Univerzita Pavla Jozefa Šafárika v Košiciach

Prírodovedecká fakulta

METÓDY TRANSFORMEROV A ICH APLIKÁCIA V OBLASTI TEXTOVEJ ANALÝZY

DIPLOMOVÁ PRÁCA

Študijný odbor: **Analýza dát a umelá inteligencia**
Školiace pracovisko: **Ústav informatiky**
Vedúci záverečnej práce: **RNDr. Šimon Horvát, PhD.**
Konzultant: **doc. RNDr. Ľubomír Antoni, PhD.**

Košice 2025

Martin Bača

Poďakovanie

Na tomto mieste by som rád vyjadril svoju úprimnú vďaku všetkým, ktorí ma podporovali počas celého štúdia a akýmkoľvek spôsobom prispeli k realizácii tejto práce.
Predovšetkým ďakujem svojmu školiteľovi RNDr. Šimonovi Horvátovi, PhD. a svojmu konzultantovi doc. RNDr. Ľubomírovi Antonimu, PhD. za odborné vedenie, cenné rady, trpezlivosť a podporu, ktorú poskytovali počas tvorby tejto práce.
Poďakovanie patrí aj mojej rodine, ktorá ma počas štúdia neustále podporovala, povzbudzovala a vytvárala mi zázemie plné porozumenia a pokoja.
V neposlednom rade ďakujem všetkým priateľom, spolužiakom a pedagógom, ktorí ma sprevádzali, motivovali a inšpirovali počas celého štúdia.

Univerzita P. J. Šafárika v Košiciach
Prírodovedecká fakulta

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Martin Bača
Študijný program:	analýza dát a umelá inteligencia (medziodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	Informatika, Matematika
Typ záverečnej práce:	Diplomová práca
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Metódy transformérov a ich aplikácia v oblasti textovej analýzy

Názov EN: Transformer methods and their applications in text analysis

Cieľ:
1. Spracovať prehľad moderných neurónových sietí, najmä transformérov, a ich využitie v úlohách zodpovedania otázok.
2. Vytvoriť alebo zozbierať dataset otázok a odpovedí z oblasti logistiky, s využitím syntetickej generácie (napr. pomocou Lamma).
3. Implementovať model typu BERT (alebo jeho variant) trénovaný na pripravenom datasete.
4. Otestovať model na úlohe zodpovedania otázok a vyhodnotiť jeho výkon pomocou rôznych metrík

Literatúra:
1. Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
2. Tunstall, L., Von Werra, L., & Wolf, T. (2022). Natural language processing with transformers. " O'Reilly Media, Inc.".
3. Devlin, Jacob, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805 (2018).
4. Rajpurkar, Pranav, et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text." Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (2016).

Vedúci:	RNDr. Šimon Horvát, PhD.
Konzultant:	doc. RNDr. Ľubomír Antoni, PhD.
Oponent:	RNDr. Zoltán Szoplák
Ústav :	ÚINF - Ústav informatiky
Riaditeľ ústavu:	doc. RNDr. Ondrej Krídlo, PhD.

Dátum schválenia: 28.04.2025

Abstrakt

Predložená diplomová práca skúma aplikácie Transformerov v oblasti textovej analýzy, s osobitným zameraním na vzťah medzi veľkosťou modelu a jeho výkonom. V úvode implementačnej časti popisuje vytvorenie rozsiahleho multilingválneho datasetu, ktorý kombinuje extraktívne, binárne a číselné otázky v anglickom, nemeckom a francúzskom jazyku. Dataset bol generovaný a anotovaný automatizovaným spôsobom s využitím veľkých generatívnych jazykových modelov, čím sa eliminovala potreba manuálnej anotácie. V ďalšej časti práca rozširuje architektúru Transformer o klasifikačnú hlavu, ktorá umožní rozlíšiť typ otázky a v prípade binárnych otázok aj predikovať odpoveď vo forme Pravda/Nepravda. Experimentálna časť porovnáva výkonnosť modelov rôznych veľkostí a výpočtových nárokov. Najlepšie výsledky v multilingválnej úlohe dosiahol model `XLM-RoBERTa Large`, ktorý vykázal accuracy 90,6 % pri klasifikácii binárnych otázok a F1-skóre 87,7 % pri určovaní odpovedí na extraktívne otázky.

