pozitivně.

Základem tohoto výpočtu je, že chyba každého výstupního neuronu je vstupem parciální derivace vzhledem ke každému z jeho vstupů. Zpětná propagace je opakována v každé iteraci a parametry neuronové sítě jsou postupně optimalizovány [39].

Celý postup zpětné propagace je složitější a pro intuitivní pochopení problematiky není klíčový, proto tu není podrobně rozebrán.

Trénink neuronové sítě tedy probíhá následovně. Síti s náhodně inicializovanými parametry je předložena část dat z trénovací datové sady. Ty jsou pomocí dopředné propagace prostupem přes skryté vrstvy přetvořeny na výstupy, čímž jsou získány predikce neuronové sítě. Predikce jsou porovnány se správnými očekávanými hodnotami (tzv. *ground truth*) a nákladovou funkcí je vyjádřeno, jak byly predikce daleko od očekávaných. Pomocí zpětné propagace jsou postupně upraveny všechny parametry neuronové sítě.

Postup je opakován, dokud úbytek nákladové funkce mezi jednotlivými iteracemi nezačne dlouhodobě stagnovat, nebo dokud nedosáhneme požadovaného počtu epoch.

Při nesprávném tréninku neuronové sítě může dojít k podtrénování (*underfitting*), nebo přetrénování (*overfitting*). Problémy při tréninku souvisí s tzv. hyperparametry, mezi které patří počet skrytých vrstev, velikost modelu nebo parametr *learning rate*[1]. Pomoci může také množství trénovacích dat nebo počet trénovacích epoch.

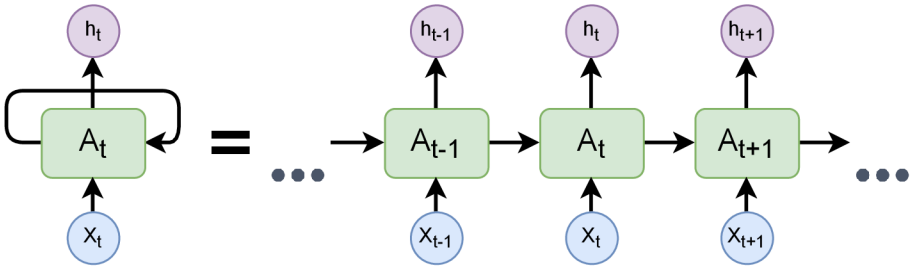
3.2 Nejlepší architektury pro porozumění textu

V této podkapitole jsou popsány dvě nejznámější a nejpoužívanější architektury pro zpracování přirozeného jazyka. Rozšiřují koncepty prezentované v podkapitole 3.1 a snaží se překonat problémy, které tradiční model neuronové sítě v souvislosti s porozuměním textu nese.

3.2.1 Long short-term memory

Informace v této podkapitole byly čerpány z prvotní práce [10], která architekturu představuje, a článku [26], poskytujícího vysvětlení některých konceptů.

LSTM, neboli Long Short-term Memory, je speciální druh *rekurentní* neuronové sítě, který se částečně vypořádává s problémem ztráty kontextu v delším úseku textu. Jednoduše řečeno, při učení neuronové sítě by bylo užitečné, aby pochopení současné věty dokázala ovlivnit i věta předcházející.



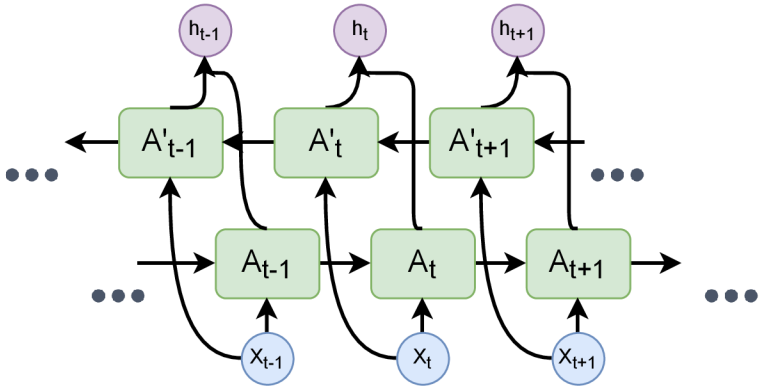
Obrázek 3.3: Schéma rekurentní neuronové sítě – inspirováno obrázkem z [26]. X_t je vstupem neuronové sítě A_t v čase t s výstupem h_t

Nejprve vysvětleme princip **rekurentní neuronové sítě** (dále RNN – *Recurrent Neural Network*). Hlavním cílem této architektury je vypořádat se se zpracováním sekvenčních dat, jako je text. Oproti například rozpoznávání vzorů v obrázcích není možné pohlížet na každé slovo v textu zvlášť. Slovo/slova předcházející by měla ovlivnit slovo současné. Stejně tak mohou následující slova ovlivnit význam slova současného (obousměrné rekurentní sítě).

Schéma jednoduché RNN je vidět na obrázku 3.3. Zjednodušeně je to tedy více jednoduchých neuronových sítí propojených za sebe. Vstupem každé sítě A_t v čase t je výstup sítě A_{t-1}. Výstup h_t je pak vstupem sítě A_{t+1}.

Analogicky funguje obousměrná RNN, která je propojena i v opačném směru. Vstup je zpracováván od konce a pro současné slovo je tedy brán v úvahu i kontext slova následujícího, jak je vidět na obrázku 3.4.

[1]Parametr *learning rate* určuje velikost kroku při každé iteraci zpětné propagace, tedy úpravě parametrů neuronové sítě.

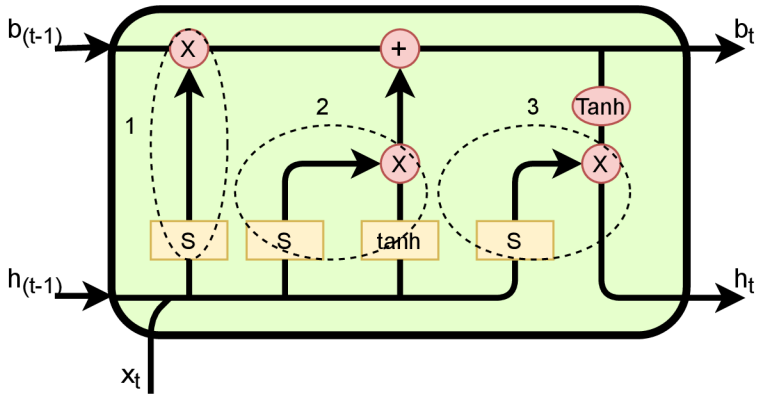


Obrázek 3.4: Schéma obousměrné rekurentní neuronové sítě

LSTM (Long Short-term Memory) představuje vylepšení standardní RNN. Mějme model, který má za úkol předpovědět následující slovo ve větě. Pokud se kontext potřebný pro predikci nachází v těsné blízkosti předpovídaného slova (př.: „vychlazené mléko z *lednice*“), je i běžná RNN schopná souvislost zachytit. Pro příklad, kdy je kontext „dále v minulosti“ (př.:„Učím se od 6 let anglicky ... Vzhledem k těmto okolnostem plynule ovládám *angličtinu*“), nemusí být RNN schopná naučit se tuto souvislost mezi kontextem a hledaným slovem.

LSTM je na rozdíl od klasické RNN navržená tak, aby byla schopná pamatovat si dlouhodobé souvislosti v textu při sekvenčním zpracování. Řeší tím tzv. *problém mizejícího gradientu*.

Model má stejnou opakující se strukturu jako RNN, rozdíl je však v komplexnosti opakující se jednotky (či buňky). Struktura naznačená na obrázku 3.5 je poměrně složitá a její hlavní komponentou jsou tzv. *brány*.



Obrázek 3.5: Schéma LSTM buňky, převzato z článku [26]. b_t je vnitřní stav buňky, $b_{(t-1)}$ předcházející buňky, h_t výstup buňky, $h_{(t-1)}$ výstup předcházející. x_t je vstup současné buňky. „1“ je *zapomínací*, „2“ *vstupní* a „3“ *výstupní* brána

Brány jsou v LSTM buňce celkem tři. Každá je složená z vrstvy neuronové sítě se sigmoidální funkcí („S“ v 3.5) a násobícího („×“ v 3.5) operátoru. Brána je komponentou určující, kolik informace je možno propustit. Brány jsou popsány zleva doprava.

První, tzv. zapomínací brána určuje, kolik vstupní informace je zahozeno[2]. Druhá, tzv. vstupní brána, určí, které hodnoty skrytého stavu buňky budou aktualizovány a vrstva s funkcí *tanh* („tanh“ v 3.5) rozhodne o nových kandidátních hodnotách, které po kombinaci s výstupem sigmoidální vrstvy vytvoří nový skrytý stav buňky. Poslední brána je výstupní, spojuje informaci z brány zapomínací s vnitřním stavem buňky, a vytváří tím tedy výstup dané buňky.

Jednotlivé LSTM buňky jsou za sebou sekvenčně propojeny stejně jako RNN. Při použití dvou LSTM sítí, kdy druhá je propojena v opačném směru, získáme obousměrnou LSTM síť. Výstupy obou sítí jsou poté pro získání výstupu zřetězeny.

3.2.2 Transformer

Informace v této podkapitole byly převzaty z prvotní práce *Attention Is All You Need* [36], dále z [3] a článku [1]. Text popisuje architekturu Transformer, používanou pro zpracování přirozeného jazyka. Také se věnuje mechanismu pozornosti (*attention*).

Přístup použitý tímto modelem se dokázal oprostit od rekurentních modelů jako LSTM a velmi rychle se etabloval jako nejlepší současná architektura na poli zpracování přirozeného jazyka. Sítě typu Transformer nepoužívají sekvenční zpracování slov, ale zpracování paralelní. Nejdůležitější myšlenkou je mechanismus pozornosti, představený článkem [3], původně pro strojový překlad. Ten se dále rozvíjí a jeho varianty pronikají i do dalších odvětví aplikací strojového učení. Stručně je popsán v následujících odstavcích.

Mechanismus pozornosti vezme vektory dvou vět přirozeného jazyka naráz a sestaví z nich matice *key(**K**)-value(**V**)* a *query(**Q**)*, obsahující v řádcích vektory pro každé vstupní slovo. Ty vzniknou vynásobením vstupní matice (vektorových reprezentací) speciální váhovou maticí, jejíž parametry jsou trénovány. Na matice jsou poté aplikovány následující operace pro získání výstupu pozornosti (rovnice 3.2).

$$softmax(\frac{Q \times K}{k}) \times V \tag{3.2}$$

V rovnici 3.2 je k podle [36] konstanta 8 (mohly by být stanoveny i jiné hodnoty) a softmax funkce je použita pro normalizaci výstupu do rozmezí (0,1).

Vektory pro všechna vstupní slova, respektive vstupní matice K, V a Q, mohou být ze stejné věty. Pro tuto variantu mechanismu se používá termín **self-attention**.

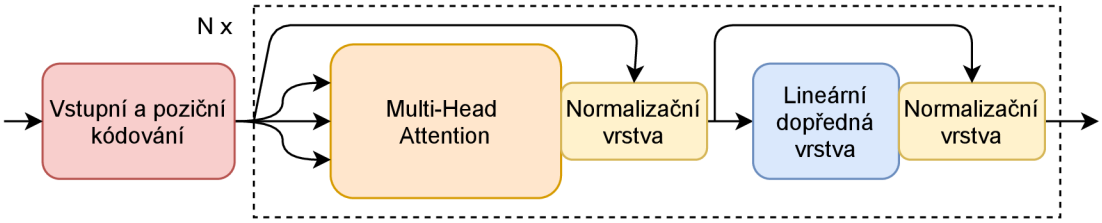
Jednoznačnou výhodou mechanismu pozornosti je, že je schopen podívat se na celý kontext současně (na rozdíl od RNN, která prochází vstup sekvenčně) a vyznačit v něm důležité pasáže. Tímto se zbavuje problému mizejícího gradientu.

Enkodér[3] **Transformer** je model, jehož architektura je kombinací dopředné neuronové sítě a mechanismu pozornosti. Jak už bylo popsáno, jeho hlavní výhodou je vlastnost zpracovávat vstupní slova paralelně, a tím se naučit i vztahy mezi slovy na dlouhé vzdálenosti v textu.

Na obrázku 3.6 můžeme vidět zjednodušenou strukturu jednotky enkodéru. V článku [36] tvoří enkodér 6 takových bloků propojených za sebe.

[2] Výstupem je číslo v rozmezí (0,1), kde 0 znamená „všechno zahodit“ a 1 znamená „všechno propustit“.

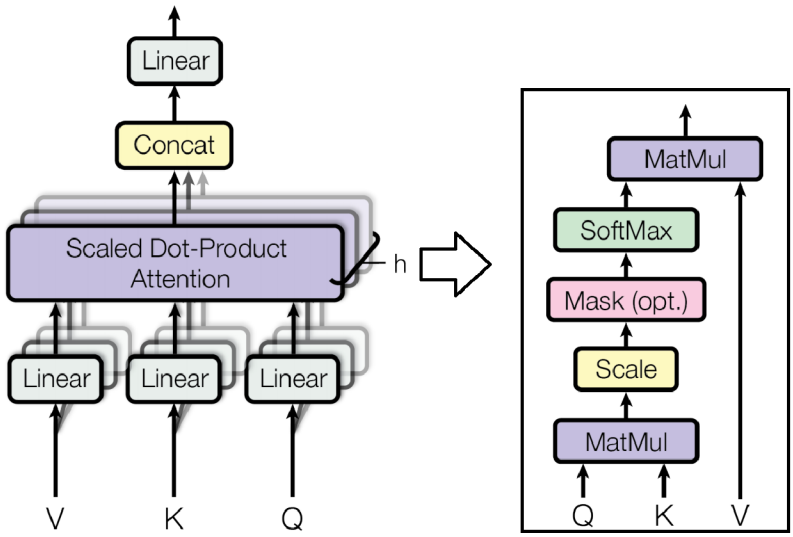
[3] Enkodér je část modelu, která vytvoří z vektorů jednotlivých slov na vstupu vektor celého vstupního textu, reprezentující jeho význam. V původním článku [36] je představen model pro *sequence to sequence* model vhodný pro překlad, který obsahuje i dekodér pro generování textu.



Obrázek 3.6: Schéma architektury enkodéru Transformer převzaté z článku [36]. V článku je uvedeno $N = 6$ pro počet identických vrstev enkodéru

Před vstupem do prvního bloku jsou jednotlivé tokeny vstupního textu převedeny na vektory (word embeddings), ke kterým je přidáno poziční kódování[4]. Každý blok se pak skládá z „*multi-head attention*“ modulu a plně propojené dopředné vrstvy. Obě části jsou následovány normalizační vrstvou.

Nejzajímavější částí bloku enkodéru je multi-head attention modul, který je dále rozkreslen na obrázku 3.7. Modul je sofistikovanou implementací self-attention mechanismu, popsaného dříve, který dokáže vyzdvihnout souvislost jednotlivých částí celého kontextu.



Obrázek 3.7: Schéma *Multi-Head* attention vlevo a *Scaled Dot-Product* attention vpravo. *Multi-head* attention je aplikováno paralelně v h vrstvách (pro práci [36] $h = 8$). Obrázky byly převzaty z článku [36]

Mechanismus na obrázku 3.7 vpravo je popsán rovnicí 3.2 a je hlavní komponentou multi-head attention modulu, znázorněným v 3.7 vlevo.

Multi-head znamená, že do modulu vstupuje 8 (dle práce [36]) trojic V,K,Q. Každá matice trojice V,K,Q je transformována lineární vrstvou a na celou trojici je potom aplikován *scaled dot-product attention* (self-attention) mechanismus. Tento postup je aplikován na každou z osmi trojic paralelně, pokaždé však s jinými váhami lineární vrstvy.

[4]Poziční kódování identifikuje pořadí tokenu ve vstupním textu.

