



**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 PŘÍROZENÉM JAZYCE**

MACHINE LEARNING FOR NATURAL LANGUAGE QUESTION ANSWERING

**BAKALÁŘSKÁ PRÁCE**

BACHELOR'S THESIS

**AUTOR PRÁCE**

AUTHOR

**JONÁŠ SASÍN**

**VEDOUcí PRÁCE**

SUPERVISOR

**Doc. RNDr. PAVEL SMRŽ, Ph.D.**

**BRNO 2021**

## Zadání bakalářské práce



23912

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 s pomocí strojového učení a využitím české Wikipedie jako báze znalostí. Popsány jsou dostupné metody, state of the art a výběr vhodných metod pro daný úkol. Prozkoumány jsou dostupné datové sady pro trénink i vyhodnocení výsledného řešení. Poté je představen návrh výsledného systému a popis implementace jeho jednotlivých komponent. Diskutovány jsou metody vyhledávání relevantních dokumentů a vhodné předzpracování prohledávaných textů i nalezení odpovědi v získaném kontextu. Celý systém je vyhodnocen na známém datasetu s použitím základních metrik. Výsledkem je systém schopný odpovídat na otázky v přirozeném jazyce.

## Abstract

This thesis describes natural language question answering using machine learning and searching Czech Wikipedia as its knowledge base. Existing methods, state of the art and choosing a suitable method for the given task is described. Available datasets for training and final system evaluation are explored. The system design is then introduced and the implementation of its individual components is described. Methods for relevant document retrieval, preprocessing the searched corpus as well as finding the answer span in the acquired context are discussed. The whole system is then evaluated on a well known dataset using the basic metrics. The result is a system being able to answer natural language questions.

## Klíčová slova

zpracování přirozeného jazyka, NLP, odpovídání na otázky, strojové učení, Wikipedia, otevřená doména, ALBERT, dolování znalostí

## Keywords

natural language processing, NLP, question answering, machine learning, Wikipedia, open domain, ALBERT, knowledge mining

## 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. Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....

Jonáš Sasín  
30. března 2021

## **Poděkování**

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.).

# Obsah

<b>1</b>	<b>Úvod</b>	<b>2</b>
<b>2</b>	<b>Reprezentace slov a zpracování textu</b>	<b>3</b>
2.1	Reprezentace slov . . . . .	3
2.2	Tokenizace a předzpracování . . . . .	5
<b>3</b>	<b>Strojové učení pro extrakci odpovědi</b>	<b>8</b>
3.1	Neuronové sítě . . . . .	8
3.2	Nejlepší architektury pro porozumění textu . . . . .	11
3.3	Předtrénované modely BERT a ALBERT . . . . .	14
<b>4</b>	<b>Řazení dokumentů dle relevance</b>	<b>17</b>
4.1	Indexování dokumentů . . . . .	17
4.2	Algoritmy pro řazení dokumentů . . . . .	18
<b>5</b>	<b>Dostupné datové sady a výběr</b>	<b>20</b>
5.1	Datové sady pro angličtinu . . . . .	21
5.2	Datové sady pro češtinu . . . . .	21
5.3	Výběr trénovacích a testovacích dat . . . . .	21
<b>6</b>	<b>Návrh systému a implementace jednotlivých komponent</b>	<b>22</b>
6.1	Zvolený přístup k problému . . . . .	22
6.2	Návrh jednotlivých částí systému . . . . .	23
6.3	Použité nástroje a technologie pro implementaci . . . . .	24
6.4	Implementace a trénink readeru . . . . .	24
6.5	Implementace retrieveru . . . . .	24
<b>7</b>	<b>Vyhodnocení systému a rozbor chyb</b>	<b>25</b>
7.1	Vysvětlení základních metrik . . . . .	25
7.2	Postup při vyhodnocování výsledného systému . . . . .	25
7.3	Porovnání výsledků se současným stavem poznání . . . . .	25
7.4	Rozbor chyb a možnosti dalšího vývoje . . . . .	25
<b>8</b>	<b>Závěr</b>	<b>26</b>
	<b>Literatura</b>	<b>27</b>

# Kapitola 1

## Úvod

Odpovídání na otázky je dobře prozkoumaný a populární úkol z oblasti zpracování přirozeného jazyka. Využití kolem nás, v běžném životě, můžeme vidět třeba u vyhledávacích nástrojů, komunikačních agentů nebo hlasových asistentů.

Výzkum se v dané oblasti soustředí na metody strojového učení pro vyznačení odpovědi na otázku v daném textu. Pro trénink neuronové sítě, která by byla schopná spolehlivě vykonávat takový úkol, je zapotřebí velké množství anotovaných dat.

Kromě porozumění textu jsou v oblasti důležité také algoritmy pro zhodnocení relevantnosti dokumentů používané všemi vyhledávacími nástroji a metody zpracování textu, které jsou užitečné pro optimální funkci vyhledávacích algoritmů.

Pro češtinu je množství i velikost dostupných datových sad oproti angličtině malé a je tedy obtížné dosáhnout s ní podobných výsledků, jako u jazyků s mnoha zdroji (angličtina, němčina, francouzština ...). Obzvlášť v kontextu toho, že pro většinu úkolů na poli zpracování přirozeného jazyka se v posledních letech nejlépe osvědčily velké modely předtrénované na velmi rozsáhlých korpusech textu. Ty pro češtinu prozatím bohužel nejsou dostupné v dostatečné kvalitě.

V této práci se snažím otestovat přístup, který využívá strojový překlad češtiny do angličtiny, aby mohlo být použito jednoho z robustních předtrénovaných modelů pro porozumění textu. Kromě metod pro porozumění textu a extrakci odpovědi v práci rozebírám algoritmy řazení dokumentů dle jejich relevance k otázce. Jako báze znalostí pro systém schopný na otázky odpovídat je použita česká Wikipedie, jejíž články jsou pro nalezení vhodné odpovědi prohledávány.

Začal jsem popsáním metod pro zpracování textu v kapitole 2. Popsány jsou zde metody reprezentace slov a způsoby zpracování textu vhodné pro porozumění textu a pro algoritmy řazení relevantních dokumentů.

Kapitola 3 se zabývá popisem neuronových sítí pro porozumění textu a extrakci odpovědi. Popsány jsou zde moderní architektury a modely použité v práci. V kapitole 5 jsou poté popsány datové sady vhodné pro trénink a vyhodnocení výsledného systému. Následuje kapitola 6, kde je popsán návrh a implementace systému založeného na zmíněných postupech a kapitola 7, kde je výsledný systém vyhodnocen s použitím standardních metrik.

## Kapitola 2

# Reprezentace slov a zpracování textu

Předtím, než se vrhneme na vysvětlení poměrně složitých jazykových modelů a vyhledávacích algoritmů, bude v této kapitole vysvětleno pár základních pojmů a technik.

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

Také je potřeba prozkoumat možnosti 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 pracovat neuronové sítě.