Kľúčové slová: *Question Answering, Transformer, generovanie datasetu, extraktívny QA, automatizovaná anotácia, binárna klasifikácia*

Abstract

The presented master's thesis explores the applications of Transformer models in the domain of text analysis, with a particular focus on the relationship between model size and its performance. The initial part of the implementation describes the construction of a large multilingual dataset that combines extractive, binary and numerical questions in English, German and French. The dataset was generated and annotated automatically using large generative language models, thereby eliminating the need for manual annotation. In the next part, the Transformer architecture is extended with a classification head that enables the model to distinguish the question type and, in the case of binary questions, predict a True/False answer. The experimental part compares the performance of models of various sizes and computational requirements. The best results in the multilingual task were achieved by the `XLM-RoBERTa Large` model, which attained 90,6 % accuracy in binary question classification and an F1-score of 87,7 % in extractive question answering.

Keywords: *Question Answering, Transformer, dataset generation, extractive QA, automated annotation, binary classification*

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

S exponenciálne rastúcim objemom dát, ktorý každodenne produkuje súčasná spoločnosť, môžeme považovať efektívne vyhľadávanie a spracovanie relevantných informácií za veľmi dôležitú výzvu. Tradičné spôsoby vyhľadávania informácií sú prevažne založené na vyhľadávaní podľa kľúčových slov a jednoduchých heuristických pravidlách. Tieto spôsoby vyhľadávania sú však relatívne jednoduché a často nedokážu splniť nároky používateľov, ktorí očakávajú okamžité, presné a relevantné odpovede na svoje otázky.

Na dané potreby reaguje oblasť odpovedania na otázky (Question Answering, QA), jedna z najrýchlejšie sa vyvíjajúcich oblastí spracovania prirodzeného jazyka (NLP). Jej hlavným zameraním je vývoj moderných QA systémov využívajúcich pokročilé modely strojového učenia, ktoré sú schopné porozumieť štruktúre a významu jazyka v kontexte. V súčasnosti sú za štandard v tejto oblasti považované modely Transformerov, ktoré od svojho predstavenia v roku 2017 reprezentujú špičku vo svojej oblasti. Tieto modely disponujú mechanizmami, ktoré im umožňujú porozumieť prirodzenému jazyku do väčšej hĺbky ako ich predchodcovia a práve vďaka nim môžu dosahovať špičkové výsledky.

Táto práca sa zameriava na skúmanie efektívnosti a presnosti QA systémov, ktoré využívajú najmodernejšie modely Transformerov. Jej hlavným zameraním je analýza a porovnanie výkonnosti týchto modelov v extraktívnych a binárnych úlohách QA. S týmto účelom definujeme niekoľko monolingválnych aj multilingválnych konfigurácií, v ktorých budeme skúmať schopnosť modelov odpovedať na rôzne typy otázok. Použité modely sa budú líšiť veľkosťou a výpočtovou zložitosťou, počnúc kompaktnými architektúrami optimalizovanými pre prostredia s obmedzenými výpočtovými zdrojmi, cez modely predstavujúce kompromis medzi výkonom a efektivitou, až po rozsiahle modely navrhnuté pre maximálnu presnosť v podmienkach bez výrazných výpočtových obmedzení.