3.3 Předtrénované modely BERT a ALBERT

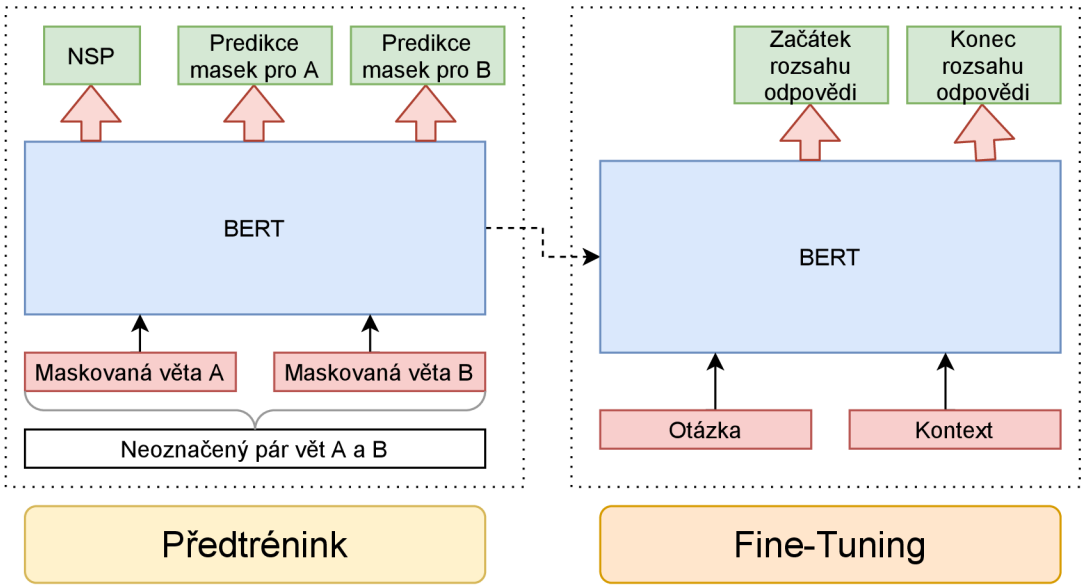
V této podkapitole jsou představeny základní modely, založené na architektuře Transformer, která byla popsána výše. Informace v této podkapitole jsou převzaty z původních článků [8] a [12].

Předtrénované modely, stavějící na této architektuře, určily nový standard pro nově vytvářené metody. Kromě jejich vysoké úspěšnosti je obrovskou výhodou také znovupoužitelnost předtrénovaných modelů. Modely je možno jednoduše dotrénovat (provést adaptaci po načtení modelu s předtrénovanými parametry) pro celou řadu úkolů na poli NLP, jako je odpovídání na otázky, rozpoznání entit, klasifikace textu a další.

3.3.1 BERT

BERT (*Bidirectional Encoder Representation from Transformers*) je model pro reprezentaci přirozeného jazyka, představený v roce 2018 prací [8]. Jeho hlavním přínosem je aplikace obousměrného tréninku architektury Transformer. Ukazuje se, že model trénovaný v obou směrech dokáže plně využít schopnosti architektury Transformer zpracovávat vstupní text naráz, nikoliv sekvenčně.

Hlavním přínosem práce [8] je způsob, kterým probíhá předtrénování na velkých textových korpusech. Trénink na úkolech, které nemusí přímo souviset s cílovým použitím, dovolí modelu získat základní pochopení vlastností jazyka pro jeho správnou reprezentaci vytvořenou enkodérem. Pro předtrénování modelu BERT byly použity následující úkoly.



Obrázek 3.8: Vlevo předtrénink enkodéru BERT na úlohách *ML modeling* a *NSP*. Vpravo předtrénovaný model BERT dotrénován na konkrétní úlohu – odpovídání na otázky. Obrázek byl inspirován obrázkem 1 z článku [8]

Maskované modelování jazyka (ML modeling) je úloha velmi podobná již dříve zmíněné úloze predikce následujícího slova ve větě. Je ale použit přístup, kdy je určité procento slov v textu *maskováno*, aby nebyl původní úkol kvůli obousměrné reprezentaci architektury Transformer triviální.

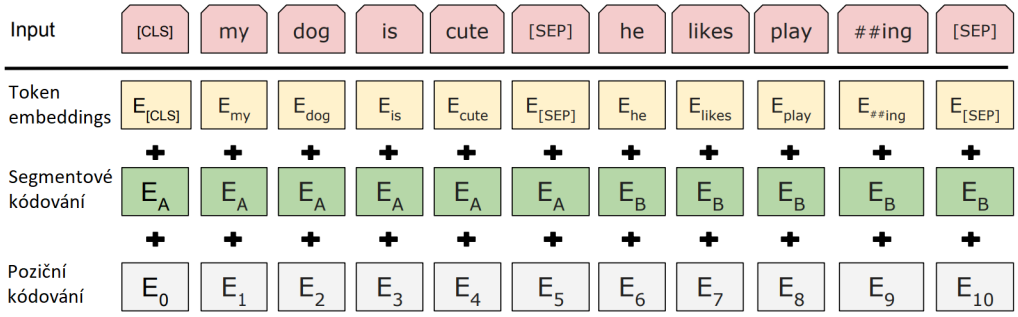
Při tréninku je 15 % slov každé věty nahrazeno tokenem [MASK], jehož původní význam je poté na základě okolního kontextu hádaný [8]. Pro tento úkol je na enkodér napojena klasifikační vrstva, pomocí které je určena pravděpodobnost pro každé slovo ve slovníku, jež by mohlo token [MASK] nahradit.

Predikce následující věty (NSP) je druhým typem úlohy pro předtrénink modelu BERT. Jeho snahou je naučit model porozumět vztahu nejen mezi jednotlivými slovy, ale taky mezi významem celých vět.

Trénovací datová sada je rozdělena na dvojice vět A a B. Model má za úkol předpovědět, je-li věta B větou navazující na větu A v kontextu. V 50 % případů je věta B opravdu větou následující. Ve zbytku je to náhodně zvolená věta z korpusu.

Práce [8] také uvádí, že navzdory jednoduchosti tohoto úkolu je jeho přínos pro pozdější aplikaci modelu pro odpovídání na otázky velký.

Formát vstupu modelu BERT je koncipován tak, aby mohl být použit pro větší spektrum aplikací, pro které je zamýšlen. Proto je umožněno na vstupu pomocí jediné sekvence tokenů reprezentovat jak jednu, tak i dvě vstupní sekvence, jako třeba otázku a odpověď.



Obrázek 3.9: Příklad formátu vstupu „*input*“ modelu BERT převzatý z [8]. Znázorněno je: poziční (*position*), segmentové kódování, token *embeddings* a speciální tokeny [CLS] a [SEP]

Pro vektorovou reprezentaci tokenů jsou používány Wordpiece embeddings [38], popsané v podkapitole 2.1.3. K těm je přidáno poziční kódování, kvůli použití architektury Transformer a segmentové kódování, které identifikuje, které sekvenci vstupu jednotlivé tokeny náleží. Příklad je naznačen na obr. 3.9.

Pro popis vstupu jsou navíc použity dva speciální tokeny [CLS] a [SEP]. Token [CLS] je přidán vždy na začátek vstupu, token [SEP] je přidán na konec každé sekvence. Token [CLS] je využit také jako výstup pro klasifikační úlohy (například předtrénink na NSP).

3.3.2 ALBERT

Informace v této podkapitole jsou převzaty ze článku [12] – *A Lite BERT for self-supervised learning of language representations.*

Hlavním problémem modelu jako BERT je jeho velikost – asi 110 milionů parametrů v jeho základní verzi. Zvětšováním modelu se většinou dá dosáhnout zlepšení, je to však problém kvůli omezené paměti grafických karet a množství času potřebného pro trénink.

ALBERT – A Lite BERT, je model představený v roce 2019 článkem [12]. Vychází z modelu BERT a dosahuje lepších výsledků na datových sadách pro porozumění textu, přestože má mnohem menší počet parametrů, než velká verze modelu BERT. V následujících odstavcích jsou shrnuty hlavní přínosy modelu ALBERT.

Faktorizace parametrů reprezentací – *Factorized embedding parameterization*
Wordpiece embeddings se učí reprezentovat tokeny nezávisle na kontextu, zatímco skryté vrstvy modelu se učí reprezentaci závislou na kontextu. Zvětšení velikosti skryté vrstvy způsobí u modelu BERT stejný nárůst velikosti reprezentace tokenů, jejichž význam není tak podstatný a model to zbytečně zatěžuje. Díky odstranění provázanosti velikosti dimenze *wordpiece embeddings* a dimenze skryté vrstvy modelu je možno lépe zvětšovat velikost skrytých vrstev bez toho, aby to ovlivnilo velikost reprezentací ve slovníku.

Sdílení parametrů mezi vrstvami – *Cross-layer parameter sharing*
Parametry napříč vrstvami jsou sdíleny, což zamezuje růstu počtu parametrů při zvětšování hloubky modelu a způsobuje jejich efektivnější využití. Ve výsledku má velká verze modelu ALBERT mnohonásobně méně parametrů, než velká verze modelu BERT.

Trénink návaznosti dvou vět – *Inter-sentence coherence loss*
Trénink modelu BERT pomocí predikce následující věty (NSP) se částečně překrýval s úkolem predikce maskovaného tokenu, protože mohla být nesouvislost vět odhadnuta na základě nepříslušnosti slov ke stejnému tématu.

Pro trénink modelu ALBERT je pro tento úkol použita stejná dvojice vět A a B, u kterých se určuje, zda B navazuje na A. Při nenávaznosti však není věta B náhodně vybraná z jiného korpusu, ale je pouze prohozená s větou A. Model je tak lépe schopen naučit se logickou souvislost vět, místo odhadu tématu, ke kterému se věty vztahují bez toho, aby na sebe logicky navazovaly.

ALBERT (resp. BERT) pro odpovídání na otázky Pro využití modelu ALBERT (resp. BERT) na konkrétní úkoly vyžadující porozumění textu je potřeba provést *naladění* (tzv. *fine-tuning*) modelu.

Pro odpovídání na otázky je na výstup modelu napojena lineární vrstva, která je natrénována na určení indexu začátku a konce odpovědi v rámci celé vstupní sekvence. Pro začátek a konec je poté vybrána validní[5] dvojice indexů, označující rozsah odpovědi.

Pro naladění na daný úkol jsou optimalizovány parametry lineární klasifikační vrstvy i enkodéru ALBERT (resp. BERT), nebo pouze lineární vrstvy (což vyžaduje mnohem kratší trénink).

[5]Podmínky jako: start idx < end idx, nebo to, že se odpověď se nenachází v první sekvenci (otázce).

Kapitola 4

Řazení dokumentů dle relevance

Tato kapitola popisuje metody pro indexaci a řazení dokumentů dle relevance, které jsou využity při vyhledávání relevantního dokumentu, rozebraném v části 7.4. Nejprve je vysvětlen význam indexu v kontextu řazení dokumentů a poté algoritmy, které jej využívají k vyhledávání. Nakonec jsou diskutovány problémy velké báze dat, jako je Wikipedie.

Informace v této kapitole jsou čerpány z prezentací stanfordského kurzu *Information Retrieval and Web Search* [18] a knihy *Introduction to Information Retrieval* [19].

4.1 Indexování dokumentů

Pro vyhledávání dokumentů s použitím webových zdrojů jako Wikipedie jsou typické rozsáhlé texty, které by bylo náročné procházet sekvenčně. Proto je důležité vytvořit strukturu, která je dále použita metodami pro řazení dokumentů a umožní optimalizované vyhledávání. Taková struktura se nazývá **index**.

Index poskytuje informace o výskytu jednotlivých termů v prohledávaných dokumentech a je vytvořen ještě před začátkem vyhledávání. Nejpoužívanější variantou je tzv. *invertovaný index*.

Před vytvořením indexu je užitečné provést co nejlepší normalizaci prohledávaných dokumentů. Metody pro zpracování textu byly podrobně popsány v kapitole 2.2 (převod na malá písmena, lemmatizace, odstranění stop slov).

Invertovaný index je struktura obsahující všechna slova (termy) prohledávaného korpusu. Pro každý term je uložen seznam dokumentů (označených *docID*), které daný výraz obsahují. Může obsahovat doplňující informace o počtu výskytů termu v dokumentu, případně přesné pozici nebo délce dokumentu. Informace jsou poté využity při řazení dokumentů, tzv. *ohodnoceném vyhledávání*.

4.1.1 Úskalí velké báze dat jako Wikipedie

Česká Wikipedie obsahuje velké množství článků (asi 3,5 GB v nekomprimované podobě). Pro prohledávání celé Wikipedie je potřeba vytvořit a udržovat index, což je pro takový objem dat poměrně výpočetně a paměťově náročné. Je složité pomocí běžných nástrojů sestavit index pro celou Wikipedii.

Dalším problémem je různá délka dokumentů, nestrukturovanost některého textu a dynamicky se měnící prostředí, kdy jsou články na Wikipedii pravidelně aktualizovány.

Způsoby prohledávání Wikipedie a vypořádání se se zmíněnými problémy jsou popsány v podkapitole 6.2 a kapitole 7, věnované implementaci.

4.2 Ohodnocené vyhledávání pro řazení dokumentů

Při dotazování na vytvořený index nestačí pouze výčet dokumentů, které termy přítomné v dotazu obsahují. Ohodnocené vyhledávání umožňuje (pomocí speciálních vztahů) získání množiny seřazených n nejrelevantnějších dokumentů.

4.2.1 TF-IDF (Term frequency – Inversed document frequency)

Četnost termu (TF – rovnice 4.1) je počet výskytů vyhledávaného termu v daném dokumentu. Vycházíme z předpokladu, že dokument s deseti výskyty termu je pro dotaz relevantnější, než dokument obsahující vyhledávaný term pouze jednou. Neznamená to však, že je dokument 10× relevantnější – růst relevance není přímo úměrný TF.

Jednotlivé termy nenesou stejnou informační hodnotu (např. stop slova, viz podkapitola 2.2.4) a ojedinělé výrazy jsou často zásadní pro relevanci k dotazu (což souvisí i s neúměrným růstem relevance s růstem TF).

$$TF_{t,d} = log(1 + tf_{t,d}) \tag{4.1}$$

Dokumentová četnost (DF) je počet dokumentů, které obsahují vyhledávaný term. Účelem je identifikace termů, vyskytujících se jenom ve zlomku dokumentů, které nejspíš ponesou zásadní informační hodnotu. Cílem je zohlednit nepřímou úměru závislosti růstu relevance dokumentu a četnosti termu. Převrácená dokumentová četnost (IDF – viz rovnice 4.2) dělí celkový počet dokumentů N dokumentovou četností pro daný term. Hodnota IDF je tedy vyšší pro vzácnější termy.

$$IDF_d = log(\frac{N}{df_t}) \tag{4.2}$$

TF-IDF je statistická metoda, která použitím uvedených dvou metrik hodnotí relevanci dokumentu. Pro výpočet váhy TF-IDF pro daný term t v dokumentu d platí tedy:

$$\text{TF-IDF}_{t,d} = TF_{t,d} \cdot IDF_d \tag{4.3}$$

Ohodnocení S dokumentu d pro dotaz q je potom sumou jednotlivých ohodnocení termů t dané otázky q.

$$S(q,d) = \sum_{t \in q \cap d} \text{TF-IDF}_{t,d} \tag{4.4}$$

4.2.2 BM25 (Best Match 25)

Skóre BM25 (označované také jako Okapi) bylo představeno v [30]. Jedná se o vylepšení klasické hodnotící funkce TF-IDF, které využívá poznatky statistiky a pravděpodobnosti pro řazení dokumentů dle jejich relevance s ohledem na vztah délky dokumentu a četnosti výskytů termu v něm.

Delší dokumenty budou mít pravděpodobně větší $tf_{t,d}$. To je potřeba ve výsledném ohodnocení zohlednit, aby nebyly delší dokumenty nutně zvýhodněny. Výsledný vzorec pro ohodnocení dokumentu d pomocí BM25 vzhledem k otázce q je:

$$S(q,d)^{BM25} = \sum_{t \in q} \left(IDF_d \cdot \frac{(k_1 + 1) \cdot tf_t}{k_1 \cdot ((1-b) + b \cdot \frac{dl}{avdl}) + tf_t} \right), \tag{4.5}$$

kde tf a idf jsou metriky vysvětlené v podkapitole 4.2.1, dl je délka dokumentu d, $avdl$ je průměrná délka dokumentu v kolekci a b, k_1 jsou konstanty.