Pro takovou reprezentaci jsou nejpoužívanější vektory pevně dané dimenze známé jako „word embeddings“. Pro představu například (běžně) 300 desetinných čísel pro popis každého slova v textu. Vektory slov ve slovníku nesou důležitou informaci o sémantické příbuznosti jednotlivých slov. Po promítnutí do spojitého prostoru jsou si vektory, které reprezentují sémanticky podobná slova, blízko, jak zobrazuje obrázek 2.1.

Po aplikaci těchto myšlenek zmíněných v článcích [13] a [14] byl zaznamenán výrazný pokrok v úlohách z oblasti zpracování přirozeného jazyka, jako strojový překlad, porozumění textu, odhad emocí, 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.

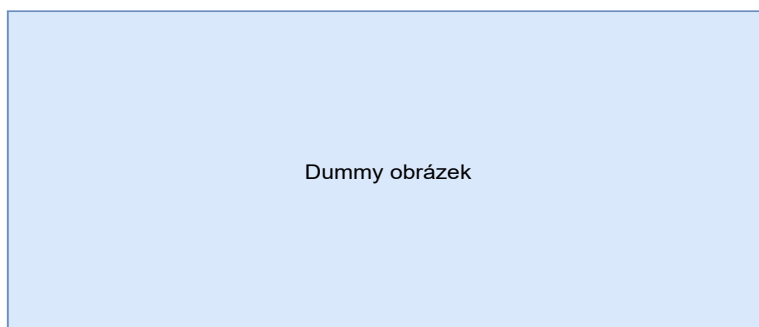
Výsledné vektory nám poté umožňují provádět se slovy také velmi zajímavé matematické operace, jako je sčítat, nebo odčítat a je schopen zachytit velmi zajímavé vztahy jako například:

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

nebo

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

V následující části budou popsány vlastnosti a nejznámějších metod pro získání vektorových reprezentací.



Obrázek 2.1: Dummy picture - znazorneni vektoroveho prostoru

### 2.1.1 Word2vec

Word2vec je technika prezentována v článku [13] 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 sousedních slov v tzv. kontextovém okně, které se postupně posouvá. Dostáváme tedy pro každé slovo pravděpodobnost, se kterou se bude vyskytovat v blízkosti slova cílového.

První model používaný pro trénink word2vec vektorů se nazývá Continuous Bag of Words, neboli CBOW. Neuronová síť se trénuje hádáním chybějícího slova v kontextu známých sousedním slov kontextového okna.

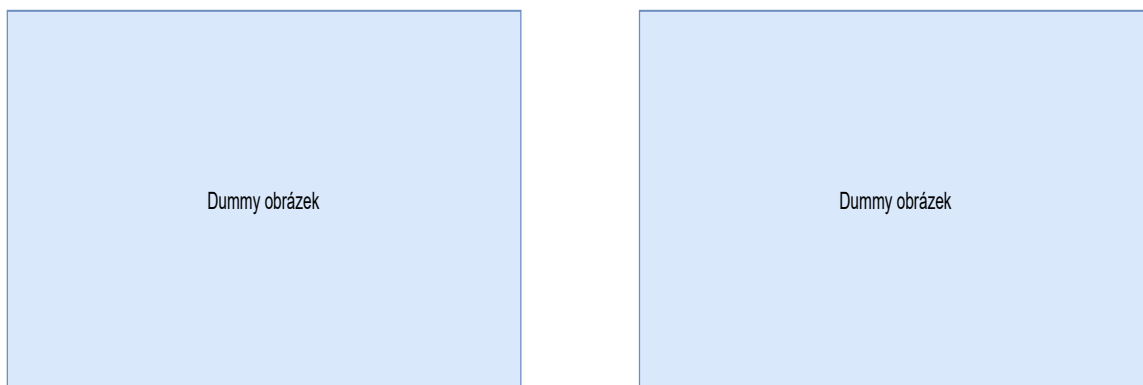
Druhá metoda funguje na opačném principu a nazývá se Continuous Skip-gram Model. Na základě současného slova se tedy model snaží předpovědět slova v určitém vzdálenosti před, i po současném slově, v závislosti na velikosti kontextového okna. Tato metoda se z článku [13] jeví jako o něco úspěšnější a je dále zdokonalena v článku [14].

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

### 2.1.2 GloVe

GloVe, neboli Global Vectors for Word Representation je technika prezentována v článku [15]. Výsledkem je vektorový prostor o pevně dané dimenzi stejně, jako u word2vec reprezentace. Liší se však způsobem získání daných reprezentací.





Obrázek 2.2: Obrázek cbow a skipgram

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 nějakou matici společných výskytů slov v korpusu.

Jeho hlavní výhodou 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, ale základní idea není konceptu word2vec příliš daleko.

### 2.1.3 Wordpiece embeddings

Wordpiece je prediktivní technika prezentována článkem [27] v roce 2016 (původně představeno pro strojový překlad) a používána modely popsány v sekci 3.3.

Hlavním záměrem přístupu je lépe se vypořádat s ojedinělými slovy tak, že jsou slova rozdělena do již známých *podslav*. 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 jazykový model je na tomto slovníku trénován. 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

Zpracování textu je důležitou součástí oblasti zpracování přirozeného jazyka. S textem se totiž pracuje lépe, pokud jsme schopni odstranit některé záludnosti a nadbytečnosti přirozeného jazyka a dostat ho do podoby, který je pro automatické zpracování vhodnější. Jednotlivé nástroje pro efektivní zpracování textu v práci jsou poté uvedeny v části 6.3.

První část této sekce navazuje na část 2.1 o reprezentaci jednotlivých slov textu a zabývá se postupem rozdělení textu na tzv. *tokeny*, které mohou být později převedeny na jejich vektorovou reprezentaci.

Druhá část je věnována postupům pro zpracování a pokud možno normalizaci textu pro optimální využití vyhledávacích algoritmů. Statistické metody využívány těmito algoritmy jsou efektivnější, pokud jsou z textu odstraněny například slova nesoucí zanedbatelnou informaci. Mohou být také zmateny různými tvary daných slov.

### 2.2.1 Tokenizace vstupního textu

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

\_Adam + \_šel + \_po + \_škole + \_domů + \_.

Primitivní přístup je rozdělení textu pomocí bílých znaků a interpunkčních znamének, který ale nebere ohled např. na tečky uprostřed zkratk v angličtině a těžko se vypořádává se složeninami. Získané tokeny je pak taky třeba normalizovat pro následný převod na word embeddings.

V přístupu uvedeném v sekci 2.1.3 jsou slova dělena na tokeny podle toho, která podslova v daném textu jsou slovníku známa. Můžeme si uvést příklad z angličtiny, s jímž podobným se ještě jistě setkáme i v sekci 3.3. Věta „I like playing.“ by tedy byla převedena na následující tokeny:

\_I + \_like + \_play + \_ing + \_.

Na uvedeném případě je vidět hlavně převedení posloupnosti znaků „playing“ na dvě podslova (tokeny) *play* a *ing*.

### 2.2.2 Normalizace délky textu a rozdělení do vět

V kontextu práce je normalizací délky textu myšlena jakási standardní maximální délka dokumentu, který je později předložen readeru k vyznačení správné odpovědi. Tento přístup je užitečný ze několika důvodů.