Dôležitou motiváciou pre vznik tejto práce sú limitácie existujúcich datasetov, ktoré

často buď obsahujú záznamy iba v jednom jazyku, alebo sú zamerané len na jeden typ otázok. Práve z tohto dôvodu vytvárame rozsiahly a kvalitne anotovaný dataset určený na tréning QA modelov. Navrhovaný dataset obsahuje tri rôzne typy otázok - extraktívne, číselné a binárne, a pokrýva tri svetovo rozšírené jazyky: anglický, nemecký a francúzsky. Dataset zároveň slúži ako demonštrácia navrhovaného postupu generovania dát a otvára priestor na jeho budúce rozširovanie ďalšími typmi otázok a ďalšími jazykovými variantmi. Jeho základom sú náhodne generované údaje uložené v štruktúrovanom formáte JSON, na ktorých transformáciu do podoby prirodzeného jazyka bol navrhnutý proces automatizovaného generovania kontextov a otázok. Proces generovania a anotácie odpovedí bol vykonaný s využitím pokročilých generatívnych jazykových modelov, čo eliminovalo potrebu prácneho a časovo náročného manuálneho anotovania.

Prínosom práce je nielen vytvorenie unikátneho multilingválneho datasetu, ale i návrh a implementácia rozšírenia existujúcich modelov Transformerov o klasifikačnú hlavu, ktorá umožňuje efektívne rozlíšiť typ otázky na extraktívny alebo binárny. V prípade, že model označí otázku ako binárnu, využíva druhý výstup klasifikačnej hlavy na predikciu korektnej binárnej odpovede (`Pravda`/`Nepravda`). Tento dvojstupňový prístup zvyšuje flexibilitu modelu a pri jeho ďalšom rozvoji môže predstavovať významný prínos v podobe vhodnejšej formulácie odpovedí pre rôzne typy otázok.

Štruktúra práce je členená do siedmich kapitol, ktoré postupne sprevádzajú čitateľa od teoretických základov k praktickej realizácii a vyhodnoteniu výsledkov. Druhá kapitola poskytuje prehľad relevantnej literatúry a zameriava sa na historický vývoj QA systémov. Tretia kapitola sa venuje podrobnej analýze úlohy Question Answering a dostupných datasetov. Štvrtá kapitola mapuje vývoj sekvenčných jazykových modelov, poukazuje na ich obmedzenia a vysvetľuje princípy architektúry Transformer, ktorá predstavuje jadro moderných jazykových modelov. Piata kapitola podrobne opisuje metodológiu a implementáciu práce, vrátane procesu generovania dát, tokenizácie a nastavenia tréningových parametrov modelov. V šiestej kapitole sú prezentované výsledky, ich dôkladná analýza a porovnanie s relevantnými významnými prácami. V záverečnej siedmej kapitole sú zhrnuté hlavné závery, prínosy a budúce smerovanie výskumu v tejto oblasti.

Kapitola 2

Prehľad literatúry

2.1 Úvod do QA systémov

Question Answering je podoblasť spracovania prirodzeného jazyka (NLP), ktorej cieľom je vyvinúť modely schopné automaticky odpovedať na otázky zo štruktúrovaných aj neštruktúrovaných dát. Táto oblasť sa postupne vyvíjala od prvých prístupov, ktoré využívali pravidlá a informačný zisk, až po modely hlbokého učenia. V súčasnosti sú najmodernejšie modely založené na architektúre Transformer [1].

Tradičné QA systémy založené na informačnom zisku (IR - Information Retrieval) sa spoliehali na techniky, ako sú TF-IDF a BM25 [2, 3], ktoré vyhľadávali dokumenty relevantné pre dopyt a následne zoradili možné odpovede na základe priradeného skóre. Zatiaľ čo tieto systémy boli účinné pri porovnávaní kľúčových slov, mali značné problémy s kontextovým porozumením [4, 5, 6]. Prístupy hlbokého učenia založené na architektúrach LSTM (Siete s dlhou krátkodobou pamäťou) a CNN (Konvolučné neurónové siete) priniesli zlepšenie v modelovaní jazyka, avšak nedokázali zachytiť vzdialené závislosti v texte. Obmedzenia týchto metód poukázali na potrebu modelov, ktoré dokážu pracovať s kontextom efektívnejšie. Túto potrebu naplnili modely založené na architektúre Transformer, ktoré sa stali kľúčovým prelomom v oblasti spracovania prirodzeného jazyka.