- k_1 ovlivňuje dopad TF na výsledné skóre dokumentu. Obvykle je volena v rozmezí $k_1 \in (1,2\,;\,2)$. Pro nízké hodnoty k_1 roste význam dokumentu s rostoucí TF poměrně rychle.
- b ovlivňuje normalizaci délky dokumentu, obvykle je volena hodnota $b = 0,75$. Hodnota $b = 1$ znamená úplnou a $b = 0$ žádnou normalizaci délky dokumentu.

Výhodu hodnotící funkce BM25 oproti obyčejnému TF-IDF lze demonstrovat na následujícím příkladě.
Mějme vyhledávaný dotaz „*term frequency*“ a dva dokumenty d_1 a d_2:

	term	frequency
d_1	1024	1
d_2	16	8

	TF-IDF	**BM25**
d_1	**87**	31
d_2	75	**43**

Tabulka 4.1: Výskyt termů „term“ a „frequency“ v dokumentech d_1 a d_2 a ohodnocení jednotlivých dokumentů podle funkcí TF-IDF a BM25 ($k_1 = 2$) [18]

Tabulka 4.1 ukazuje, jak BM25 lépe zohlednilo počet výskytů termů a skutečně ohodnotilo lépe relevantnější dokument.

Pro BM25 existují také modifikace, jako BM25+ [14], BM25L [13] a další, které se snaží vypořádat se znevýhodněním příliš dlouhých dokumentů klasickou funkcí BM25. Byly porovnány v [35]. Závěr toho článku ale ukazuje, že neexistuje modifikace, která by systematicky zlepšila dosažené výsledky vyhledávání ve všech případech.

4.2.3 DPR (Dense Passage Retrieval)

DPR je technika, představená článkem [11], která využívá architekturu dvou enkodérů. Pomocí jednoho z enkodérů je každý dokument převeden na vektorovou reprezentaci. Z těch je sestaven vyhledávací index. Druhý enkodér je určen pro získání reprezentace otázky. Pro vektor otázky je poté nalezen dokument (nebo více dokumentů), jehož vektor je vektoru otázky nejblíže (rovnice 4.6).

$$sim(q, d) = E_Q(q) \cdot E_D(d) \tag{4.6}$$

V rovnici 4.6 [11] je $sim(q, d)$ skóre relevance otázky q a dokumentu d, E_Q je enkodér použitý pro reprezentaci otázky a E_D pro reprezentaci dokumentu.

Jako enkodéry jsou používány modely BERT (je tomu tak v [11], ale stejně je tomu také např. v [9] pro implementaci systému R2-D2). Ty jsou trénovány tak, aby po výpočtu blízkosti vektorů otázky a relevantních pasáží měly relevantní pasáže co nejmenší vzdálenost.

Kapitola 5

Dostupné datové sady a výběr

Tato kapitola se zabývá popisem a výběrem datových sad (datasetů) pro realizaci a evaluaci systému.

Datová sada je v oblasti strojového učení velká anotovaná kolekce dat pro trénink a vyhodnocování neuronových sítí nebo celých systémů. Obvykle obsahuje pro každý příklad vstupní data a jejich anotaci.
Datové sady je možno rozdělit podle toho, v jaké fázi vývoje s nimi pracujeme.

- **Trénovací (train)** Používá se pro naučení neuronové sítě. Při zpracovávání trénovací datové sady se model učí a upravuje své parametry. Na konci tréninku může model konvergovat ke 100% úspěšnosti na trénovací datové sadě.
- **Validační (dev)** Slouží pro průběžné vyhodnocení existujícího modelu při vývoji. Model se z validačních dat neučí. Může být využita například k úpravě hyperparametrů nebo zastavení tréninku modelu, když se jeho úspěšnost na validační datové sadě přestane zlepšovat. Vytvořena může být vybraným zlomkem trénovacího datasetu nepoužitým pro trénink.
- **Testovací (test)** Obsahuje příklady z reálného světa, na kterých je model/systém vyhodnocen po dokončení jeho vývoje. Testovací dataset je často shodný s validačním.

Pro trénink neuronových sítí pro odpovídání na otázky by měla datová sada obsahovat hlavně:

- otázku
- kontext (dokument obsahující odpověď)
- odpověď
- případně doplňující informace (jako začátek odpovědi, délku odpovědi apod.)

Vytvoření takové datové sady je složitý úkol, vyžadující velké množství lidských zdrojů. Je potřeba nasbírat dostatečné množství dokumentů, pro které někdo musí vymyslet otázky a vyznačit v dokumentech příslušné odpovědi. Tvorba datových sad obvykle probíhá pomocí *crowdsourcingu*[1].

[1] *Crowdsourcing* označuje proces získání potřebné služby, nápadů, příspěvků, či pomoci při řešení problémů, od velké skupiny lidí. (https://wikisofia.cz/wiki/Crowdsourcing [cit. 22-03-2021])

5.1 Datové sady pro angličtinu

Pro angličtinu existuje celá řada datových sad zaměřených na odpovídání na otázky. Nejznámější z nich je SQuAD (*Stanford Question Answering Dataset*) [29]. Dataset obsahuje asi 100 000 anotovaných dvojic otázka – odpověď z více než 500 různých článků z anglické Wikipedie. Existuje také rozšířená verze tohoto datasetu. SQuAD 2.0 [28] obsahuje navíc 50 000 otázek, na které v přiloženém kontextu neexistuje odpověď. Pokud tedy dokument odpověď neobsahuje, model to musí umět určit.

Další datasety jsou například NQ (Natural Questions) se zaměřením na otevřenou doménu nebo HOTPOTQA, vyžadující pro zodpovězení otázky spojování informací z více dokumentů. Čeština takovou diverzitou datových sad nedisponuje.

5.2 Datové sady pro češtinu

Čeština nabízí poměrně omezené zdroje[2] oproti anglickým datovým sadám.

Nejznámější je datová sada SQAD (Simple Question Answering Database) [32]. Obsahuje přes 13 000 příkladů otázek s odpovědí v přiloženém kontextu, pocházejícího z české Wikipedie. Datová sada obsahuje otázky směřované na časový údaj, entitu, lokaci, osobu atd., včetně 16 % otázek typu ano/ne. Velikost datasetu je přesně popsána v tabulce 5.3.

Existuje také česká verze anglického datasetu SQuAD, prezentovaná v práci [16]. Datová sada je strojovým překladem původního datasetu SQuAD 1.1 a SQuAD 2.0. Oproti anglické verzi však obsahuje asi 70 000 otázek pro první (1.1) a 110 000 otázek pro druhou (2.0) verzi datové sady SQuAD (tabulka 5.1). Ztráta části rozsahu datové sady byla způsobena problémy s vyznačením odpovědi ve strojově přeložené verzi dokumentu.

5.3 Výběr trénovacích a testovacích dat

Pro systém je potřeba vybrat data pro natrénování modelů pro extrakci odpovědi a pro validaci výsledného systému.

Pro trénink anglického modelu byl vybrán anglický SQuAD 1.1 a 2.0 (obsahující i validační část datasetu). Po zvážení bylo rozhodnuto o použití a porovnání obou verzí datasetu. Testovací data obsahují pouze otázky, pro které existuje nad otevřenou doménou odpověď, a je tedy porovnáno, která verze datasetu byla pro trénink komponenty reader vhodnější.

Pro trénink českého (vícejazyčného) modelu byl vybrán český překlad anglického datasetu SQuAD 1.1 a 2.0 [16] kvůli jeho velikosti a porovnatelnosti s anglickou verzí datasetu.

Jako testovací datová sada byl zvolen český dataset SQAD v3.0, ze kterého bylo odstraněno 16 % otázek, očekávajících odpovědi ano/ne. Datová sada nejlépe odpovídá reálnému scénáři užití systému, protože obsahuje otázky a odpovědi z článků české Wikipedie (na rozdíl od trénovacích datasetů).

V době použití byly zjištěny velikosti datových sad, které jsou uvedeny v tabulce 5.2 (neodpovídají přesně informacím z původního článku).

[2] Co se týká datasetů pro odpovídání na otázky, existuje pouze zmíněný SQAD. Angličtina nabízí datasety specializované např. pro otevřenou doménu.

Dataset		Anglické otázky	České otázky
SQuAD 1.1	Trénink	87 599	64 164
	Validace	10 570	8 739
SQuAD 2.0	Trénink	130 319	107 088
	Validace	11 873	10 845

Tabulka 5.1: Velikost datasetů SQuAD 1.1 a 2.0 pro češtinu a angličtinu podle [16]

Dataset		Anglické otázky	České otázky
SQuAD 1.1	Trénink	88 638	69 169
	Validace	10 808	9 615
SQuAD 2.0	Trénink	131 958	115 415
	Validace	12 171	12 066

Tabulka 5.2: Velikost datasetů SQuAD 1.1 a 2.0 zjištěná při jejich použití

Úskalí datasetu SQAD

Dataset SQAD má pro vyhodnocení systému nad otevřenou doménou podobné nevýhody jako SQuAD. Otázky jsou přímo vázané ke konkrétnímu článku, z čehož vyvstávají problémy pro jejich jiné využití.

Některé otázky nemusí dávat bez přiloženého kontextu smysl, a je tedy nemožné (nebo velmi obtížné) pro ně najít odpověď. Některé mohou být také nejednoznačné a správných odpovědí pro ně může být bez kontextu několik, v datasetu se však nachází jako referenční pouze jediná, která je extrakcí z přiloženého kontextu.

Dalším z možných problémů je také vázanost datasetu na časové období, kdy byl vytvořen/zveřejněn (2019). Příkladem může být otázka „Kolik evropských států tvoří Evropskou unii?“, pro kterou po brexitu už odpověď „28“ není realitou.

Procházet ručně a vybrat pro odstranění takové otázky z celé datové sady je pro jednoho člověka náročné, a proto je tento aspekt zohledněn až při vyhodnocení výstupů systému, diskutován v podkapitole 8.1.

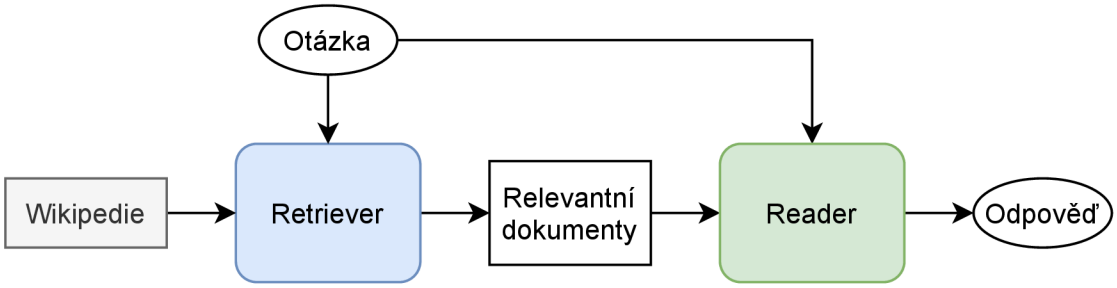
Dataset	počet záznamů	bez ano/ne
SQAD v3.0	13 476	**11 273**

Tabulka 5.3: Velikost datasetu SQAD v3.0 při jeho použití pro vyhodnocení systému

Kapitola 6

Návrh systému a jeho komponent

V této kapitole je rozebrán návrh systému, než přijde popis implementace jeho jednotlivých částí (v kapitole 7). Nejprve je popsán generický koncept systému pro odpovídání na otázky, zvolený přístup k řešení a motivace finálního návrhu. Následuje rozdělení problému na části a jejich návrh, včetně popisu blokových schémat hlavních komponent. Diskutován je návrh vyhodnocení výsledného systému v kontextu otevřené domény. Na závěr kapitoly je zmíněno několik existujících řešení.



Obrázek 6.1: Generické schéma (prezentované například v [5]) open-domain QA systému

6.1 Zvolený přístup k problému

Odpovídání na otázky nad otevřenou doménou (open-domain QA) má obvykle dvě hlavní (na sobě téměř nezávislé) části, jak je naznačeno na obrázku 6.1. První částí je *retriever*, který má za úkol získat z internetu (pro nás z Wikipedie) relevantní dokumenty, které by mohly obsahovat odpověď na danou otázku. Druhým blokem je *reader*, který provádí extrakci odpovědi ze získaného kontextu [5].

Pro návrh systému byla největším dilematem realizace části *reader* pro kvalitní extrakci odpovědi v češtině. Chtěl jsem využít některého z populárních předtrénovaných modelů (viz podkapitola 3.3), založených na architektuře Transformer [36] (podkapitola 3.2.2). Proto jsem se rozhodl vyzkoušet přístup, kdy je nativně anglickému modelu ALBERT, který dosahuje špičkových výsledků[1], strojově přeložen získaný dokument i otázka z češtiny do angličtiny.

[1] Žebříček nejúspěšnějších modelů je dostupný z https://rajpurkar.github.io/SQuAD-explorer/.

Pro porovnání jsem zvolil pro extrakci odpovědi vícejazyčný model BERT (mBERT[2]), který podporuje 104 světových jazyků s nejrozsáhlejší Wikipedií. Naladění modelu je provedeno na českém překladu datové sady SQuAD 1.1 a 2.0 [16, 29]. Výsledný systém využívá variantu, která se ukázala v experimentech jako nejúspěšnější (viz kapitola 8).

6.2 Návrh jednotlivých částí systému

V této podkapitole je popsán návrh systému, tedy hlavně částí *retriever* a *reader*. Diskutován je také návrh vyhodnocení systému na datasetu SQAD v3.0.

6.2.1 Návrh části retriever

Získání n relevantních dokumentů, které považujeme za nejlepší kandidáty pro zodpovězení otázky, vyžaduje několik na sebe navazujících kroků (viz obrázek 6.2). V návrhu a implementaci jsou použity metody předzpracování textu a řazení dokumentů, popsané v kapitolách 2 a 4.

Retriever dostává na vstupu otázku, která je také hlavním vstupem do celého systému. Dříve, než se na základě otázky začnou přímo prohledávat odstavce české Wikipedie, je dobré z otázky získat co nejvíce informací, které by mohly vést k těm *správným článkům*. Až po jejich získání jsou jednotlivé články rozděleny na odstavce, které jsou následně řazeny. K získání článku jsou použity tři metody:

- **základní Wiki Search** nezpracované otázky, který někdy přináší uspokojivé výsledky, ale většinou je nedostačující,
- **získání relevantního titulku článku** (pomocí BM25 – viz 4.2.2) z úplného výpisu všech titulků a abstraktů české Wikipedie[3] po lemmatizaci otázky a odstranění stop slov,
- **rozpoznání (pojmenované) entity** v otázce, která by se mohla vyskytovat v titulku článku a usnadnit tím *Wiki Search*.

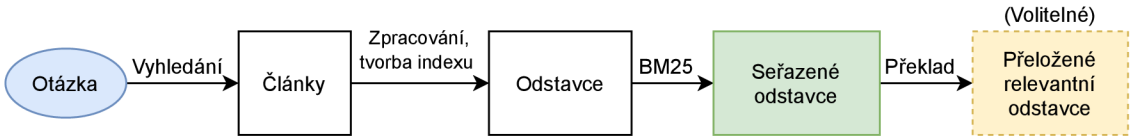
Tímto je tedy získán seznam článků z Wikipedie, který je následně upraven tak, aby se v něm každý získaný článek vyskytoval pouze jednou.

Jednotlivé články je následně třeba rozdělit na odstavce, normalizovat jejich délku, lemmatizovat, odstranit stop slova a vytvořit index. Odstavce jsou poté seřazeny hodnotící funkcí BM25 a z nich vybrány první tři s nejlepším skóre. Pro vyhledávání mezi odstavci je použita lemmatizovaná otázka s odstraněnými stop slovy. Tři nejvhodnější odstavce jsou získány v jejich původní, nezpracované podobě a případně (při použití nativně anglického modelu ALBERT) jsou odstavce i otázka přeloženy do angličtiny.