Algoritmy pro řazení dokumentů dle relevance sice berou na relativní délku každého textu ohled, nicméně některé články jako [10] nasvědčují, že mohou být příliš dlouhé dokumenty znevýhodněny. Článek [10] sice představuje úpravu klasického algoritmu pro minimalizaci tohoto problému, v práci jsou ale 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<sup>1</sup> a jak je naznačeno v článku [28], není na dlouhých dokumentech příliš úspěšný.

Dalším důvodem je snaha nepřekládat pro účel vyznačení správné odpovědi příliš dlouhé důvody textu.<sup>2</sup>

**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 bude 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 v půlce.

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.

<sup>1</sup>Pro každou otázku je jen jeden odstavec kontextu, ne celý článek z Wikipedie.

<sup>2</sup>Hlavně kvůli určitým omezením dostupných nástrojů pro překlad.

### 2.2.3 Převod na malá písmena

Převod na malá písmena je sice velmi jednoduchý, ale přesto užitečný úkol. Při psaní otázky na klávesnici je velmi pravděpodobně, že pomyslný uživatel napíše „sssr“ místo „SSSR“ nebo „Albert“ místo „ALBERT“, ale bylo by vhodné, 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 písmena 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* většinou souborem nejčastěji používaných slov v daném jazyce [26]. Většinou jsou odstraňována, protože kvůli jejich vysokému výskytu nesou menší informační hodnotu a jejich odstranění vede k lepší funkci vyhledávacích algoritmů [21].

Nejužívanější slova daného jazyka je možno například ve frekvenčních seznamech jako [24], který obsahuje 100 nejpoužívanějších slov v češtině. Získaný seznam je vhodné kriticky zhodnotit a vyčlenit z něj některá slova, která by pro vyhledávání mohla mít kritický význam.<sup>3</sup> Většinou je zanedbatelná alespoň většina 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*.<sup>4</sup>

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 [25].

Využití lemmatizace je různé, ale v případě této práce je podobné, jako účel postupu uvedeného v sekci 2.2.3. Jednoduše se hodí, když pro dotaz obsahující „... Liparských ostrovů“ budou nalezeny i dokumenty obsahující základní tvar hledaného výrazu „Liparské ostrovy“

Lemmatizace je také užitečným nástrojem při odstranění stop slov, kdy můžeme například odstranit všechny tvary slovesa „být“, zatímco seznam stop slov může obsahovat pouze jeho základní tvar.

<sup>3</sup>Příklad: „první“, „země“, „člověk“ ...

<sup>4</sup>Podobným postupem je *stematizace*, která ale algoritmicky pouze odstraní koncovky a předpony pro nalezení slovního kmene. Lemmatizátor pro nalezení základních tvarů používá morfologický slovník.

## Kapitola 3

# Strojové učení pro extrakci odpovědi

Rozmach strojového učení umožnil obrovský pokrok v oblastech NLP jako strojový překlad, rozpoznání entit, generování textu nebo také odpovídání na otázky. Proto na tomto přístupu staví všechny nejmodernější metody a tímto směrem se také ubírá 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 budou ve zkratce vysvětleny principy fungování a učení neuronových sítí. Poté budou rozebrány nejznámější architektury neuronových sítí používané pro vypořádání se s úkoly na poli zpracování přirozeného jazyka a nakonec budou popsány nejmodernější modely, které na nich staví.

### 3.1 Neuronové sítě

V této části bude vysvětlen princip fungování neuronových sítí. Informace v sekci byly převzaty z článku [20] a z prezentací Stanfordského kurzu NLP.<sup>1</sup>

Neuronové sítě 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.

Fungují tu tři základní jednoduché vrstvy (3.1). Vstupní, skrytá (zde probíhá proces přetváření vstupů) a výstupní, na které se přetvořený vstup objeví. Skrytá vrstva se navíc běžně skládá z mnoha dalších vrstev.

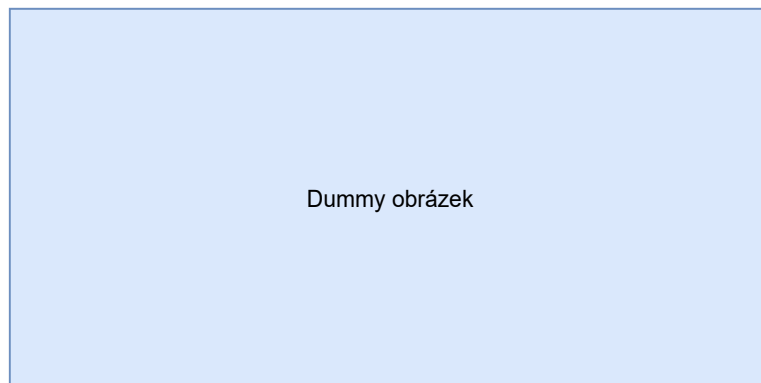
Každá z těchto vrstev, ať už je vstupní, výstupní, nebo skrytá, se skládá z dalších primitiv a ty jsou nazývány jak jinak, než neurony. Každý neuron v každé vrstvě je propojen s každým neuronem vrstvy další, a tak je propojena celá síť.

#### 3.1.1 Umělý neuron a aktivační funkce

Umělý neuron je primitivní jednotka inspirována neuronem skutečným. Ten vypadá asi tak, že má tělo, do kterého přichází krátkými výběžky, tzv. *dendrity* elektrické signály z ostatním neuronů. Z těla tohoto neuronu pak vychází jediný dlouhý výběžek, tzv. *axon*.

---

<sup>1</sup>Dostupné z: <http://web.stanford.edu/class/cs224n/>



Obrázek 3.1: Dummy picture - znázornění neuronové sítě se třemi vrstvami a váhami.

Neuron se tedy na základě signálů, které do něj přichází dendrity rozhodne (na základě nějaké minimální hranice přijatého signálu), zda-li bude, nebo nebude vysílat signál dál [23].

Funkce umělého neuronu je analogická. Jak už bylo popsáno v úvodu sekce 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 každým neuronem má svoji váhu  $w$ , která ovlivňuje důležitost jednotlivého vstupu do neuronu. Každý neuron má pak svůj práh  $B$  a aktivační funkci. Výstupem neuronu je tedy vážená suma vstupů do neuronu s přičteným prahem a aplikovanou aktivační funkcí, neboli

$$F\left(\sum_{i=1}^n(x_i w_i) + B\right)$$

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

Všechny propojení jednotlivých neuronů a jejich prahy v 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 tak 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 (čtyři) základní varianty aktivačních funkcí:

- (Skoková)
- Sigmoidální
- Hyperbolické tangenty
- ReLU (Rectified Linear Unit)

Funkce jsou znázorněny na obrázku 3.2. Vhodnost použití každé z nich může být různá. 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 méně výpočetně náročné [1].



Obrázek 3.2: Obrázek sigmoid, tanh a relu funkce

### 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. Tímto postupem se postupně hodnoty vstupní vrstvy přetvoří na hodnoty ve vrstvě výstupní a může se určit, jak byla síť 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. Nejznámější nákladovou funkcí 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. Celý postup zpětné propagace je velmi složitý a pro intuitivní pochopení problematiky není klíčový, proto tu nebude podrobně rozebrán.