2.2 Vývoj QA systémov založených na architektúre Transformer

Zavedením mechanizmu self-attention priniesol model Bidirectional Encoder Representations from Transformers (BERT) revolúciu v oblasti QA, keď umožnil vytváranie

hlbokých kontextových reprezentácií slov. Na rozdiel od predchádzajúcich modelov, Transformerove modely pomocou mechanizmu self-attention zachytávajú vzťahy medzi každou dvojicou prvkov v rámci kontextového okna. Tento prístup umožňuje modelu budovať bohatšie a kontextovo závislé reprezentácie, čo vedie k lepšiemu porozumeniu významu slov v rôznych jazykových súvislostiach [7].

BERT a jeho rozšírenia

BERT je predtrénovaný na úlohách modelovania maskovaného jazyka (MLM) a predikcie nasledujúcej vety (NSP), vďaka čomu je vhodný na úlohy QA. Jemné ladenie (fine-tuning) BERT na datasetoch, ako je SQuAD, preukázalo jeho špičkový extrakčný výkon v oblasti QA, ktorý výrazne prekonáva modely založené na LSTM a CNN [7]. BERT má však kľúčové obmedzenia, medzi ktoré patrí obmedzenie dĺžky vstupných údajov na 512 tokenov, ktoré obmedzuje jeho schopnosť spracovať dlhé dokumenty [7], a závislosť na statických predtrénovaných úlohách (MLM a NSP), ktoré nemusia poskytovať dostatočnú prispôsobivosť pre špecifické domény.

Od predstavenia BERT bolo vyvinutých mnoho jeho variantov, ktoré zlepšili jeho schopnosti a prispôsobili ho na použitie v rôznych doménach:

- **RoBERTa (Robustly Optimized BERT Pretraining Approach):** Vylepšený variant BERT, ktorý dosahuje špičkové výsledky na benchmarkoch GLUE, RACE a SQuAD [8].
- **Bangla-BERT:** Doménovo špecifický model BERT prispôsobený pre jazyk Bangla, ktorý vykazuje lepší výkon v transferovom učení a viacjazyčných aplikáciách NLP Kowsher et al. [1].
- **IndoBERT:** Tento model sa zameriava na indonézsky jazyk, pričom zdôrazňuje prispôsobivosť variantov BERT v rôznych jazykových prostrediach [9].
- **MédicoBERT:** Adaptácia BERT prispôsobená na úlohy QA v medicínskej oblasti v španielskom jazyku, ktorá ukazuje potenciál doménovo špecifického predtréningu [7].

RoBERTa: Prekonanie obmedzení BERT v QA úlohách

S cieľom odstrániť obmedzenia BERT výskumníci predstavili model RoBERTa (Robustly Optimized BERT Pretraining Approach), ktorý z procesu tréningu odstránil úlohu NSP, zvýšil veľkosť dávky (batch size) a použil dynamickú stratégiu maskovania. Tieto úpravy viedli k dosiahnutiu špičkového výkonu na benchmarkoch QA vrátane SQuAD, GLUE a RACE [8].

Empirické vyhodnotenia (metrík) ukazujú, že RoBERTa prekonáva BERT v extrakčnom QA s lepšími výsledkami Exact Match (EM) a F1-skóre. Dokáže si lepšie poradiť so zložitými jazykovými štruktúrami a zlepšuje schopnosť modelu presnejšie odpovedať na otázky, najmä v open-domain úlohách QA. Okrem toho architektúra RoBERTa umožňuje lepšie transferové učenie pre viacjazyčné a doménovo špecifické úlohy QA [8].