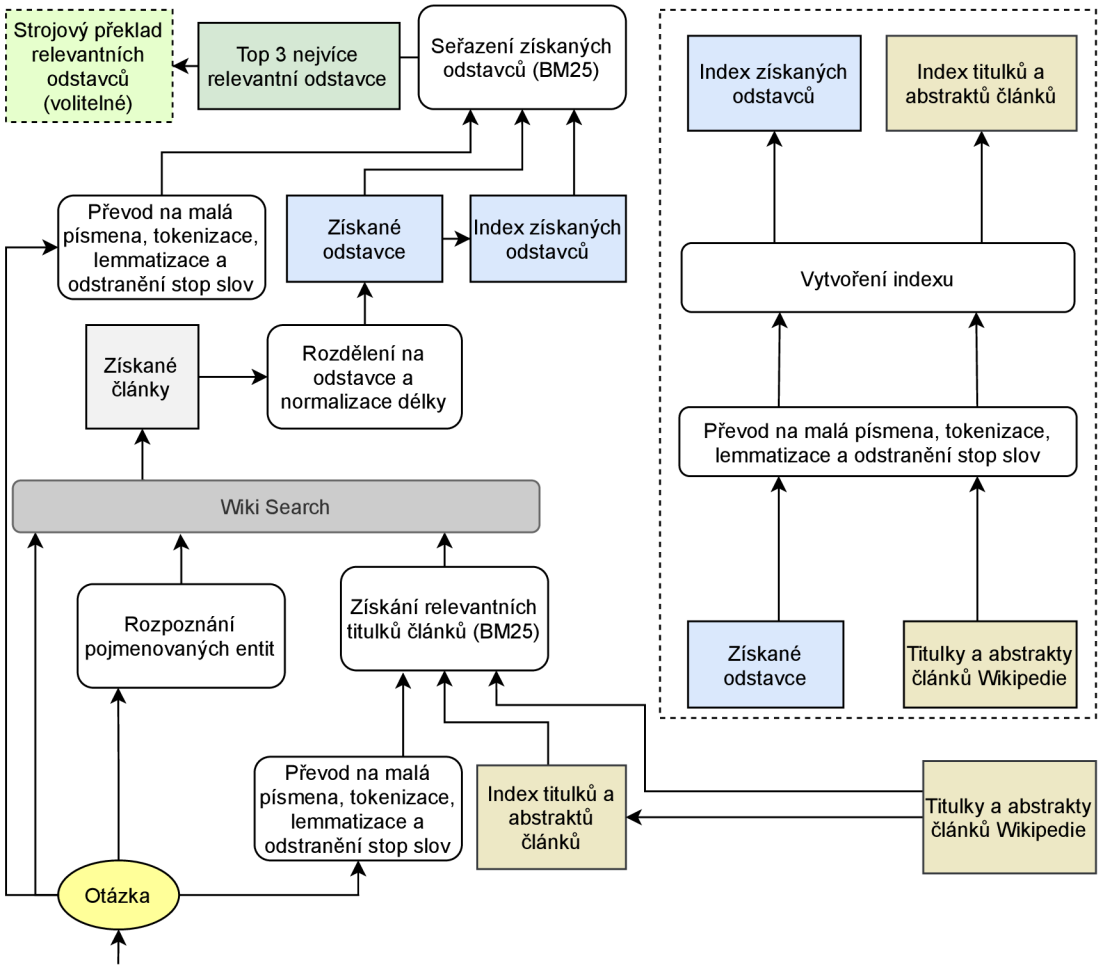
Následuje návrh struktury komponenty *retriever*, znázorňující podrobně jednotlivé komponenty na obrázku 6.3.

[2] https://github.com/google-research/bert/blob/master/multilingual.md – model BERT předtrénovaný na korpusech mnoha jazyků

[3] https://cs.wikipedia.org/wiki/Wikipedie:St%C3%A1hnut%C3%AD_datab%C3%A1ze



Obrázek 6.2: Zjednodušené schéma komponenty *retriever*

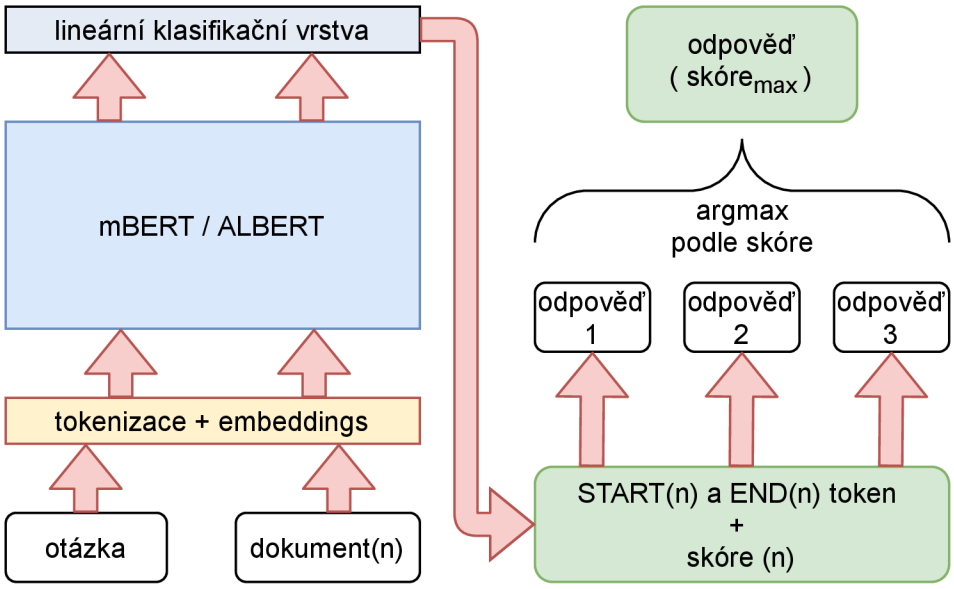


Obrázek 6.3: Podrobné schéma komponenty *retriever*

6.2.2 Návrh komponenty reader

Část systému pro extrakci odpovědi ze získaného dokumentu byla navržena ve dvou variantách.

- **Anglický model ALBERT** s lineární klasifikační vrstvou pro určení začátku a konce odpovědi, pro který jsou vstupní dokumenty a otázka strojově přeloženy do angličtiny. Výsledná odpověď je přeložena zpět do češtiny.
- **Vícejazyčný model BERT (mBERT)**, opět s lineární klasifikační vrstvou pro určení začátku a konce odpovědi, který dokáže odpovídat přímo v češtině.



Obrázek 6.4: Schéma komponenty *reader* (kombinovaně pro oba navržené systémy) pro 3 relevantní dokumenty, kde $n \in (1, 3)$. Pro model ALBERT by musel proběhnout ještě strojový překlad finální odpovědi do češtiny

Pro obě varianty je postup získání odpovědi (až na překlad pro anglický model ALBERT, který je však součástí části *retriever*) shodný.

Model dostane k přečtení tři nejrelevantnější odstavce, získané komponentou *retriever*. Pro každý odstavec je provedena extrakce odpovědi, kdy je vybrána vždy právě jedna validní odpověď s největším „skóre“ (nenormalizovaná logaritmická pravděpodobnost) pro daný dokument. Ze získaných odpovědí je vybrána ta, kterou si je model nejvíce jistý (má nejvyšší skóre), a případně (při použití modelu ALBERT) je přeložena zpět do češtiny. Ostatní odpovědi jsou také ukládány – pro získání statistických dat při vyhodnocení. *Reader* tak vybere kromě odpovědi také dokument/odstavec, ze kterého je odpověď získána. Může tak být použit pro získání některých dalších informací/kontextu.

Schéma komponenty *reader* pro extrakci finální odpovědi je naznačeno na obrázku 6.4. Modely, které byly použity, byly blíže popsány v kapitole 3, konkrétně v podkapitole 3.3.

6.2.3 Návrh vyhodnocení

Jak už bylo zmíněno, pro vyhodnocení byl vybrán český dataset SQAD v3.0. Musely však být navrženy metriky, respektující charakteristiku datové sady a také to, že se systém zaměřuje na otevřenou doménu. Kromě tradičních metrik EM (*exact match*) a F1 (popsány blíže v podkapitole 8.1) jsem se při návrhu vyhodnocení systému snažil zohlednit několik aspektů:

- Kvůli otevřené doméně může být odpověď nalezena z jiného kontextu, v jiném tvaru.
- Potřebujeme i metriku pro ohodnocení samotné komponenty *retriever* v úspěšnosti nalezeného dokumentu.
- Je nutné zohlednit i správné odpovědi, které byly nalezeny, i když nebyly vybrány komponentou *reader* jako nejvhodnější.

Podrobnější popis zvolených metrik je uveden v podkapitole 8.1. Metriky jsou použity pro porovnání variant systému i vyhodnocení výsledné verze.

6.3 Existující systémy pro otevřenou doménu

Příkladem existujícího systému pro QA nad otevřenou doménou je systém DrQA, popsaný v dříve zmíněném článku [5], prezentující koncept návrhu *retriever* a *reader*. Systém je poměrně jednoduchý, svojí strukturou inspiroval i moje řešení. Byl navržen pro odpovídání na otázky nad anglickou Wikipedií a jeho úspěšnost EM na datasetu SQuAD 1.1 je podle [5] necelých 30 %.

Novějším systémem je R2-D2 [9], který patřil k nejúspěšnějším systémům soutěže NeurIPS 2020 [25]. R2-D2, neboli *Rank twice – Read twice*, používá architekturu sestávající ze čtyř částí: *retriever, reranker,* extraktivní a generativní *reader*. Retriever používá DPR [11] (popsáno v části 4.2.3) a reranker využívá enkodér architektury Transformer. Jeho úspěšnost na datasetu NQ (Natural Questions) dosahuje 55 % EM. Implementace R2-D2 se navíc snaží omezit paměťovou náročnost modelu. Pro modely bez omezení dosahuje úspěšnost těch nejlepších F1 skóre více než 60 %[4].

Pro češtinu existuje systém AQA (Automatic Question Answering), poprvé prezentovaný v práci [20] a od té doby zdokonalovaný. Zmíněná práce z roku 2016 poskytuje navíc *benchmark* pro evaluaci systému nad otevřenou doménou pro dataset SQAD (v1.0). Pozdější vyhodnocení pro zdokonalený systém a dataset SQAD v1.1 z roku 2018 jsou uvedeny v článku [21]. Výsledky systému, vyhodnoceného v roce 2018, jsou shrnuty v tabulce 6.1. Novější vyhodnocení systému fungujícího nad otevřenou doménou pro dataset SQAD v3.0 nejsou k dispozici.

AQA	EM [%]	počet testovacích vzorků
SQAD v1.0	49,83	3301
SQAD v1.1	38,08	3301

Tabulka 6.1: Úspěšnost systému AQA v přesné shodě s referenční odpovědí na datových sadách SQAD 1.0 a 1.1

Paralelně s vývojem mého systému vzniká také diplomová práce s názvem *Vícejazyčný systém pro odpovídání na otázky nad otevřenou doménou* studenta Bc. Michala Slávky, pod vedením Ing. Martina Fajčíka. Ta se zabývá tvorbou open-domain systému schopného odpovídat na otázky v 17 jazycích. Pro komponentu retriever používá práce taktéž hodnotící funkci BM25. Co se týká části reader, experimentuje práce, podobně jako ta má, s více přístupy. Jeden z nich vyhledává dokumenty pouze v angličtině a využívá strojového překladu původních otázek. Odpověď v angličtině je pak znovu přeložena do původního jazyka. Druhý se zabývá použitím vícejazyčného modelu. Pro reader je použit, na rozdíl od této práce, generativní model T5 a (vícejazyčný) mT5.

[4]Nejlepší výsledky na datasetu Natural Questions jsou dostupné z https://ai.google.com/research/NaturalQuestions/leaderboard.

Kapitola 7

Implementace a použité technologie

Obsahem této kapitoly je výčet použitých nástrojů a popis implementace jednotlivých částí systému. Je popsáno, jak a pomocí jakých knihoven byl návrh z kapitoly 6 realizován a jaké problémy bylo potřeba vyřešit. Rozebráno je také zpracování dat potřebných pro funkci systému i jeho vyhodnocení.

7.1 Použité nástroje a technologie pro implementaci

V této podkapitole jsou popsány technologie a knihovny použité při implementaci výsledného systému.

Pro implementaci byl vybrán jazyk **Python** (3.7.10), který nabízí největší množství knihoven a nástrojů pro realizaci systému. Zároveň je vhodný ke zpracování dat a je sympatický svou jednoduchou syntaxí. Nabízí také prostředí *Jupyter notebook*, které je velmi pohodlné pro práci s daty a vývoj.

Některé knihovny implementují poznatky popsané v kapitolách 2, 3 a 4. Knihovny běžného charakteru, jako je například `json`, jsou z následujícího výčtu vynechány.

- **PyTorch**[1] je open-source framework pro strojové učení, zejména pro zpracování obrazu a přirozeného jazyka.

  Nabízí implementace mnohých druhů neuronových sítí a tensorových operací s GPU akcelerací (CUDA).

- **HuggingFace – Transformers**[2] je populární (open-source) implementací nejznámějších architektur Transformers, jako je BERT, ALBERT, RoBERTa, XLM atd., vyvinutá společností HuggingFace. Nabízí nástroje pro jednoduchou práci s modely, předzpracování jejich vstupu, trénink apod. Pro implementace modelů obecného využití typu BERT nabízí také jejich verze pro konkrétní úlohy, jako je odpovídání na otázky (extrakce odpovědi). Pro každou architekturu je k dispozici typicky několik předtrénovaných verzí (BERT base, BERT large . . .), které je možné dotrénovat pro specifický úkol.

[1] https://pytorch.org/
[2] https://huggingface.co/transformers/

- **HuggingFace – Datasets**[3] umožňuje snadnou práci s veřejně dostupnými datasety. Usnadňuje jejich načtení a efektivní předzpracování.

  Díky dostupnosti zdrojových kódů umožňuje také úpravu skriptu pro načtení jiné datové sady, která doposud v oficiálním seznamu knihovny není dostupná.

- **DeepPavlov**[4] je open-source framework, postavený na Pytorch, TensorFlow[5] a Keras, pro usnadnění vývoje a výzkumu v oblasti zpracování přirozeného jazyka, konkrétně dialogových systémů.

- **Rank-bm25**[6] je knihovnou implementující hodnotící algoritmy z rodiny BM25 [35] (Okapi, BM25+, BM25L).

  Nabízí velmi jednoduché API pro indexaci a řazení dokumentů.

- **NumPy**[7] nabízí jako knihovna s licencí BSD velmi efektivní operace pro práci s vektory, maticemi a vícerozměrnými poli.

- **NLTK**[8] (Natural Language Toolkit) je open-source knihovna nabízející nástroje pro zpracování textu a přirozeného jazyka.

  Prostřednictvím jednoduchého API umožňuje tokenizaci (včetně dělení na věty), stematizaci, nebo tagování částí textu.

- **MorphoDiTa**[9] je open-source nástrojem pro morfologickou analýzu, umožnující například tokenizaci nebo lemmatizaci českého textu [34]. Pro použití je potřeba také natrénovaný jazykový model, který je k dispozici na webových stránkách[9] pro češtinu a angličtinu. Jednoduché API pro práci s MorphoDiTou nabízí knihovna **corpy**[10].

- **Majka**[11] [42] je morfologickým analyzátorem, založeným na morfologických slovnících. Slovníky jsou dostupné pro celou řadu jazyků, mj. pro češtinu, slovenštinu, polštinu a angličtinu.

- **Wikipedia (api)**[12] je knihovna, usnadňující vyhledávání, získávání a parsování článků z Wikipedie. Nabízí velmi jednoduché API pro vyhledávání článků a získávání jejich textového obsahu.

- **Googletrans**[13] nabízí API pro přístup k funkcionalitě Překladače Google. Jedná se o neoficiální implementaci přístupu k překladači a kvůli omezením na množství překládaných znaků nezajišťuje stabilní funkčnost.