Celý postup zpětné propagace je opakován a parametry neuronové sítě jsou postupně optimalizovány.

Trénink neuronové sítě tedy probíhá následovně. Síť s náhodně inicializovanými parametry je předložena část dat z trénovací datové sady. Ty jsou pomocí dopředné propagace postupem přes skrytou vrstvu přetvořeny na výstupy a tím získány predikce neuronové sítě. Predikce jsou porovnány se základní pravdou (tzv. *ground truth*) a nákladovou funkcí je vyjádřeno, jak byly predikce daleko od pravdy. 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 *learning rate*<sup>2</sup>. Pomocť může také množství trénovacích dat nebo počet trénovacích epoch.

<sup>2</sup>Konstanta určující velikost kroku při každé iteraci zpětné propagace při úpravě parametrů neuronové sítě.

## 3.2 Nejlepší architektury pro porozumění textu

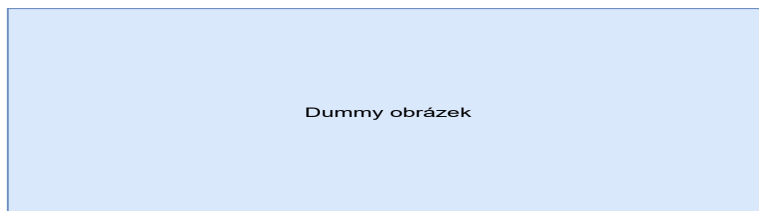
V této sekci budou popsány dvě nejznámější a nejpoužívanější architektury používané pro zpracování přirozeného jazyka. Rozšiřují koncepty prezentované v sekci 3.1 a snaží se překonat problémy, které tradiční model představuje.

První model v části 3.2.1 využívá k překonání ztráty kontextu rekurentní neuronové sítě. Druhý model v části 3.2.2 spoléhá na paralelní zpracování a vzájemnou pozornost mezi otázkou a kontextem.

### 3.2.1 Long short-term memory

Informace v této sekci byly převzaty z originální práce [6], která architekturu představuje a ze článku [4].

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 bychom byli rádi, aby pochopení současné věty dokázala ovlivnit i věta předcházející.

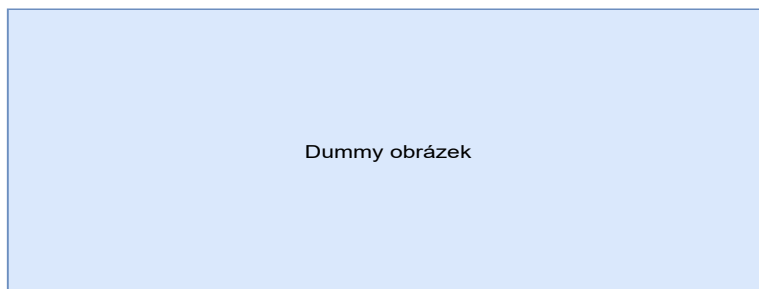


Obrázek 3.3: dummy obrázek - schéma RNN

Nejprve bude vysvětlen princip **rekurentní neuronové sítě** (dále RNN - Recurrent Neural Network). Jejich hlavním úkolem 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 měly ovlivnit slovo současné. Stejně tak by měly 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  $h_n$  je výstup předchozí sítě  $h_{n-1}$  a slovo  $x_n$ . Výstup  $y_n$  je potom vstupem do další jednotky  $h_{n+1}$ .

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



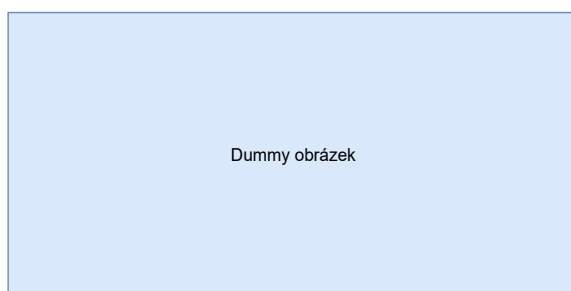
Obrázek 3.4: dummy obrázek - schéma obousměrné RNN

**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 si souvislost propojit. Pro příklad, kdy je kontext „dále v minulosti“ (př.: „Učím se od 6 let anglicky ... Vzhledem k těmto okolnostem plynule ovládám angličtinu“) není RNN schopná naučit se tuhle souvislost mezi kontextem a hledaným slovem.

LSTM je tedy narozdíl od klasické RNN navržena 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 jejím hlavním komponentem jsou tzv. *brány*.



Obrázek 3.5: dummy obrázek - schéma lstm buňky

Brány jsou v LSTM buňce celkem tři. Každá je složena z vrstvy neuronové sítě se sigmoidální funkcí a násobícího operátoru. Brána je komponentem určujícím, kolik informace je možno propustit. Brány budou popsány zleva doprava.

První, tzv. zapomínací brána určuje, kolik vstupní informace je zahazeno<sup>3</sup>. Druhá, tzv. vstupní brána, určí, které hodnoty skrytého stavu buňky budou aktualizovány a vrstva s funkcí *tanh* rozhodne o nových kandidátních hodnotách, které po kombinaci s výstupem sigmoidální vrstvy vytvoří nový skrytý stav buňky. Poslední bránou je výstupní, která spojuje informaci z brány zapomínací a 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 s tou druhou propojenou v opačném směru získáme obousměrnou LSTM síť. Výstupy obou sítí jsou poté pro získání výstupu zřetězeny.

### 3.2.2 Transformers

Informace v této sekci byly převzaty z originální práce *Attention Is All You Need* [22], práce [2] a článku [7]. Sekce popisuje Transformer model používaný pro zpracování přirozeného jazyka a mechanismus pozornosti, který s ním úzce souvisí.

Přístup použitý tímto modelem se dokázal oprostit od rekurentních modelů jako LSTM a nahradil je na poli zpracování přirozeného jazyka jako nová state of the art architektura. Transformers nepoužívá sekvenční zpracování slov, ale zpracování paralelní.

<sup>3</sup>Výstupem je číslo v rozmezí (0,1), kde 0 znamená „všechno zahodit“ a 1 znamená „všechno propustit“.



Jeho nejdůležitější myšlenkou je tzv. **mechanismus pozornosti** (anglicky *attention mechanism*) představený článkem [2] původně pro strojový překlad. Mechanismus pozornosti se však dále rozvíjel a popularizoval do dalších odvětví strojového učení. Stručně bude tedy popsán.