[3] https://huggingface.co/docs/datasets/
[4] https://deeppavlov.ai/
[5] https://tensorflow.org/
[6] https://github.com/dorianbrown/rank_bm25
[7] https://numpy.org/
[8] https://www.nltk.org/
[9] https://ufal.mff.cuni.cz/morphodita
[10] https://corpy.readthedocs.io/en/stable/api/morphodita.html
[11] https://nlp.fi.muni.cz/czech-morphology-analyser/
[12] https://github.com/goldsmith/Wikipedia
[13] https://github.com/ssut/py-googletrans

- **Jupyter Notebook**[14] je open-source webová aplikace pro interaktivní spouštění Python kódu v tzv. buňkách. Kromě zrojového kódu umožňuje také vkládání textu, vizualizaci dat, vkládání obrázků a grafů nebo spouštění skriptů.
- **Google Colab**[15] je služba, nabízející přístup k virtuálnímu stroji, umožňující spouštění Jupyter Notebooku v cloudu. Virtuální stroje nabízí (bez rozšířené placené verze) asi 12 GB RAM, úložiště s možností připojení a přístupu ke Google Drive a výkonné GPU s pamětí 16 GB, vhodné pro akceleraci výpočtů. Dokonce jsou dostupná i běhová prostředí s TPU pro paralelizaci výpočtů. Běhová prostředí vždy disponují množstvím předinstalovaných, běžně používaných knihoven (PyTorch, TensorFlow a další).

7.2 Zpracování používaných dat

Tato podkapitola popisuje data, se kterými se v průběhu implementace nakládá. Nejprve je popsáno zpracování datasetů SQuAD, na kterých je trénována komponenta *reader*. Poté je objasněno zpracování *dumpů*, které obsahují titulky a abstrakty článků české Wikipedie. Vysvětleno je také parsování českého datasetu SQAD v3.0 a formát uložení odpovědí systému pro pozdější evaluaci.

7.2.1 Předzpracování datové sady SQuAD

Pro stažení a zpracování datasetu SQuAD byla použita knihovna Huggingface `datasets`, která načte trénovací dataset dostupný ve formátu JSON. Pro načtení je použit skript *squad.py*, který lze díky open-source povaze knihovny upravit a použít i pro načtení českého překladu datasetu SQuAD [15] (viz *czech_squad.py* a *czech_squad2.py*). Stejně jednoduchý postup lze použít i pro načtení verze 2.0 datasetu.

Po načtení obsahuje dataset položku *train* a *validation* – pro trénink a vyhodnocení modelu pro extrakci odpovědi. Velikosti jednotlivých částí jsou zmíněny v tabulce 5.2. Každá otázka ve SQuAD datasetu obsahuje id, titulek, kontext (dokument), otázku a odpověď. Možných odpovědí může být i více. Pro odpověď je také uveden index znaku, na kterém v pasáži začíná.

Z knihovny `transformers` je potřeba načíst tokenizér, který převede otázku a kontext každé položky v datasetu do podoby stravitelné modelem (tokeny reprezentovány pomocí token ID předtrénovaného slovníku modelu a speciální tokeny jako [CLS] a [SEP]). Pro vícejazyčný BERT lze použít `BertTokenizerFast`, pro ALBERT zas `AlbertTokenizerFast`.

Modely BERT a ALBERT založené na Transformer architektuře mají pevně danou maximální délku vstupní sekvence tokenů. Proto je potřeba tokenizér nastavit tak, aby byly příliš dlouhé vstupy rozděleny na sekvenci kratších, ideálně s nějakým přesahem za sebou následujících oddělených částí (`return_overflowing_tokens=True`).

Při práci s odpovědí je v datasetu vyznačen pouze index počátečního znaku. Nastavením tokenizéru `return_offsets_mapping=True` získáme pro každý token také indexy vstupního řetězce, na kterých se daný token nachází.

[14] https://jupyter.org/
[15] https://colab.research.google.com/notebooks/intro.ipynb

S použitím popsaných metod jsou k původnímu datasetu přidány informace o indexu *počátečního a koncového tokenu* každé odpovědi (může se lišit pro různé tokenizéry).

Funkce, která zpracování implementuje, byla vypůjčena z HuggingFace notebooku[16] a pomocí metody `map()` třídy `Dataset` je aplikována na všechny vzorky datasetu.

7.2.2 Práce s dumpem Wikipedie

V práci byly použity dva různé soubory pro získání relevantních článků Wikipedie. Prvním je dump s názvy článků a druhým je dump abstraktů.

Soubor s názvy článků je pouze jednoduchý text snadný pro zpracování. Na každý řádek souboru připadá jeden název článku. Před vytvořením indexu je tedy třeba pouze nahradit podtržítka v názvech mezerou a uložit názvy jednotlivých článků do seznamu.

Pro abstrakty článků je to o něco složitější, protože jsou uloženy v poměrně rozměrném XML souboru. Každý záznam obsahuje titulek, url, text abstraktu a odkazy. Po načtení je dump zpracován tak, že je ponechána pouze informace o názvu článku a text jeho abstraktu. Výsledný zpracovaný dump je uložen jako soubor JSON o asi čtvrtinové velikosti původního XML souboru. Získání a zpracování konkrétních článků je popsáno až v podkapitole 7.4.

7.2.3 Extrakce informací z datasetu SQAD

Dataset SQAD v3.0 [22] je dostupný ve velmi specifické struktuře. Jednotlivé otázky datové sady jsou rozesety po 13 476 složkách (každá obsahuje jeden záznam), z nichž každá čítá 7–9 souborů formátu VERT. Důležité pro nás jsou pouze soubory obsahující otázku a extrakci odpovědi. Na obrázku 7.1 je příklad formátu, ve kterém jsou otázky a odpovědi uloženy.

```
<s>
Kdy       kdy       k6eAd1
se        se        k3xPyFc4
narodil   narodit   k5eAaPmAgMnS
herec     herec     k1gMnSc1
Ivan      Ivan      k1gMnSc1
Trojan    Trojan    k1gMnSc1
<g/>
?         ?         kIx.
</s>
```

Obrázek 7.1: Příklad formátu položky (otázky) datasetu SQAD v3.0

Pro každý záznam musí být tedy otevřeny a zpracovány soubory obsahující otázku a odpověď. První sloupec souboru obsahuje na každém řádku *token* otázky, resp. odpovědi, druhý sloupec jejich lemmata. Při zpracování je také potřeba brát ohled na speciální znaky.

Pro každý záznam je uložena otázka, extrakce odpovědi a lemma extrakce odpovědi. Záznamy obsahující ano/ne odpovědi byly ze zpracování vynechány. Zpracovaný dataset je uložen ve formátu JSON (obr. 7.2) pro jeho pozdější snadnější a rychlejší použití.

[16] Notebook je dostupný z adresy https://colab.research.google.com/github/huggingface/notebooks/blob/master/examples/question_answering.ipynb.

```
{
  1: {
      question: "Kdo je autorem novely Létající jaguár? ",
      answer: "Josefa Formánka ",
      answer_lemma: "Josef Formánek "
  },
  2: {
      question: "Jak se nazývá věda zabývající se houbami? ",
      answer: "mykologie ",
      answer_lemma: "mykologie "
  },
```


Obrázek 7.2: Příklad formátu zpracovaného datasetu SQAD v3.0

7.2.4 Ukládání dat pro vyhodnocení systému

Pro uložení výsledků byl opět zvolen formát JSON, aby mohla být evaluace provedena až po získání odpovědí.


```
3: {
    question: "Jaký je oficiální název Rakouska? ",
    answer: "Rakouská republika ",
    answer_orig: "Republic of Austria",
    answer_sqad: "Rakouská republika ",
    answer_sqad_lemma: "rakouský republika ",
  articles: [
        "Dějiny Rakouska",
        "Oficiální server",
        "Mistrovství Evropy ve volejbale žen 2007",
        "Rakousko"
    ],
    document: "Rakousko, plným názvem Rakouská republika
  all_answers: [
        " Rakouská říše ",
        " Rakouská republika ",
        " Österreich"
    ],
  scores: [
        "12.37",
        "15.91",
        "6.53"
    ],
  all_documents: [
        "Záhy po svém nástupu na trůn musel František II. (1
        "Rakousko, plným názvem Rakouská republika (něme
        "Německý název Österreich pochází ze staroněmecke
    ]
},
```


Obrázek 7.3: Příklad formátu uložené odpovědi s doplňkovými informacemi

Pro každý zodpovězený záznam je uloženo číslo otázky – „3“, otázka – *question*, odpověď – *answer*, originální/nepřeložená odpověď – *answer_orig* (při použití modelu mBERT shodná s *answer*), referenční odpověď dle datasetu SQAD – *answer_sqad* a její lemma – *answer_sqad_lemma*. Dále jsou pro podrobnější analýzu výsledků k dispozici názvy článků – *articles*, které systém prohledával, kontext – *document*, ze kterého byla finální odpověď vybrána, první tři nejlepší odpovědi – *all_answers* a skóre – *scores* jednotlivých odpovědí z *all_answers*. K dispozici jsou také plné texty nejrelevantnějších pasáží získaných komponentou retriever – *all_documents*.

Soubor zodpovězených záznamů je pak pojmenován podle toho, jaký rozsah otázek obsahuje a přidáno je i časové razítko.

answers_1-3000___04-04-2021_09:53.json

7.3 Implementace a trénink komponenty Reader

Pro implementaci komponenty *reader* byla použita knihovna `transformers` od HuggingFace, která nabízí velmi jednoduché rozhraní pro trénink modelů.

7.3.1 Trénink

Pro trénink ALBERT modelu byla použita třída `AlbertForQuestionAnswering` a předtrénovaný model `albert-base-v2`[17] (11 milionů parametrů). Model není příliš velký a dosahuje podobných výsledků, jako jeho větší verze `large` nebo `xlarge`. Pro jeho naladění (*fine-tuning*) byl použit dataset SQuAD 1.1 a 2.0.

Pro druhý model byl použit `BertForQuestionAnswering` z knihovny `transformers` a předtrénovaný model `bert-base-multilingual-cased`[18] (110 milionů parametrů). Pro fine-tuning byl použit český překlad datasetu SQuAD 1.1 a 2.0. Pro načtení obou datasetů byl použit upravený skript `squad.py`[19] (viz skript `czech_squad(2).py`).

Všechny modely byly trénovány pomocí třídy `Trainer` s následujícími parametry:

- *batch size* (velikost dávky) – 16
- *learning rate* (rychlost učení) – $2e{-}5$
- *weight decay* – $0,01$
- *počet trénovacích epoch* – 3

Po naladění je úspěšnost modelů (viz podkapitola 8.1) na jejich validačních datasetech následující (tabulka 7.1). Oba modely byly trénovány na GPU ve službě Google Colab.

ALBERT	SQuAD 1.1	SQuAD 2.0
EM	82,14	78,33
F1	89,58	81,55

mBERT	cz_SQuAD 1.1	cz_SQuAD 2.0
EM	68,52	66,47
F1	77,72	69,54

Tabulka 7.1: Úspěšnost modelů na validačních datasetech (v %)

7.3.2 Extrakce odpovědi

Třída zajišťující extrakci odpovědi se nazývá `Reader` a pro extrakci odpovědi z kontextu je nejprve potřeba vytvořit její instanci. Většina parametrů pro inicializaci je volitelná, musí však být zadána alespoň cesta k natrénovanému modelu v úložišti (`model_checkpoint`) a jeho typ (`model_type`). Může být upravena také maximální délka získané odpovědi (pomocí parametru `max_answer_length`) a max. počet validních odpovědí (`n_best_size`),

[17] https://huggingface.co/albert-base-v2
[18] https://huggingface.co/bert-base-multilingual-cased
[19] https://github.com/huggingface/datasets/blob/master/datasets/squad/squad.py

které model vybere. `Reader` načte tokenizér a model, který bude pro extrakci odpovědi využívat a použije pro jejich akceleraci GPU, pokud je k dispozici.

Pro extrakci odpovědi je poté možné volat jedinou instanční metodu `get_answers`, která vyžaduje jako parametry otázku a kontext. Vrací pole o velikosti `n_best_size` obsahující nejlepší kandidáty pro odpověď. Každá odpověď se skládá z textu "text" a skóre "score".

Pro každý dokument je vybrána nejlépe ohodnocená odpověď s neprázdným textem.

7.4 Implementace komponenty Retriever

Pro získání relevantních dokumentů je implementována třída `Retriever`. Pro její inicializaci je třeba zadat cestu k modelu pro nástroj MorphoDiTa (`dita_file`), který provádí lemmatizaci, cestu k souboru s titulky článků Wikipedie (`wiki_titles`) a ke zpracovanému JSON souboru s abstrakty Wikipedie (`wiki_abstracts`). Místo MorphoDiTy může `Retriever` použít i nástroj Majka (`majka=True`), pro který je potřeba zadat cestu k souboru s morfologickým slovníkem (`majka_file`). Pokud není k dispozici zpracovaný soubor s abstrakty, lze původní dump abstraktů převést na formátovaný JSON pomocí funkce `parse_abstracts()`.

Při vytváření instance třídy `Retriever` je načten lemmatizátor, sestaven index titulků článků a index abstraktů článků Wikipedie, načten tokenizér (rozdělení na věty) a model pro rozpoznávání entit (pro stažení je třeba nastavit parametr `download_ner_model=True`). Lze dodat také vytvořený index abstraktů (parametr `index_file`).

Použitá stop slova – Pokud je potřeba při zpracování textů v průběhu vyhledávání relevantního dokumentu odstranit stop slova, je pro to využit následující seznam [37]:

> kdy být a se v na ten on že s z který mít do já o k i jeho ale svůj jako za moci pro tak po tento co když všechen už jak aby od nebo říci jeden jen můj jenž ty stát u muset chtít také až než ještě při jít pak před však ani vědět hodně podle další celý jiný mezi dát tady tam kde každý takový protože nic něco ne sám bez či dostat nějaký proto

Dále je k dispozici seznam pro odstranění interpunkce:

> . , ? ! ... "( ) ; - /

Tvorba indexu – Pro vytvoření indexu pro prohledávání je použita třída `BM25Okapi` dříve zmíněné knihovny `rank-bm25`. Každý titulek/abstrakt je převeden na malá písmena, lemmatizován a poté jsou z něj odstraněna stop slova. Získány jsou zpracované tokeny každého titulku/abstraktu. Index titulků/abstraktů je poté vytvořen pomocí `BM25Okapi` z takto zpracovaných titulků/abstraktů.

Rozpoznání pojmenovaných entit (NER - *Named Entity Recognition*) je další úloha na poli NLP. Uveďme alespoň příklad jejího použití:

$$\text{Kdy vynalezl Alfred Nobel dynamit?} \longrightarrow \text{Alfred Nobel}$$

NER může usnadnit vyznačení klíčových slov pro vyhledávání.

Pro rozpoznání pojmenované entity v otázce je použita knihovna `deeppavlov`. Využívá natrénovaný model `ner_ontonotes_bert_mult`, který je načten funkcí `build_model()`.

7.4.1 Získání relevantních dokumentů

Inicializace instance třídy `Retriever` trvá, hlavně kvůli tvorbě indexů a načtení modelu pro NER, poměrně dlouho[20]. Po jeho inicializaci je potřeba pro získání relevantních dokumentů pouze jediná instanční metoda `retrieve()`, která pro zadanou otázku získá n relevantních dokumentů (dle parametru `max_docs`).

Získání dokumentů probíhá v těle metody `retrieve()` následovně:

1. Nejprve jsou získány názvy článků metodou `get_doc_list()`, které jsou výsledkem vyhledávání metody `search()` knihovny `wikipedia`. Jednotlivé vstupy do vyhledávání jsou následující:

   - V otázce jsou rozpoznány pojmenované entity (pomocí NER).
   - Otázka je převedena na malá písmena, lemmatizována a jsou z ní odstraněna stop slova.
     - Pomocí metody vytvořeného indexu titulků (`get_top_n()`) je získán nejrelevantnější titulek článku vzhledem k otázce.
     - Pomocí metody vytvořeného indexu abstraktů (`get_top_n()`) je získán nejrelevantnější titulek článku vzhledem k otázce.
   - Otázka je pro vyhledání použita nezpracovaná.

   Pokud jsou některé výsledky vyhledávání stejné, je použit pouze jeden z nich.

2. Pro každý získaný název článku je získán jeho textový obsah metodou `page()` knihovny `wikipedia`. Ten je rozdělen na jednotlivé odstavce a jsou z něj odstraněny nepotřebné části jako reference a odkazy.

3. Je zkontrolováno, jestli některý z odstavců nepřesahuje maximální délku. Je-li potřeba odstavec dále rozdělit, je použita metoda `normalize_length()`, která využívá tokenizéru knihovny `nltk`. Ten rozdělí odstavec na věty, které spojuje, dokud není dosaženo maximální délky pasáže. Pro zbývající věty je vytvořen další odstavec, který obsahuje také poslední 2 věty odstavce předcházejícího.