Mechanismus vezme vektory dvou vět přirozeného jazyka naráz a sestaví z nich  $key(\mathbf{K})$ - $value(\mathbf{V})$  a  $query(\mathbf{Q})$  matice (řádek – vektor pro každé vstupní slovo). Ty vzniknou vynásobením vstupní word embedding matice speciální váhovou maticí, jejíž parametry se natrénují. Na matice jsou poté aplikovány následující operace pro získání výstupu attention mechanismu

$$\text{softmax}\left(\frac{\mathbf{Q} \times \mathbf{K}}{k}\right) \times \mathbf{V} \quad (3.1)$$

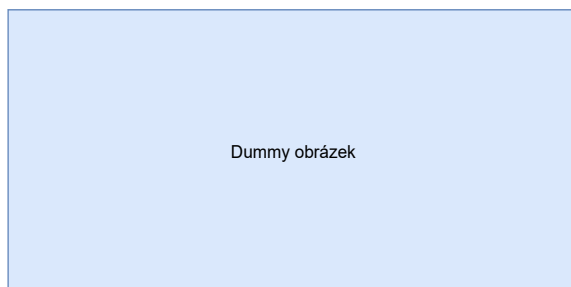
kde  $k$  je podle [22] konstanta 8 (mohly by být stanoveny i jiné hodnoty) a softmax funkce pro normalizaci výstupu do rozmezí (0,1).

Vektory pro všechny vstupní slova respektive vstupní matice  $\mathbf{K}$ ,  $\mathbf{V}$  a  $\mathbf{Q}$  mohou být ze stejné věty. Potom se používá termín **self-attention**.

Tím je určeno, která část vstupu je relevantní (self-attention), nebo například která část dokumentu je relevantním kontextem dané otázky.

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

**Transformer** enkodér<sup>45</sup> je model, jehož architektura je kombinací dopředné neuronové sítě a mechanismu pozornosti. Jak už bylo popsáno, jeho hlavní výhodou je vlastnost *mechanismu pozornosti* zpracovávat vstupní slova paralelně, a tím se naučit i vztahy mezi slovy na dlouhé vzdálenosti v textu.



Obrázek 3.6: dummy obrázek - schéma transformer enkodéru

Na obrázku 3.6 můžeme vidět zjednodušenou strukturu jednotky enkodéru. V článku [22] tvoří enkodér 6 takových bloků propojených za sebe.

Před vstupem do prvního bloku jsou jednotlivé tokeny vstupního textu převedeny na vektory (word embeddings), ke kterým je přidáno poziční kódování<sup>6</sup>.

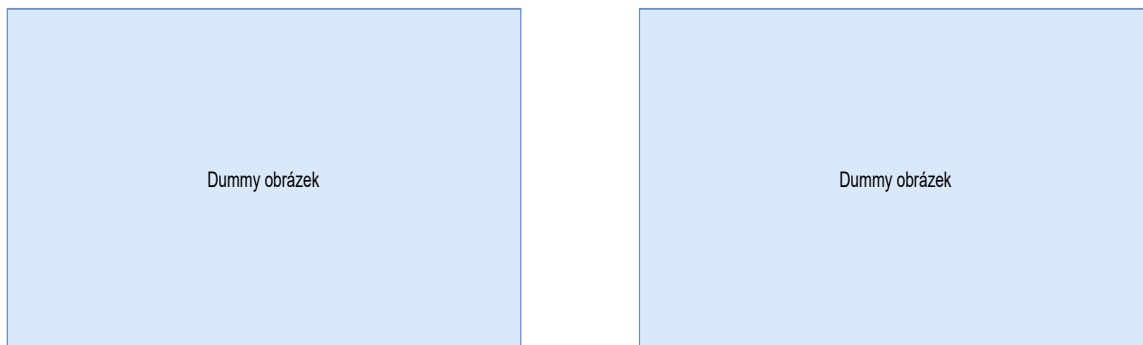
Každý blok se pak skládá z „multi-head attention“ modulu a plně propojené dopředné vrstvy. Obě části jsou následovány normalizační vrstvou.

<sup>4</sup>Enkodér je část modelu, která vytvoří z vektorů jednotlivých slov na vstupu vektor celého vstupního textu, reprezentující jeho význam.

<sup>5</sup>V původním článku představen model pro *sequence to sequence* model vhodný pro překlad, který obsahuje i dekodér pro generování textu.

<sup>6</sup>identifikuje pořadí tokenů ve vstupním textu.

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



Obrázek 3.7: Obrázek multihead a scaled dot-product attention

Mechanismus na obrázku 3.7 vlevo je popsán rovnicí (3.1) a je hlavním komponentem multi-head attention modulu znázorněném v 3.7 vpravo.

Multi-head znamená, že do modulu vstupuje 8 (dle práce [22]) trojic  $V, K, Q$ . Každá matice trojice  $V, K, Q$  je transformována lineární vrstvou a na celou trojici je potom aplikován self dot-product attention (self-attention) mechanismus. Tento postup je aplikován na každou z osmi trojic paralelně, pokaždé však s jinými váhami lineární vrstvy.

### 3.3 Předtrénované modely BERT a ALBERT

V této sekci budou představeny modely založené na architektuře Transformer popsané v sekci 3.2.2. Informace v této sekci jsou převzaty z původních článků [5] a [8].

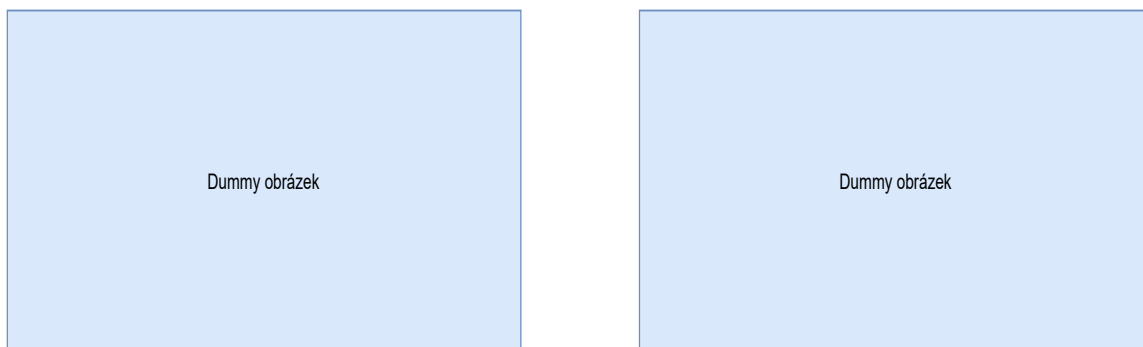
Předtrénované modely stavějící na této architektuře určily nový standard pro přicházející state of the art metody. Kromě jejich vysoké úspěšnosti je obrovskou výhodou také znovupoužitelnost předtrénovaných modelů. Modely je možno jednoduše dotrénovat (provést tzv. *fine-tuning* po načtení modelu s předtrénovanými 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 [5]. Jeho hlavním přínosem je aplikace obousměrného tréninku Transformer architektury. Ukazuje, že model trénovaný v obou směrech dokáže plně využít schopnost Transformerů zpracovávat vstupní text naráz, ne sekvenčně.

Hlavním přínosem práce [5] je způsob, kterým probíhá předtrénink 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 předtrénink modelu BERT byly použity následující úkoly.

**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 transformerů triviální.



Obrázek 3.8: dummy obrázek - pre-training a fine-tuning berta

Při tréninku je 15 % slov každé věty nahrazeno [MASK] tokenem, jehož původní význam je poté na základě okolního kontextu hádán.

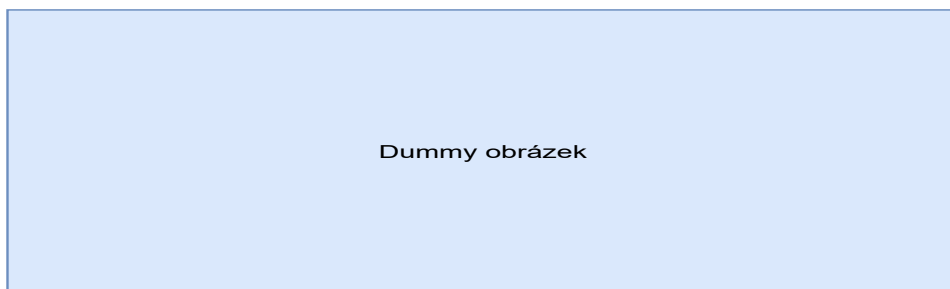
Pro tento úkol je na enkodér BERTa napojena klasifikační vrstva, pomocí které je určena pravděpodobnost pro každého slovo ve slovníku, které má [MASK] token nahradit.

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

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

Práce [5] 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: dummy obrázek - poziční, segmentové a token embeddings + special tokeny

Pro vektorovou reprezentaci jednotlivých tokenů jsou použity Wordpiece embeddings [27] popsány v 2.1.3. K těm je přidáno poziční kódování kvůli použití Transformer architektury a segmentační 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]. [CLS] token je přidán vždy na začátek vstupu. Token [SEP] je přidán na konec každé sekvence. [CLS] token je využit také pro klasifikační úlohy (například předtrénink na NSP).

### 3.3.2 ALBERT

Informace v této sekci jsou převzaty ze článku [8] *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 [8]. Vychází z modelu BERT a dosahuje lepších výsledků na datových sadách pro porozumění textu, přestože má mnohem menší počet parametrů, než velká verze BERTa. 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 BERTa 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 nepřislušnosti slov stejnému tématu.

Pro trénink modelu ALBERT je pro tento úkol použita stejná dvojice vět A a B, u kterých se určuje, zda-li B navazuje na A. Při nenávaznosti však není věta B náhodně vybrána z jiného korpusu, ale je pouze prohozena 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 pro odpovídání na otázky** Pro využití modelu ALBERT na konkrétní úkoly vyžadující porozumění textu je potřeba provést *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<sup>7</sup> dvojice indexů označující rozsah odpovědi.

Pro *fine-tuning* na daný úkol jsou optimalizovány parametry lineární klasifikační vrstvy i enkodéru ALBERT (vyžaduje delší trénink), nebo pouze lineární vrstvy.

---

<sup>7</sup>podmínky jako: start idx > end idx, 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 popsaného v části 6.5. Nejprve bude vysvětlen význam indexu v kontextu řazení dokumentů a poté algoritmy, které ho využívají k vyhledávání. Nakonec budou diskutovány problémy velké báze dat jako je Wikipedie.

Informace v této kapitole jsou převzaty z prezentací Stanfordského kurzu *Information Retrieval and Web Search*<sup>1</sup> a knihy *Introduction to Information Retrieval* [12].

### 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 všechny 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 a jeho délce. 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 části 6, konkrétně 6.2 a 6.5.

---

<sup>1</sup>Dostupné z: <https://web.stanford.edu/class/cs276/>

## 4.2 Algoritmy pro řazení dokumentů

Při dotazování na vytvořený index nestačí výčet dokumentů, které termy přítomné v dotazu obsahují. Ohodnocené vyhledávání umožňuje získání pouze množiny seřazených  $n$  nejvíce relevantních dokumentů.

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

**Četnost termu** (TF) je počet výskytů vyhledávaného termu v daném dokumentu. Vycházíme z předpokladu, že dokument s deseti výskyty termu bude pro dotaz více relevantní, než dokument obsahující vyhledávaný term pouze jednou. Neznamená to však, že bude dokument  $10\times$  více relevantní – růst relevance není přímo úměrný TF.

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

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

**Četnost dokumentů** (DF) je počet dokumentů, které obsahují vyhledávaný term. Účel je v identifikaci termů vyskytujících se jenom ve zlomku dokumentů, které nejspíš ponese zásadní informační hodnotu a adresovat tím neúměrnou závislost růst relevance dokumentu a četnosti termu. Obrácená četnost dokumentů (IDF) dělí celkový počet dokumentů  $N$  četností dokumentů pro daný term. Hodnota IDF je tedy vyšší pro vzácnější termy.

$$IDF_d = \log\left(\frac{N}{df_t}\right)$$

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

$$TF\text{-}IDF_{t,d} = TF_{t,d} \cdot IDF_d$$

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

$$S(q, d) = \sum_{t \in q \cap d} TF\text{-}IDF_{t,d}$$

### 4.2.2 BM25 (Best Match 25)

(Okapi) BM25 představeno v [18] je vylepšením klasické hodnotící funkce TF-IDF. Využívá poznatky statistiky a pravděpodobnosti pro řazení dokumentů dle jejich relevance s ohledem na vztah délky dokumentu a četnost 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:

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

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

- $k_1$  je konstanta ovlivňující 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 a 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 příkladu.

Mějme vyhledávaný dotaz „*term frequency*“ a dva dokumenty  $d_1$  a  $d_2$ :

	term	frequency		<b>TF-IDF</b>	<b>BM25</b>
$d_1$	1024	1	$d_1$	<b>87</b>	31
$d_2$	16	8	$d_2$	75	<b>43</b>

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 ( $k_1 = 2$ )

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

Pro BM25 existují také modifikace, jako BM25+, BM25L a další, které byly porovnány v [21].

- BM25L [10] adresuje znevýhodnění příliš dlouhých dokumentů klasickou BM25 funkcí.
- BM25+ [9] adresuje stejný problém jako BM25L, ale implementuje vylepšení odlišně

Závěr článku [21] ale ukazuje, že neexistuje modifikace, která by systematicky zlepšila dosažené výsledky vyhledávání.

## 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 (člověkem popsaná) kolekce dat pro trénink a vyhodnocování neuronových sítí. Obvykle obsahuje pro každý příklad vstupní data a jejich popis (anotaci).

Datové sady je možno rozdělit podle toho, v jaké fázi vývoje s nimi pracujeme.

- **Trénovací** Používá se pro naučení neuronové sítě. Při zpracovávání trénovací datové sady se model učí a upravuje své parametry. Na konci tréninku by měl model konvergovat ke 100 % úspěšnosti na trénovací datové sadě.
- **Validační** Slouží pro průběžné vyhodnocení existujícího modelu v průběhu vývoje. Model se z validačních dat neučí. Může být využit například k úpravě hyperparametrů nebo zastavení tréninku modelu, když se jeho úspěšnost na validační datové sadě přestane zlepšovat. Vytvořen může být vybraným zlomkem trénovacího datasetu nepoužitým pro trénink.
- **Testovací** 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 ...)

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*<sup>1</sup>.