4. Po normalizaci délky získaných odstavců je potřeba získat jejich zpracovanou verzi, aby mohl být sestaven index pro vyhledávání. Každý odstavec je tedy převeden na malá písmena, lemmatizován a jsou z něj odstraněna stop slova (a interpunkce). Pro každý odstavec je získán seznam takto normalizovaných tokenů, které jsou použity pro tvorbu indexu `BM25Plus` (v závislosti na vyhodnocení provedené v článku [35]) pro vyhledávání.

5. Původní lemmatizovaná otázka bez stop slov převedená na malá písmena je použita pro vyhledání v indexu získaných odstavců pomocí metody `get_top_n()`. Tím jsou získány (nezpracované[21]) nejrelevantnější odstavce – pasáže pro danou otázku.

[20] Inicializace trvá asi 10 minut, pokud není potřeba stáhnout `deeppavlov` model pro NER. Pokud lze načíst vytvořený index pomocí modulu `pickle`, může to být pouze kolem jedné minuty.

[21] Zpracování odstavců probíhá pouze pro tvorbu indexu jednotlivých pasáží.

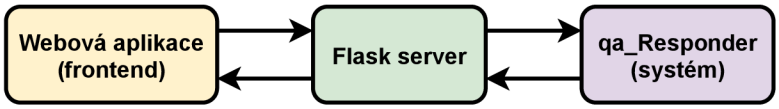
Nalezení odpovědi na otázku

Celý tok programu systémem je následující:

- Je vytvořena instance třídy pro extrakci odpovědi `Reader`, instance třídy pro získání relevantních dokumentů `Retriever` a také instance překladače `Translator` knihovny `googletrans` (použití modelu ALBERT).
- Po zadání otázky je volána funkce `find_answer()`, která získá odpověď na otázku.
  - Pomocí komponenty *retriever* jsou získány relevantní dokumenty
  - V případě použití nativně anglického modelu ALBERT jsou dokumenty spolu s otázkou přeloženy pomocí funkce `translate()` do angličtiny.
  - Pro každý získaný dokument je provedena extrakce odpovědi a vybrána nejlépe ohodnocená neprázdná odpověď.
  - Ze získaných extrakcí odpovědi z jednotlivých dokumentů je vybrána ta, která má nejvyšší skóre. Použita je metoda `argmax()` knihovny `numpy`. Pomocí získaného indexu (nejvyšší hodnoty skóre) je vybrán dokument i odpověď.
  - Vybraná odpověď je při použití modelu ALBERT přeložena zpět do češtiny.
  - Funkce vrací odpověď. Pro podrobné vyhodnocení dále: dokument, ve kterém byla odpověď nalezena, seznam nejlepších odpovědí, seznam skóre nejlepších odpovědí, seznam článků Wikipedie, které byly pro nalezení odpovědi procházeny a seznam nejlepších pasáží, ze kterých byla provedena extrakce.

Implementace demonstrační aplikace

Pro předvedení funkčnosti systému byla vytvořena demonstrační aplikace[22], pro kterou byl implementován webový server. Ten je napojen na již dříve existující webový klient. V rámci práce byl tedy pro aplikaci vytvořen pouze backend.



Obrázek 7.4: Schéma architektury demonstrační webové aplikace

Schéma architektury je naznačeno na obrázku 7.4. Pro realizaci jednoduchého serveru byl použit framework Flask[23]. Pomocí něj je implementována obsluha několika jednoduchých požadavků, které jsou klientem zasílány.

V případě dotazu `GET` může server poskytnout konfiguraci modelu, výčet předdefinovaných ukázkových otázek a JSON schémata, která určují strukturu konfigurace, požadavku a odpovědi.

Při zaslání požadavku `POST` získá server navíc data ve formátu JSON, která obsahují otázku a požadovanou konfiguraci systému. Server volá pro získání odpovědi metodu `find_answer()` třídy `qa_Responder`, která zastřešuje systém pro odpovídání.

[22] Demonstrační aplikace je dostupná z adresy `http://r2d2.fit.vutbr.cz/.cs`.
[23] `https://flask.palletsprojects.com/`

Metoda získá odpověď na otázku a vrátí ji v určeném formátu. Server poté zašle odpověď ve formátu JSON klientovi, a tím je obsloužen jeho požadavek.
Odpověď serveru ve formátu JSON obsahuje:

- otázku,
- jednotlivé odstavce, jejich skóre (získáno hodnotící funkci BM25) a titulek článku každé pasáže,
- jednotlivé odpovědi a jejich skóre, pro každou index odstavce, ze kterého pochází a také index znaku začátku, resp. konce odpovědi (tzv. `char_offset`).

Maximální počet získaných pasáží a odpovědí je určen konfigurací požadované klientem, která je zaslána jako součást dotazu.

Obrázek 7.5: Příklad použití rozhraní webové aplikace. Na obrázku je vidět otázka, nejlepší nalezená odpověď a jeden ze získaných úryvků textu. U daného úryvku je uveden název článku, ze kterého byl získán, skóre získané komponentou retriever a nejlepší odpovědi. Pro každou odpověď je uvedeno její skóre dle extrakce komponentou reader

Kapitola 8

Vyhodnocení systému a rozbor chyb

V této kapitole jsou popsány výsledky dosažené vytvořeným systémem. Popsán je vliv použitého modelu, trénovacího datasetu a použitého nástroje pro morfologickou analýzu. Nejprve jsou popsány standardní i speciální metriky, na nichž je úspěšnost systému prezentována. Porovnány jsou jednotlivé varianty systému a je provedena i jejich ruční evaluace. Výsledky jsou poté poměřeny s jiným existujícím řešením. Jsou analyzovány chyby a jejich příčiny. Kapitola končí diskuzí o dalším vývoji a vylepšeních.

8.1 Vysvětlení základních metrik

Tato podkapitola vysvětluje metriky použité pro vyhodnocení systému. Popsány jsou standardní metriky jako EM a F1 používané pro QA systémy a také metriky zachycující specifika otevřené domény.

Metody EM a F1 (používané pro vyhodnocení modelů na datasetech SQuAD [29, 28]), MRR (např. pro [32]) a částečně T3M jsou standardní metriky používané pro vyhodnocení většiny podobných systémů. Ostatní metriky jsou obměnou těch standardních, navržené specificky pro vyhodnocení tohoto systému.

8.1.1 Exact Match (EM)

Exact match, neboli přesná shoda, je jednoduchá standardní metrika, která ale může diskriminovat dostačující odpovědi s pouze malými rozdíly oproti referenčním výsledkům. Její hodnota může být buď 1 (pokud je shoda dosažena) nebo 0 (pokud se oproti referenční hodnotě výsledek liší, byť jen v jednom znaku). Na příkladu je ukázka penalizace odpovědi, která byla dokonce specifičtější, než referenční odpověď.

otázka: Kde se narodil Gabriel Jonas Lippmann?
referenční odpověď: Bonnevoie
odpověď systému 1: Bonnevoie, Lucembursko – **EM** = 0
odpověď systému 2: Bonnevoie – **EM** = 1

Při vyhodnocování je referenční i získaná odpověď převedena před porovnáním na malá písmena. Při vyhodnocování varianty systému s modelem ALBERT je brána v úvahu i původní, nepřeložená odpověď.

8.1.2 Exact Lemma Match (ELM)

ELM je speciálně navržená metrika pro toleranci drobných rozdílů oproti referenční odpovědi. Kromě převedení na malá písmena jsou odpovědi před porovnáním také lemmatizovány. Skóre ELM je tedy typicky vyšší, než EM.

> **otázka:** Kdy se koná dostihový závod Velká pardubická?
> *referenční odpověď:* druhou říjnovou neděli
> *odpověď systému:* druhá říjnová neděle
> **EM** = 0 , **ELM** = 1

Skóre ELM nabývá opět hodnoty buď 1 (správně), nebo 0 (špatně), stejně jako metrika EM. ELM pro příklad uvedený v 8.1.1 by bylo také 0.

8.1.3 Naivní skóre (NS)

Naivní skóre je nejbenevolentnější metrikou. Zjišťuje se, zda-li je odpověď, nebo její lemma, podřetězcem referenční odpovědi, nebo jejího lemmatu, případně naopak. Snaha je tedy zachytit co nejvíce alespoň částečně správných odpovědí. Naivní skóre (NS) příkladu uvedeného v podkapitole 8.1.1 a 8.1.2 je tedy 1.

8.1.4 Top 3 Match (T3M)

T3M je obdobou EM (podkapitola 8.1.1), pro shodu je však testována každá z nejlepších třech (3) nalezených odpovědí (případně TkM – nejlepších k odpovědí). Můžeme tedy určit, v kolika případech (oproti EM) systém našel správnou odpověď, ale nevybral ji, protože nedosáhla nejvyššího skóre. T3M nabývá opět hodnoty 0, nebo 1.

8.1.5 MRR

MRR (*Mean Reciprocal Rank*) je obdobou T3M, zohledňuje však, na kolikátém místě byla odpověď nalezena (T3M zohledňuje pouze to, jestli vůbec). Pokud má odpověď shodná s tou referenční nejvyšší skóre (byla vybrána), hodnota MRR je 1, pokud byla na druhém místě, hodnota je 1/2, pokud na třetím, hodnota MRR je 1/3 a tak dále. Pokud nebyla odpověď nalezena vůbec, hodnota MRR je 0 [7].

8.1.6 F1 skóre

Je jakýmsi kompromisem mezi naivním skóre a EM. Při jeho výpočtu jsou používány dvě metriky – *recall* (úplnost) a *precision* (přesnost) [19].

$$\mathbf{F1} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{8.1}$$

$$recall = \frac{tp}{tp + fn} \quad precision = \frac{tp}{tp + fp} \tag{8.2}$$

- tp – *true positive* – znaky (počet) společné pro získanou i referenční odpověď
- fn – *false negative* – znaky, které jsou v referenční odpovědi, ale chybí v získané
- fp – *false positive* – znaky, které nejsou v referenční odpovědi, ale přebývají v získané

Obvykle jsou definice tp, fn a fp uváděny pro tokeny, ne znaky, chceme ale více zohlednit morfologii češtiny a drobné rozdíly v odpovědích, vztahující se k otevřené doméně. Například augustiánský/augustiánských – při porovnávání tokenů bychom nedosáhli žádné shody. F1 skóre je počítáno pouze, pokud je získaná odpověď podřetězcem té referenční, nebo naopak.

Pro příklad:

> **otázka:** Odkud byla anglická kapela The Beatles?
> *referenční odpověď:* z Liverpoolu
> *odpověď systému:* Liverpoolu

je tedy $precision = 1$, $recall = 0,83$ a $\mathbf{F1} = 0,91$.

Pro příklad z 8.1.1

> **otázka:** Kde se narodil Gabriel Jonas Lippmann?
> *referenční odpověď:* Bonnevoie
> *odpověď systému:* Bonnevoie, Lucembursko

je $precision = 0,41$, $recall = 1$ a $\mathbf{F1} = 0,82$.

F1 tedy zohledňuje přesnost i úplnost odpovědi a je velmi směrodatnou metrikou při odpovídání na otázky.

8.1.7 Lidské vyhodnocení

Otevřená doména je problematická pro automatické vyhodnocování. Jak je uvedeno například v [25], otázka může mít více možných odpovědí, pokud není jednoznačně položena. Odpověď může být také obecnější, nebo specifičtější (př. z podkapitoly 8.1.1), ale stále odpovídat na otázku. Z článku [25] tedy vyplývá, že systémy jsou hodnoceny lépe při vyhodnocení lidmi. Konkrétně, procento odpovědí označených jako „rozhodně správně“ bylo u systémů v průměru o 17–25 % lepší, než pouhé porovnání řetězců odpovědí pomocí EM.

Proto je náhodně vybráno 200 otázek pro lidské vyhodnocení každé varianty systému, aby se vzal alespoň minimální ohled na diskriminační povahu automatické evaluace.

8.2 Vyhodnocení výsledného open-domain systému

Pro vyhodnocení celého systému byla vybrána datová sada SQAD v3.0 [32], ze které byly odebrány ano/ne otázky.

Pro každou metriku z podkapitoly 8.1 bylo použito vzorce 8.3:

$$\text{skóre} = \frac{1}{N} \cdot \sum_{k=1}^{N} S_k, \tag{8.3}$$

kde S_k je skóre S (např. EM) pro odpověď k a N je celkový počet odpovědí. Skóre na celém datasetu je tedy průměrem jednotlivých skóre S_k všech zodpovězených otázek.

V této podkapitole jsou shrnuty a porovnány výsledky systému na testovací sadě. Pro každou variantu systému bylo provedeno vyhodnocení pouze na (stejné) čtvrtině[1] datasetu, která byla získána vybráním každého čtvrtého vzorku. Pro vítěznou variantu bylo vyhodnocení provedeno na všech (více jak 11 000) otázkách.

[1] Velikost čtvrtiny datasetu je **2819 vzorků** – srovnatelné s počtem vzorků při vyhodnocení AQA v [21].

8.2.1 Model ALBERT a strojový překlad

Tato část prezentuje vyhodnocení systému s reader modelem ALBERT, pro který jsou získané pasáže a otázka strojově překládány do angličtiny. Vyhodnocení proběhlo pro 4 varianty: porovnán byl trénovací dataset – SQuAD 1.1/2.0 (SQ1.1/2.0) a nástroj použitý pro morfologickou analýzu – Majka/MorphoDiTa (Maj/MD). Nejlepší výsledky a důležité metriky, jako EM a F1, jsou zvýrazněny tučně.

ALBERT	**EM**	ELM	NS	T3M	MRR	**F1**	prec.	rec.
SQ1.1+MD	11,96	25,63	37,77	23,61	21,17	22,93	21,80	25,81
SQ2.0+MD	12,85	26,08	39,57	23,63	21,66	23,87	22,77	26,80
SQ1.1+Maj	11,64	25,37	37,90	**23,78**	21,40	23,03	21,88	25,98
SQ2.0+Maj	**12,88**	**26,15**	**39,60**	23,60	**21,68**	**24,11**	**22,98**	**27,09**

Tabulka 8.1: Úspěšnost systému [%] s reader modelem ALBERT a strojovým překladem textů – vyhodnoceno pro čtvrtinu datasetu SQAD v3.0 (2819 otázek)

Jak je vidět v tabulce 8.1, výsledky tohoto přístupu nejsou příliš dobré. Za povšimnutí stojí výrazný rozdíl mezi EM a ELM. Příčinou tak velké odchylky od přesné shody je nejspíš ztráta původního tvaru slov/frází při překladu, kdy však slovní základ zůstává stejný. Naivní skóre dosahuje téměř 40 %, nějaké shody tedy systém dosáhl, ale nepřesnosti způsobené překladem převládají.

8.2.2 Model vícejazyčný BERT

Vyhodnocení systému používajícího model vícejazyčný BERT jako reader proběhlo opět ve čtyřech variantách, stejně jako v předchozí podkapitole 8.2.1. Značení je stejné s jediným rozdílem, kdy SQ1.1/2.0 jsou českými strojovými překlady datové sady SQuAD použité pro trénink komponenty reader. Nejlepší výsledky jsou opět zvýrazněny tučně.

mBERT	**EM**	ELM	NS	T3M	MRR	**F1**	prec.	rec.
SQ1.1+MD	41,22	43,17	59,67	46,68	43,73	52,31	54,77	53,07
SQ2.0+MD	**42,85**	**44,84**	**61,83**	**47,11**	**44,80**	**54,33**	**56,94**	**55,04**
SQ1.1+Maj	40,90	42,85	59,13	46,29	43,38	51,79	54,27	52,55
SQ2.0+Maj	42,36	44,38	61,16	46,65	44,32	53,63	56,26	54,35

Tabulka 8.2: Úspěšnost systému [%] s reader modelem mBERT trénovaném na českém překladu datasetu SQuAD – vyhodnoceno pro čtvrtinu datasetu SQAD v3.0 (2819 otázek)

Při použití vícejazyčného modelu BERT bylo dosaženo lepších výsledků (tab. 8.2) a jeho nejlepší varianta byla dále vyhodnocena na všech 11 273 otázkách testovací datové sady.