---

<sup>1</sup> Crowdsourcing je nové slovo označující proces získání potřebné služby, nápadů, příspěvků, pomoci při řešení problémů od velké skupiny lidí. (<https://wikisofia.cz/wiki/Crowdsourcing>)



## 5.1 Datové sady pro angličtinu

Pro angličtinu existuje celá řada datových sad zaměřených na odpovídání na otázku. Nejznámější z nich je však SQuAD (*Stanford Question Answering Dataset*) [17]. Dataset obsahuje asi 100 000 anotovaných dvojic otázka–odpověď z více jak 500 různých článků z anglické Wikipedie. Existuje také rozšířená verze SQuAD datasetu. SQuAD2.0 [16] obsahuje navíc 50 000 otázek, na které v kontextu neexistuje odpověď. Model tedy musí umět určit, když dokument odpověď neobsahuje.

## 5.2 Datové sady pro češtinu

Čeština nabízí poměrně omezené zdroje oproti anglickým datovým sadám.

Nejznámější je datová sada SQuADv3 (Simple Question Answering Database) [19]. Obsahuje přes 13 000 příkladů otázek s odpovědí v přiloženém kontextu pocházejícím z české Wikipedie. Datová sada otázky směřované na časový údaj, entitu, lokaci, osobu ... včetně 16 % ano/ne otázek.

Existuje také česká verze anglického datasetu SQuAD prezentovaná v práci [11]. Datová sada je strojovým překladem původního datasetu SQuAD 1.1 a SQuAD2.0. 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. Ztráta části rozsahu datové sady je 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 vyznačení odpovědi v získaném textu a pro validaci výsledného systému.

Pro trénink anglického modelu byl vybrán anglický SQuAD 1.1 (obsahující i validační část datasetu). Po zvážení bylo rozhodnuto použít první verzi datasetu, protože budou testovací data obsahovat pouze otázky, pro které existuje na otevřené doméně odpověď.

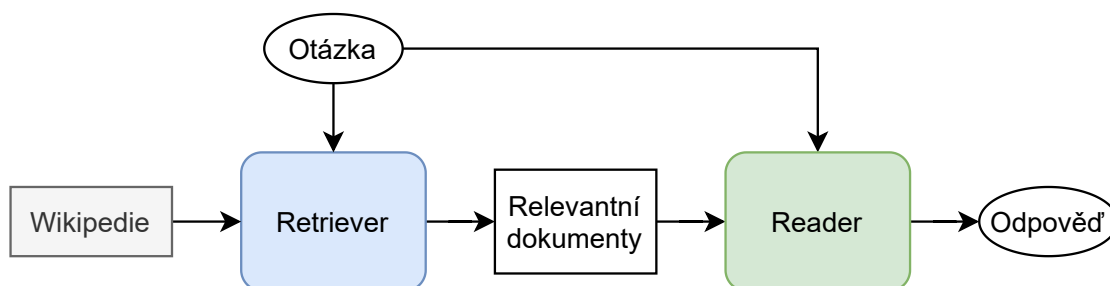
Pro trénink českého modelu byl vybrán český překlad anglického datasetu SQuAD 1.1 [11] kvůli jeho velikosti a porovnatelnosti s anglickou verzí datové sady.

Jako testovací datová sada byl zvolen český dataset SQuADv3, ze kterého bylo odstraněno 16 % otázek očekávajících ano/ne odpovědi. Datová sada nejlépe odpovídá reálnému scénáři užití systému, protože obsahuje otázky a odpovědi z článků z české Wikipedie (narozdíl od trénovacích datasetů).

## Kapitola 6

# Návrh systému a implementace jednotlivých komponent

V této kapitole je rozebrán návrh systému a implementace jeho částí. Nejprve je popsán zvolený přístup k řešení problému a rozdělení problému na části reprezentující blokové schéma celého systému. Následuje popis nástrojů a knihoven použitých pro implementaci jednotlivých komponent. Nejdůležitější částí jsou sekce o implementaci dvou hlavních nezávislých částí 6.4 a 6.5. Nakonec je shrnut výsledný tok programu pro odpovídání na otázky nad českou Wikipedií.



Obrázek 6.1: Generické schéma (prezentované například v [3]) open-domain QA systému

### 6.1 Zvolený přístup k problému

Odpovídání na otázky na otevřené doméně (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 a druhou je *Reader*, který provede extrakci odpovědi ze získaného kontextu [3].

*Retriever* části systému je věnována pozornost v sekci 6.5.

Pro návrh systému bylo však největší dilema v realizaci *Reader* části systému pro kvalitní extrakci odpovědi v češtině. Chtěl jsem využít některého z populárních předtrénovaných modelů (3.3) založených na Transformers [22] architektuře (3.2.2). Proto jsem se rozhodl vyzkoušet přístup, kdy je nativně anglickému ALBERT modelu, který dosahuje špičkových výsledků<sup>1</sup>, strojově přeložen vstupní dokument i otázka z češtiny do angličtiny.

<sup>1</sup><https://rajpurkar.github.io/SQuAD-explorer/>

Pro porovnání jsem zvolil vícejazyčný model BERTa (m-BERT<sup>2</sup>), který podporuje 104 světových jazyků s nejrozsáhlejší Wikipedií, trénovaný na českém překladu datové sady SQuAD 1.1 [11] [17].

## 6.2 Návrh jednotlivých částí systému

V této sekci bude popsán návrh systému, tedy hlavně *Retrieveru*, *Readeru* a části pro zpracování datové sady SQuAD v3.0 [19] pro závěrečné vyhodnocení.

### 6.2.1 Návrh Retrieveru

Získání  $n$  relevantních dokumentů kandidujících pro zodpovězení otázky vyžaduje několik na sebe navazujících kroků. 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 Wikipedii, bude dobré z otázky získat co nejvíce informací, které by mohly vést ke *správným článkům*. Až po jejich získání jsou jednotlivé články rozděleny na odstavce a ty prohledávány.

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 4.2.2) z *dumpu* titulků<sup>3</sup> české Wikipedie po lemmatizaci otázky a odstranění stop slov,
- **rozpoznání (pojmenované) entity**, která by se mohla vyskytovat v titulku článku a usnadnit tím *Wiki Search*.

Tímto tedy získáme 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 jednotlivé odstavce, normalizovat jejich délku, lemmatizovat je, odstranit stop slova a indexovat je. Odstavce jsou poté seřazeny algoritmem BM25 a z nich vybrány první tři nejvhodnější. 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 *readeru*) jsou včetně otázky přeloženy do angličtiny.