Následuje ještě ale podkapitola o ručním vyhodnocení lidmi, která zachycuje diskriminační povahu automatických metrik a lépe charakterizuje použitelnost systému.

8.2.3 Vyhodnocení člověkem

Z celé datové sady SQAD v3.0 bylo náhodně vybráno 200 otázek, které byly vyhodnoceny také ručně – lidmi (inspirováno [25]). Porovnána na nich je úspěšnost jednotlivých systémů dle automatického vyhodnocení[2] a vyhodnocení lidmi. Je tedy zjišťováno, zda by danou odpověď systému bral člověk jako postačující.

Následující tabulka 8.3 prezentuje přehledně zjištěné hodnoty.

ALBERT	**EM**	**F1**	**člověk**
SQ1.1+MD	11,00	24,46	55,50
SQ2.0+MD	11,00	24,48	54,50
SQ1.1+Maj	11,00	24,48	**56,50**
SQ2.0+Maj	**12,00**	**24,92**	53,50

mBERT	**EM**	**F1**	**člověk**
SQ1.1+MD	43,00	53,15	64,00
SQ2.0+MD	**46,00**	**56,81**	**69,00**
SQ1.1+Maj	42,00	52,55	63,00
SQ2.0+Maj	45,00	56,21	67,50

Tabulka 8.3: Porovnání automatického a ručního vyhodnocení variant systémů vyhodnocených v 8.2.1 a 8.2.2 na 200 náhodně vybraných otázkách (v %)

Pro metriku „člověk“ byla odpověď ohodnocena 1, pokud byla při ručním vyhodnocení posouzena jako dostačující. U výsledků je vidět rozdíl mezi EM a lidským vyhodnocením až 45,5 % (pro systém využívající strojový překlad a reader ALBERT). Až 23 % pak pro systém využívající vícejazyčný BERT.

Výsledky tedy indikují použitelnost odpovědi až v 70 % případů (pro nejlepší variantu systému). Také ukazují mnohem vyšší použitelnost výsledků prvního přístupu (reader ALBERT), než se při automatickém vyhodnocení mohlo zdát. Nedosahují však kvalit systému využívajícího vícejazyčný BERT.

Pro ilustraci je uvedeno pár příkladů, které byly při ručním vyhodnocení posouzeny jako dostačující:

č.	**Otázka**
1	Kolik má čeština slovních druhů?
2	Kdo je Oldřich Homuta?
3	Která diecéze má jako hlavní patronku svatou Zdislavu?
4	Jaké je jméno současného papeže?
5	Kdo je vrchním velitelem ozbrojených sil ČR?
6	Kde zemřel Platón?

č.	**Odpověď systému**	**Referenční odpověď**
1	deset	10
2	český elektrotechnik	vynálezce remosky
3	litoměřické diecéze	litoměřický
4	kardinál Bergoglio	František
5	Miloš Zeman	prezident České republiky
6	Atény	v Athénách

Tabulka 8.4: Příklady odpovědí systému (různých variant z vyhodnocovaných), které byly posouzeny člověkem jako dostačující

[2] U vzorku vybraného pro ruční vyhodnocení byly pro porovnání zvoleny pouze metriky EM a F1.

8.2.4 Porovnání dosažených výsledků

Vyhodnocení jednotlivých variant systému proběhlo v předchozích podkapitolách, a mohou být tedy shrnuty závěry experimentů.

Pro variantu ALBERT došlo oproti variantě mBERT pouze k velmi malé přesné shodě získaných výsledků. Pro obě varianty systému byl úspěšnější *reader* trénovaný na verzi 2.0 datasetu SQuAD, tedy na tom, který pro trénink obsahuje i velké množství dokumentů neobsahujících odpověď na položenou otázku. Tento trénink nejspíš přispěl k častějšímu výběru správného dokumentu.

Co se týká lexikálních analyzátorů a jejich vlivu na funkci hodnotící funkce BM25, zvítězil v první variantě analyzátor Majka, ne však příliš přesvědčivě a procentuální rozdíl oproti MorphoDiTě je téměř zanedbatelný. Pro druhou variantu zvítězil analyzátor MorphoDiTa, a to o něco přesvědčivěji. Ve všech experimentech byl však rozdíl v systémech používajících různý lexikální analyzátor velmi malý. Kvůli tomu, že byl však ve druhé (úspěšnější) variantě nástroj MorphoDiTa systematicky (byť jen o trochu) lepší, je použit pro vyhodnocení systému na celé testovací datové sadě právě on.

Co se týká ručního vyhodnocení lidmi, byly všechny varianty mnohem úspěšnější, než podle automatického vyhodnocení. Větší rozdíl mezi automatickým a lidským vyhodnocení byl zaznamenán u první varianty systému s reader modelem ALBERT, kde jsou fráze deformovány opakovaným překladem textu do angličtiny a zpět do češtiny. Reálná použitelnost této varianty systému však není tak špatná, jak se z automatického vyhodnocení zdá.

Z provedených experimentů je tedy vítěznou variantou systém mBERT SQ2.0+MD. Tedy systém používající jako reader vícejazyčný model BERT natrénovaný na českém překladu datasetu SQuAD 2.0 a nástroj MorphoDiTa pro morfologickou analýzu.

8.2.5 Vyhodnocení nejlepšího dosaženého systému

V této části jsou uvedeny výsledky vyhodnocení nejlepší varianty systému na celém datasetu SQAD. Experimentováno bylo s množstvím dokumentů, které jsou předány komponentě reader k extrakci odpovědi. V tabulce 8.5 jsou uvedeny výsledky vyhodnocení systému.

	EM	ELM	NS	TkM	MRR	**F1**	prec.	rec.
top 3	42,30	44,21	61,20	46,67	44,27	53,69	56,12	54,44
top 5	43,71	45,64	62,80	50,02	46,41	55,20	57,62	55,95
top 8	**43,89**	**45,83**	**62,96**	52,36	47,34	**55,42**	**57,70**	**56,28**
top 10	43,77	45,68	62,87	**53,07**	**47,48**	55,28	57,52	56,17

Tabulka 8.5: Vyhodnocení systému [%] na celém datasetu SQAD 3.0 (bez ano/ne otázek – viz tabulka 5.3) s využitím Wikipedie jako báze znalostí. Porovnáno je množství čtených dokumentů „top k“. TkM je varianta metriky T3M (podkapitola 8.1) pro různý počet top k čtených dokumentů. Varianta systému: extraktivní reader – vícejazyčný BERT trénován na datasetu cz_SQuAD 2.0 (viz podkapitola 7.3), morfologický analyzátor – MorphoDiTa

Při vyhodnocení na celé testovací sadě se hodnoty od výsledků na čtvrtině datasetu v podkapitole 8.2.2 o mnoho neliší. Z tabulky 8.5 vyplývá, že nejlepší pro maximalizaci skóre EM i většiny dalších je čtení 8 nejvíce relevantních dokumentů. Při hodnotě 10 už úspěšnost většiny metrik klesla. Zvýšilo se však skóre TkM i MRR, což je s větším množstvím získaných dokumentů očekávatelné.

Porovnání výsledků práce se současným stavem poznání

V této podkapitole jsou výsledky práce porovnány s podobnými systémy stejného účelu, pro které je dostupné vyhodnocení na standardních metrikách.

systém	dataset	počet testovacích otázek	**EM**	**NS**
AQA 2018	SQAD v1.0	3 301	49,83	59.58
AQA 2018	SQAD v1.1	3 301	38,08	46,26
Tato práce	SQAD v3.0	11 273	43,89	62,96

Tabulka 8.6: Porovnání výsledků této práce se systémem AQA [20] (uvedené výsledky jsou převzaty z [21]). Výsledky jsou uvedeny v %

Výsledky práce jsou porovnány s výsledky systému AQA [20], konkrétně s poslední dostupnou vyhodnocenou variantou tohoto systému z roku 2018 [21], která byla vyhodnocena na 3301 otázkách z datasetu SQAD v1.0 a 1.1. Výsledky jsou shrnuty v tabulce 8.6 a jsou porovnány pomocí metrik EM a NS. Metrika NS není v práci specifikována, pracuje se v ní však s „plnou shodou“ (EM) a „částečnou shodou“, která je definována jako odpověď, která se přesně neshoduje s referenční, protože některá slova přebývají, nebo chybí. *Částečná shoda* uváděná při vyhodnocení systému AQA je tedy značená jako NS – naivní skóre (viz podkapitola 8.1), což je metrika této práce, která popisu nejvíce odpovídá. NS pro AQA je součtem hodnot *plné* a *částečné* shody (uvedených v [21]).

Z výsledků uvedených v tabulce 8.6 je patrné, že systém AQA převyšuje výsledky této práce pouze v metrice EM, a to pouze na části zastaralého datasetu SQAD v1.0. Z tabulky je patrné (a je tak uvedeno i v práci [21]), že při vyhodnocení na stejně velkém vzorku vybraného z aktualizované verze datasetu 1.1 úspěšnost systému výrazně klesla (z EM 49,83 % na EM 38,08 %). Takový (možná i výraznější) pokles by šlo očekávat i při vyhodnocení systému na datasetu SQAD v3.0 použitého v této práci, která navíc vyhodnocuje systém na výrazně větším počtu vzorků. Systém prezentovaný v mé práci navíc dosahuje výrazně vyšších hodnot naivního skóre, které koresponduje s procentem použitelných výsledků více, než přesná shoda (jak je uvedeno v části 8.2.3).

Výsledky práce tedy považuji v porovnání s dostupným vyhodnocením systému AQA, který představuje relevantní porovnání s výsledky mé práce, za úspěch.

Výsledky mohou být porovnány také se *state of the art* systémy, které však existují pouze pro angličtinu a jsou vyhodnoceny na datasetu NQ. Například v podkapitole 6.3 zmíněný systém R2-D2 (55 % EM), nebo nejlepší výsledky na datasetu NQ uvedené v jeho oficiálním žebříčku (až 64 % F1). Rozdílnost datasetu (a jazyku) sice neposkytuje příliš validní porovnání, jsem ale přesvědčen, že výsledky této práce takových kvalit nedosahují. Přesto si systém v porovnání s existujícími systémy nevede špatně.

8.3 Rozbor chyb a možnosti dalšího vývoje

V této podkapitole jsou rozebrány nedostatky systému, příčiny nesprávných odpovědí a možnosti vylepšení/nahrazení jeho existujících komponent pro jeho optimálnější funkci.

8.3.1 Analýza chyb a jejich příčin

Dataset je pro analýzu chyb nejprve rozdělen na 5 částí, které byly vyhodnoceny samostatně. Z každé části jsou uloženy nesprávné odpovědi. Ty jsou blíže zkoumány pro část testovacích vzorků s nejnižší úspěšností. Úspěšnost na jednotlivých částech je uvedená v tabulce 8.7.

	EM	**F1**	**NS**	**MRR**
1	57,69	65,70	71,26	60,89
2	40,75	**52,09**	59,69	44,61
3	41,65	53,57	61,54	44,89
4	**38,70**	53,36	62,19	**42,16**
5	40,68	52,38	**60,07**	44,18

Tabulka 8.7: Vyhodnocení systému na jednotlivých pětinách datasetu (1–5), kde každá pětina má asi 2255 vzorků. Pro první pětinu jsou tedy vyhodnoceny vzorky s číslem 1 až 2255. Tučně jsou uvedeny nejnižší hodnoty (v %)

Nejlépe dopadla při vyhodnocení s velkým náskokem první část vyhodnocovací sady. Nejhůře část 4 (otázky 6765–9020), kde přesná shoda nedosáhla 40 %. Pro tuto část je tedy proveden rozbor chyb. Pro rozbor byly vybrány vzorky, u kterých nebylo dosaženo přesné shody, nebo bylo dosažené F1 skóre menší než 0,5. Získáno tak bylo 982 vzorků pro bližší analýzu.

Prvním směrodatným údajem je skóre TkM – tedy přesná shoda v alespoň jedné extrakci odpovědi, která byla pro přečtené dokumenty provedena. Hodnota tohoto skóre dosahuje 15,78 %. Tyto chyby jsou tedy způsobeny špatným výběrem získané odpovědi, která je závislá na skóre určeném komponentou *reader*.

Ze zbývajících 827 vzorků bylo dosaženo NS 13,66 %, což indikuje alespoň minimální použitelnost tohoto zlomku získaných odpovědí. Zbývá tedy zjistit příčinu chyb u 714 odpovědí, kde nebylo dosaženo žádné shody.

Pro zbývající vzorky bylo provedeno vyhodnocení, které má za úkol zjistit, jestli se referenční odpověď nachází v některém z top n (pro analyzovaný systém $n = 8$) dokumentů. Ta byla nalezena ve 37,11 % analyzovaných vzorků. Pro tyto shody bylo vypočítáno také skóre MRR 25,68 %, které zhodnocuje i pořadí relevance těchto dokumentů, ve kterém byly získány komponentou retriever.

Z původních 982 analyzovaných chybných vzorků můžeme tedy 533 považovat za chybnou extrakci odpovědi, nebo její špatné ohodnocení a vybrání nesprávné. Pro zbylých 449 vzorků nebyla správná odpověď nalezena v jediném ze 353 získaných dokumentech. Vzhledem k původnímu vzorku pětiny datasetu (2255 zodpovězených otázek) je to tedy asi 16 %. Výsledky této analýzy prezentuje tabulka 8.8.

typ chyby	**zastoupení (v %)**	**počet**	**původ chyby**
nevybrání správné odpovědi	15,78	155	reader (54 %)
nízká shoda s referenční odpovědí	11,51	113	
chybná extrakce (nulová shoda)	26,99	265	
nenalezení relevantního dokumentu	35,95	353	retriever / auto. vyhod.
správná odpověď (ručně)	9,77	96	

Tabulka 8.8: Příčiny chyb v analyzovaném vzorku 982 chybně zodpovězených otázek

Pro otázky, které nebyly správně zodpovězeny a byly klasifikovány jako chyba komponenty retriever, protože nedošlo k nalezení relevantního dokumentu obsahujícího referenční odpověď, bylo provedeno ruční vyhodnocení. V těchto 449 otázkách bylo nalezeno 96 odpovědí, které jsou při ručním vyhodnocení člověkem postačující. Zbylo tedy 353 vzorků a z ručního vyhodnocení vyplývá několik hlavních příčin chyb:

- mnohoznačnost otázky/otázka nedává bez kontextu smysl,
  - „Proč se Severu nedostávalo zkušených vojevůdců?“, „Proč Franco zaútočil na severu?“, „Co udělal Smith v roce 1999?“, ...
- správná odpověď se nachází mezi nevybranými, je však v jiném tvaru,
- po převodu na malá písmena (viz 7.4) se ztrácí část kontextu,
- nebyly získány dostatečně relevantní dokumenty,
- neschopnost extraktivního modelu identifikovat ve složitém kontextu odpověď – je také možné, že byla nalezena, neměla však nejvyšší skóre v dané pasáži.

8.3.2 Další vývoj a možnosti navazujících prací

Práce pokládá základ systému, který nabízí velký prostor pro další vylepšení. Návrhy toho, jak by mohlo být řešení rozšířeno například v rámci navazujících prací je popsáno v této části.

Nejdříve, co se týká první části systému (retriever) pro získávání relevantních pasáží. Řazení pomocí hodnotící funkce BM25 by mohlo být nahrazeno modernějším přístupem DPR popisovaným v části 4.2.3, ať už za použití vícejazyčného modelu BERT, nebo jiného modelu (enkodéru) pro získání vektorové reprezentace. Pro získávání pasáží by oproti přístupu v této práci, kdy jsou procházeny pouze relevantní články, mohla být vhodně zpracována a indexována celá Wikipedie, případně by mohly být zpracovány i zvláštní struktury, jako tabulky, kde se často nachází spousta relevantních informací. Při zpracování textu by mohl být vylepšen také převod na malá písmena, aby nedocházelo ke ztrátám kontextu.