---

<sup>2</sup><https://github.com/google-research/bert/blob/master/multilingual.md>

<sup>3</sup>[https://cs.wikipedia.org/wiki/Wikipedie:St%C3%A1hnut%C3%AD\\_dat%C3%A1ze](https://cs.wikipedia.org/wiki/Wikipedie:St%C3%A1hnut%C3%AD_dat%C3%A1ze)

- 6.2.2 Návrh Readeru
- 6.2.3 Návrh vyhodnocení
- 6.2.4 Popis výsledného systému
- 6.3 Použité nástroje a technologie pro implementaci
- 6.4 Implementace a trénink readeru
- 6.5 Implementace retrieveru

## Kapitola 7

# Vyhodnocení systému a rozbor chyb

7.1 Vysvětlení základních metrik

7.2 Postup při vyhodnocování výsledného systému

7.3 Porovnání výsledků se současným stavem poznání

7.4 Rozbor chyb a možnosti dalšího vývoje

## Kapitola 8

## Závěr

# Literatura

- [1] AVINASH SHARMA V. *Understanding Activation Functions in Neural Networks*. 2017. [Online; accessed 20-March-2021]. Dostupné z: <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>.
- [2] BAHDANAU, D., CHO, K. a BENGIO, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv*. Zář 2014, sv. 1409.
- [3] CHEN, D., FISCH, A., WESTON, J. a BORDES, A. Reading Wikipedia to Answer Open-Domain Questions. *CoRR*. 2017, abs/1704.00051. Dostupné z: <http://arxiv.org/abs/1704.00051>.
- [4] CHRISTOPHER OLAH. *Understanding LSTM Networks*. 2015. [Online; accessed 20-March-2021]. Dostupné z: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [5] DEVLIN, J., CHANG, M., LEE, K. a TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*. 2018, abs/1810.04805. Dostupné z: <http://arxiv.org/abs/1810.04805>.
- [6] HOCHREITER, S. a SCHMIDHUBER, J. Long Short-term Memory. *Neural computation*. Prosin 1997, sv. 9, s. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.
- [7] JAY ALAMMAR. *The Illustrated Transformer*. 2018. [Online; accessed 17-March-2021]. Dostupné z: <http://jalammar.github.io/illustrated-transformer/>.
- [8] LAN, Z., CHEN, M., GOODMAN, S., GIMPEL, K., SHARMA, P. et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *CoRR*. 2019, abs/1909.11942. Dostupné z: <http://arxiv.org/abs/1909.11942>.
- [9] LV, Y. a ZHAI, C. Lower-Bounding Term Frequency Normalization. In: New York, NY, USA: Association for Computing Machinery, 2011, s. 7–16. CIKM '11. DOI: 10.1145/2063576.2063584. ISBN 9781450307178. Dostupné z: <https://doi.org/10.1145/2063576.2063584>.
- [10] LV, Y. a ZHAI, C. When Documents Are Very Long, BM25 Fails! In: New York, NY, USA: Association for Computing Machinery, 2011, s. 1103–1104. SIGIR '11. DOI: 10.1145/2009916.2010070. ISBN 9781450307574. Dostupné z: <https://doi.org/10.1145/2009916.2010070>.
- [11] MACKOVÁ, K. a STRAKA, M. *Reading Comprehension in Czech via Machine Translation and Cross-lingual Transfer*. 2020.

- [12] MANNING, C. D., RAGHAVAN, P. a SCHÜTZE, H. *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008. ISBN 978-0-521-86571-5. Dostupné z: <http://nlp.stanford.edu/IR-book/information-retrieval-book.html>.
- [13] MIKOLOV, T., CHEN, K., CORRADO, G. a DEAN, J. *Efficient Estimation of Word Representations in Vector Space*. 2013.
- [14] MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G. a DEAN, J. Distributed Representations of Words and Phrases and their Compositionality. *CoRR*. 2013, abs/1310.4546. Dostupné z: <http://arxiv.org/abs/1310.4546>.
- [15] PENNINGTON, J., SOCHER, R. a MANNING, C. Glove: Global Vectors for Word Representation. In: Leden 2014, sv. 14, s. 1532–1543. DOI: 10.3115/v1/D14-1162.
- [16] RAJPURKAR, P., JIA, R. a LIANG, P. Know What You Don't Know: Unanswerable Questions for SQuAD. *CoRR*. 2018, abs/1806.03822. Dostupné z: <http://arxiv.org/abs/1806.03822>.
- [17] RAJPURKAR, P., ZHANG, J., LOPYREV, K. a LIANG, P. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. *CoRR*. 2016, abs/1606.05250. Dostupné z: <http://arxiv.org/abs/1606.05250>.
- [18] ROBERTSON, S., WALKER, S., JONES, S., HANCOCK BEAULIEU, M. a GATFORD, M. Okapi at TREC-3. In: 1994. Dostupné z: [https://www.researchgate.net/publication/221037764\\_Okapi\\_at\\_TREC-3](https://www.researchgate.net/publication/221037764_Okapi_at_TREC-3).
- [19] SABOL, R., MEDVEĎ, M. a HORÁK, A. Czech Question Answering with Extended SQuAD 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* [tištěná verze "print"]. Brno: Tribun EU, 2019, s. 99–108. ISBN 978-80-263-1530-8.
- [20] TONY YIU. *Understanding Neural Networks*. 2019. [Online; accessed 19-March-2021]. Dostupné z: <https://towardsdatascience.com/understanding-neural-networks-19020b758230>.
- [21] TROTMAN, A., PUURULA, A. a BURGESS, B. Improvements to BM25 and Language Models Examined. In: *Proceedings of the 2014 Australasian Document Computing Symposium*. New York, NY, USA: Association for Computing Machinery, 2014, s. 58–65. ADCS '14. DOI: 10.1145/2682862.2682863. ISBN 9781450330008. Dostupné z: <https://doi.org/10.1145/2682862.2682863>.
- [22] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L. et al. Attention Is All You Need. *CoRR*. 2017, abs/1706.03762. Dostupné z: <http://arxiv.org/abs/1706.03762>.
- [23] WIKIPEDIA CONTRIBUTORS. *Neuron — Wikipedia, The Free Encyclopedia*. 2020. [Online; accessed 19-March-2021]. Dostupné z: <https://cs.wikipedia.org/wiki/Neuron>.
- [24] WIKIPEDIA CONTRIBUTORS. *Príloha: Frekvenční seznam (čeština) — Wikislovník*. 2020. [Online; accessed 17-March-2021]. Dostupné z:



[https://cs.wiktionary.org/wiki/P%C5%99%C3%ADloha:Frekven%C4%8Dn%C3%AD\\_seznam\\_\(%C4%8De%C5%A1tina\)](https://cs.wiktionary.org/wiki/P%C5%99%C3%ADloha:Frekven%C4%8Dn%C3%AD_seznam_(%C4%8De%C5%A1tina)).

- [25] WIKIPEDIA CONTRIBUTORS. *Lemmatizátor* — *Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 19-March-2021]. Dostupné z: <https://cs.wikipedia.org/wiki/Lemmatiz%C3%A1tor>.
- [26] WIKIPEDIA CONTRIBUTORS. *Stop word* — *Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 17-March-2021]. Dostupné z: [https://en.wikipedia.org/wiki/Stop\\_word](https://en.wikipedia.org/wiki/Stop_word).
- [27] 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. *CoRR*. 2016, abs/1609.08144. Dostupné z: <http://arxiv.org/abs/1609.08144>.
- [28] ZHU, M., AHUJA, A., JUAN, D.-C., WEI, W. a REDDY, C. K. Question Answering with Long Multiple-Span Answers. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Listopad 2020, s. 3840–3849. DOI: 10.18653/v1/2020.findings-emnlp.342. Dostupné z: <https://www.aclweb.org/anthology/2020.findings-emnlp.342>.