Co se týká modulu pro extrakci odpovědi, jsou cesty dalšího vývoje velmi otevřené. Prvním vylepšením by mohl být modul pro odhad předmětu otázky, aby došlo k lepší shodě při výběru odpovědi (otázka se ptá na datum – uvažujme pouze odpovědi, které jsou také datum). Zajímavé by mohlo být také experimentovat s další variantou modelu BERT použitelného pro češtinu – například Slavic BERT [2], nebo moderním vícejazyčným modelem XLM-RoBERTa [6]. Mimo existující předtrénované enkodéry by měl být předmětem výzkumu také předtrénink ryze české varianty (například modelu ALBERT). Takový pokus již sice byl uskutečněn (práce [40]), model však nedosahoval dostatečných kvalit. Dále by se dalo experimentovat s přidáním generativního (abstraktního) modelu reader, jak je učiněno například v [9] (systém R2-D2).

Zajímavá by mohla být práce s datasetem COSTRA [4], který se zaměřuje na embedding na úrovni vět, konkrétně pochopení jejich významu bez ohledu na drobné rozdíly v jejich formulaci. Modifikace, na které se dataset zaměřuje, jsou různé druhy parafrází, generalizace, konkretizace, převrácení významu a další. Dataset původně vznikl pro češtinu a mohl by pomoct k jejímu lepšímu pochopení modelem.

Přínosné by bylo také další rozšíření datasetu SQAD nebo tvorba nového datasetu zaměřeného na otevřenou doménu, obsahující více reálných příkladů toho, na co se lidé ptají na internetu – ať už pro účel tréninku jednotlivých komponent (např. DPR), nebo pro reálnější vyhodnocení schopností takového systému.

Kapitola 9

Závěr

Práce se zabývá tvorbou systému schopného odpovídat na otázky nad českou Wikipedií.

Po rozebrání teorie vztahující se k tématu popisuje práce zvolený přístup k problému a strukturovaný návrh celého systému. Popsány jsou použité nástroje a to, jak byl výsledný systém implementován. Představeno je několik variant, pro které byly provedeny experimenty a jejich podrobné vyhodnocení. Analyzovány jsou výsledky, příčiny chyb i možnosti dalšího vývoje. Porovnány jsou také morfologické analyzátory MorphoDiTa a Majka a jejich vliv na získání relevantních pasáží při použití hodnotící funkce BM25.

Podařilo se vytvořit systém, který splňuje požadavky zadání. Při řešení se experimentovalo se dvěma přístupy: použití anglického modelu ALBERT, pro který je otázka se získanými pasážemi strojově překládána, a použití vícejazyčného modelu BERT. Oba modely byly natrénovány pro extrakci odpovědi. Při tréninku modelu ALBERT byl použit dataset SQuAD 1.1 a 2.0. Pro trénink vícejazyčného modelu BERT (mBERT) byl vybrán strojový překlad těchto datasetů do češtiny.

Z provedených experimentů se překládání pasáží pro model ALBERT neukázalo jako nejlepší varianta realizace komponenty reader. Přístupu však nelze upřít alespoň částečnou použitelnost a potenciál využití pro jazyky s méně zdroji. Lepší výsledky by šlo očekávat při vyhledávání v anglických dokumentech, kdy by byla pro model přeložena pouze otázka (zadání se však zaobírá českou Wikipedií). Pomohl by také lepší způsob překladu získaných pasáží a odpovědí.

Co se týká morfologických analyzátorů Majka a MorphoDiTa, ukázaly se být pro tento úkol téměř ekvivalentní. Nejlépe se osvědčila varianta systému využívající vícejazyčný model BERT trénovaný na českém překladu datasetu SQuAD 2.0 a nástroj MorphoDiTa pro lemmatizaci. Tato varianta systému byla vyhodnocena na datasetu SQAD v3.0 čítajícím 11 273 otázek (bez ano/ne).

Pro systém byla provedena analýza chyb a jejich příčin a také ruční vyhodnocení části chybně zodpovězených otázek. Ruční vyhodnocení nasvědčuje, že by mohla být reálná použitelnost odpovědí až o 23 % vyšší (až 69 %), než ukazuje metrika přesné shody EM. Pro systém využívající anglický model ALBERT je tento rozdíl mnohem větší (12,9 % EM a přes 50 % při lidském vyhodnocení), což je způsobeno deformací frází při opakovaném překladu pasáže/odpovědi. Diskutována jsou také úskalí použití datasetu SQAD pro vyhodnocení takového systému.

Nejlepší systém dosahuje úspěšnosti 43,9 % EM a 55,4 % F1 skóre. Převyšuje tím výsledky systému AQA z roku 2018, který je nejrelevantnějším porovnáním pro výstupy práce. Nasazuje také laťku pro další práce na podobných systémech. Pro tuto variantu systému byla vytvořena demonstrační aplikace, která prezentuje funkčnost výsledného řešení.

Po případném rozšíření by mohla sloužit jako nástroj pro sběr dat využitelných při rozšiřování nebo tvorbě datasetů.

Hlavním přínosem práce je zhodnocení několika přístupů při tvorbě odpovídacího systému pro češtinu. Výsledný systém, který dosahuje velmi dobrých výsledků, poskytuje základ pro navazující práce, které by měly cílit na vylepšení jeho jednotlivých komponent.

Další vývoj by se mohl zabývat použitím DPR pro efektivnější ohodnocené vyhledávání pasáží textu. Mělo by být zahrnuto také zpracování a vytvoření indexu celé Wikipedie, včetně vhodného nakládání s údaji v tabulkách. Kvalitu extrakce odpovědi by mohlo zlepšit začlenění modulu pro klasifikaci otázek a odpovědí nebo trénink ryze českých předtrénovaných modelů, jako je ALBERT. Zajímavá by byla také práce s datasetem COSTRA, který se zaměřuje na zachycení významu vět v různých tvarech, což by vedlo k lepšímu pochopení parafrázované otázky. Přínosné by rozhodně bylo rozšíření existujících a tvorba nových datasetů pro češtinu – ať už pro trénink kvalitních komponent reader a retriever nebo směrodatnější vyhodnocení vytvářených systémů nad otevřenou doménou.

Literatura

[1] Alammar, J. *The Illustrated Transformer* [online]. 2018. [cit. 17-03-2021]. Dostupné z: http://jalammar.github.io/illustrated-transformer/.

[2] Arkhipov, M., Trofimova, M., Kuratov, Y. a Sorokin, A. Tuning Multilingual Transformers for Language-Specific Named Entity Recognition. In: *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*. Florence, Italy: Association for Computational Linguistics, Srpen 2019, s. 89–93. DOI: 10.18653/v1/W19-3712.

[3] Bahdanau, D., Cho, K. a Bengio, Y. Neural machine translation by jointly learning to align and translate. *ArXiv preprint arXiv:1409.0473*. 2014.

[4] Barancikova, P. a Bojar, O. COSTRA 1.0: A Dataset of Complex Sentence Transformations. In: Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C. et al., ed. *Proceedings of the 12th International Conference on Language Resources and Evaluation*. Paris, France: European Language Resources Association, 2020, s. 3535–3541. ISBN 979-10-95546-34-4.

[5] Chen, D., Fisch, A., Weston, J. a Bordes, A. Reading wikipedia to answer open-domain questions. *ArXiv preprint arXiv:1704.00051*. 2017.

[6] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G. et al. Unsupervised cross-lingual representation learning at scale. *ArXiv preprint arXiv:1911.02116*. 2019.

[7] Craswell, N. Mean Reciprocal Rank. In: LIU, L. a ÖZSU, M. T., ed. *Encyclopedia of Database Systems*. Boston, MA: Springer US, 2009, s. 1703–1703. DOI: 10.1007/978-0-387-39940-9_488. ISBN 978-0-387-39940-9.

[8] Devlin, J., Chang, M.-W., Lee, K. a Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv preprint arXiv:1810.04805*. 2018.

[9] Fajcik, M., Docekal, M., Ondrej, K. a Smrz, P. Pruning the Index Contents for Memory Efficient Open-Domain QA. *ArXiv preprint arXiv:2102.10697*. 2021.

[10] Hochreiter, S. a Schmidhuber, J. Long short-term memory. *Neural computation*. 1. vyd. MIT Press. 1997, sv. 9, č. 8, s. 1735–1780.

[11] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L. et al. Dense passage retrieval for open-domain question answering. *ArXiv preprint arXiv:2004.04906*. 2020.

[12] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. et al. Albert: A lite bert for self-supervised learning of language representations. *ArXiv preprint arXiv:1909.11942*. 2019.

[13] Lv, Y. a Zhai, C. Lower-Bounding Term Frequency Normalization. In: Macdonald, C., Ounis, I. a Ruthven, I., ed. *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*. New York, NY, USA: Association for Computing Machinery, 2011, s. 7–16. CIKM '11. DOI: 10.1145/2063576.2063584. ISBN 9781450307178.

[14] Lv, Y. a Zhai, C. When Documents Are Very Long, BM25 Fails! In: Ma, W., Nie, J., Baeza-Yates, R., Chua, T. a Croft, W. B., ed. *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*. New York, NY, USA: Association for Computing Machinery, 2011, s. 1103–1104. SIGIR '11. DOI: 10.1145/2009916.2010070. ISBN 9781450307574.

[15] Macková, K. a Straka, M. *Czech Translation of SQuAD 2.0 and 1.1* [online]. 2020. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. [cit. 29-03-2021]. Dostupné z: http://hdl.handle.net/11234/1-3249.

[16] Macková, K. a Straka, M. Reading Comprehension in Czech via Machine Translation and Cross-lingual Transfer. *ArXiv preprint arXiv:2007.01667*. 2020.

[17] Manning, C. a Hewitt, J. *Natural Language Processing with Deep Learning course* [online]. 2020. Prezentace 1-4. [cit. 13-03-2021]. Dostupné z: http://web.stanford.edu/class/cs224n/.

[18] Manning, C. a Nayak, P. *Information Retrieval and Web Search course* [online]. 2020. Prezentace 2-3, 6-9, 12-13. [cit. 13-03-2021]. Dostupné z: https://web.stanford.edu/class/cs276/.

[19] Manning, C. D., Raghavan, P. a Schütze, H. *Introduction to information retrieval*. 1. vyd. Cambridge University Press, 2008. ISBN 978-0-521-86571-5.

[20] Medved, M. a Horák, A. AQA: Automatic Question Answering System for Czech. In: Sojka, P., Horák, A., Kopeček, I. a Pala, K., ed. *Text, Speech, and Dialogue*. Cham: Springer International Publishing, 2016, s. 270–278. ISBN 978-3-319-45510-5.

[21] Medved, M. a Horák, A. Sentence and Word Embedding Employed in Open Question-Answering. In: Rocha, A. P. a Herik, H. J. van den, ed. *Proceedings of the 10th International Conference on Agents and Artificial Intelligence, ICAART 2018, Volume 2, Funchal, Madeira, Portugal, January 16-18, 2018*. SciTePress, 2018, s. 486–492. DOI: 10.5220/0006595904860492. ISBN 978-989-758-275-2.

[22] Medveď, M. a Horák, A. *Sqad 3.0* [online]. 2019. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. [cit. 25-03-2021]. Dostupné z: http://hdl.handle.net/11234/1-3069.

[23] Mikolov, T., Chen, K., Corrado, G. a Dean, J. Efficient estimation of word representations in vector space. *ArXiv preprint arXiv:1301.3781*. 2013.

[24] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. a Dean, J. Distributed representations of words and phrases and their compositionality. *ArXiv preprint arXiv:1310.4546*. 2013.

[25] Min, S., Boyd Graber, J., Alberti, C., Chen, D., Choi, E. et al. NeurIPS 2020 EfficientQA Competition: Systems, Analyses and Lessons Learned. *ArXiv preprint arXiv:2101.00133*. 2021.

[26] Olah, C. *Understanding LSTM Networks* [online]. 2015. [cit. 20-03-2021]. Dostupné z: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[27] Pennington, J., Socher, R. a Manning, C. D. Glove: Global Vectors for Word Representation. In: Moschitti, A., Pang, B. a Daelemans, W., ed. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2014, s. 1532–1543. DOI: 10.3115/v1/d14-1162. ISBN 9781643680491.

[28] Rajpurkar, P., Jia, R. a Liang, P. Know what you don't know: Unanswerable questions for SQuAD. *ArXiv preprint arXiv:1806.03822*. 2018.

[29] Rajpurkar, P., Zhang, J., Lopyrev, K. a Liang, P. Squad: 100,000+ questions for machine comprehension of text. *ArXiv preprint arXiv:1606.05250*. 2016.

[30] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. a Gatford, M. Okapi at TREC-3. In: Harman, D. K., ed. *Proceedings of The Third Text Retrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*. National Institute of Standards and Technology (NIST), 1994, 500-225, s. 109–126. NIST Special Publication.

[31] Ruder, S. *On word embeddings - Part 1* [online]. 2016. [cit. 15-03-2021]. Dostupné z: https://ruder.io/word-embeddings-1/.

[32] Sabol, R., Medveď, M. a Horák, A. Czech Question Answering with Extended SQAD v3.0 Benchmark Dataset. In: Horák, A., Rychlý, P. a Rambousek, A., ed. *Proceedings of the Thirteenth Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2019*. Brno: Tribun EU, 2019, s. 99–108. ISBN 978-80-263-1530-8.

[33] Sharma, A. *Understanding Activation Functions in Neural Networks* [online]. 2017. [cit. 15-03-2021]. Dostupné z: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.

[34] Straková, J., Straka, M. a Hajič, J. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition. In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, s. 13–18.

[35] Trotman, A., Puurula, A. a Burgess, B. Improvements to BM25 and Language Models Examined. In: Culpepper, J. S., Park, L. A. F. a Zuccon, G., ed. *Proceedings of the 2014 Australasian Document Computing Symposium, ADCS 2014, Melbourne, VIC, Australia, November 27-28, 2014*. ACM, 2014, s. 58. DOI: 10.1145/2682862.2682863. ISBN 9781450330008.

[36] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is all you need. *ArXiv preprint arXiv:1706.03762*. 2017.

[37] Wikipedia contributors. *Frekvenční seznam (čeština)* [online]. 2020. [cit. 28-03-2021]. Dostupné z: https://cs.wiktionary.org/wiki/P%C5%99%C3%ADloha:Frekven%C4%8Dn%C3%AD_seznam_(%C4%8De%C5%A1tina).

[38] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M. et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv preprint arXiv:1609.08144*. 2016, abs/1609.08144.

[39] Yiu, T. *Understanding Neural Networks* [online]. 2019. [cit. 19-03-2021]. Dostupné z: https://towardsdatascience.com/understanding-neural-networks-19020b758230.

[40] ZELINA, P. *Pretraining and Evaluation of Czech ALBERT Language Model*. 2020. Bakalářská práce. Masarykova univerzita, Fakulta informatiky, Brno.

[41] Zhu, M., Ahuja, A., Juan, D., Wei, W. a Reddy, C. K. Question Answering with Long Multiple-Span Answers. In: Cohn, T., He, Y. a Liu, Y., ed. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*. Association for Computational Linguistics, 2020, s. 3840–3849. DOI: 10.18653/v1/2020.findings-emnlp.342.

[42] Šmerk, P. a Rychlý, P. *Majka – rychlý morfologický analyzátor* [online]. Masarykova univerzita, 2009. [cit. 22-03-2021]. Dostupné z: http://nlp.fi.muni.cz/ma/.

Příloha A

Obsah přiloženého paměťového média

```
/
├── benchmarks/ ..... odpovědi systému pro vyhodnocení
│   └── všechny odpovědi systému z experimentů ve formátu json
├── data/ ..... data potřebná pro experimenty
│   └── modely, zpracovaná data, ...
├── notebooks/
│   ├── question_answering.ipynb ..... experimenty se systémem a zpracování dat
│   ├── mbert_czech_squad_fine-tuning.ipynb ..... trénink mBERT modelů
│   ├── albert_squad_fine-tuning.ipynb ..... trénink ALBERT modelů
│   ├── czech_squad.py
│   └── czech_squad2.py ..... skripty pro knihovnu datasets
├── source_tex/ ..... text práce v jeho zdrojové podobě
│   └── zdrojové soubory latexu pro překlad
├── web_server/ ..... server pro demonstrační aplikaci
│   ├── server.py
│   ├── requirements.txt ..... potřebné Python knihovny
│   ├── data/ ..... potřebná data pro spuštění serveru
│   └── app/
│       └── zdrojové soubory demonstrační aplikace
├── poster.pdf ..... plakát prezentující výsledky práce
├── README.md
├── thesis.pdf ..... text práce ve formátu PDF
└── thesis_print.pdf ..... text práce pro tisk
```