2.3 Rozšírenie QA o binárnu klasifikáciu

Binárna QA (klasifikácia Áno/Nie) predstavuje ďalšie výzvy nad rámec extraktívnej QA. Zatiaľ čo modely ako BERT a RoBERTa dosahujú vysokú presnosť pri extrakcii textových úsekov, nemajú explicitné mechanizmy na spracovanie binárnej klasifikácie. Pri binárnej QA musí model odvodiť odpoveď „Áno“ alebo „Nie“ na základe dôkazov nájdených v texte, čo často vyžaduje logické uvažovanie, overovanie faktov a detekciu negácie.

Výzvy v binárnej QA spočívajú v spracovaní nejednoznačných a protichodných informácií. Taktiež je často potrebné zobrať do úvahy informácie z viacerých viet, odstavcov, či dokumentov. Na rozdiel od extraktívnej QA, ktorá sa zameriava na priame porovnávanie fráz, binárna QA vyžaduje pochopenie implicitných vzťahov a jemných kontextových súvislostí. Navyše, faktory, ako sú negácia, špekulatívny jazyk a syntaktická zložitosť, často vedú k zvýšenej chybovosti [10]. Clark et al. [11] vo svojom článku porovnali rôzne prístupy ku zodpovedaniu otázok typu binárna. Autori experimentovali s plytkými modelmi (TF-IDF), rekurentnými neurónovými sieťami (RNN) s mechanizmom pozornosti, transferovým učením zo súboru dát s viacerými správnymi odpoveďami, predtréningom založeným na entailmente (logickej implikácii) pomocou MultiNLI a modelmi založenými na architektúre Transformerov ako sú OpenAI GPT a BERT. Experimentálnym prístupom zistili, že BERT dosiahol najlepšie výsledky vďaka obojsmerným kontextovým reprezentáciám a robustnému predtréningu bez učiteľa na veľkých jazykových korpusoch, pričom dosiahol accuracy 76,90 %.

Ďalším prístupom k riešeniu binárnej QA je stratégia viacstupňového pipeline modelu [12], kde autori použili dvojstupňový prístup kombinujúci extraktívne QA na získanie relevantných textových úsekov z vedeckých abstraktov a binárnu QA na klasifikáciu odpovedí ako pravdivé, nepravdivé alebo neutrálne. V tomto článku boli vykonané experimenty s rôznymi modelmi založenými na architektúre Transformer, pričom autori zistili, že RoBERTa dosahuje najlepšie výsledky (F-1 skóre: 97,07 %) vďaka svojim schopnostiam kódovania viet, zatiaľ čo TinyBERT (F-1 skóre: 95,60 %) ponúka rovnováhu medzi

kvalitou predikcií a výpočtovou efektivitou.

Empirické výsledky potvrdzujú účinnosť priameho dolaďovania, ako je vidieť v prípade BioASQ, kde pridanie 256-neurónovej skrytej vrstvy do klasifikátora BioBERT zvýšilo Macro-F1-skóre z 0,64 na 0,72 [10] a v prípade PubMedQA, kde štruktúrované stratégie dolaďovania zvýšili accuracy z 55 % na 68 % [13].

Alternatívne a hybridné metódy

Pred predstavením Transformerov sa v úlohe binárnej QA často používali modely založené na BiLSTM, kde obojsmerný LSTM spracúval otázku a kontext, pričom finálna predikcia bola získaná vrstvou Softmax. Zavedenie kontextuálnych ELMo embeddingov do BiLSTM architektúry zvýšilo dev-set accuracy zo 70,28 % na 71,41 %, čo predstavuje zlepšenie o približne 1,1 %. Avšak BERT-large dosiahol accuracy 78,09 % na dev-sete (76,70 % na test-sete), čím významne prevýšil výkon LSTM-based modelov a preukázal svoju schopnosť zachytiť zložité jazykové súvislosti [11]. Tieto výsledky potvrdzujú, že hoci BiLSTM modely sú schopné naučiť sa základné vzory, hlbšie Transformerové architektúry prinášajú podstatne lepšiu generalizáciu a inferenčnú silu.

Bežný hybridný prístup kombinoval vyhľadávanie informácií (IR) s klasifikáciou, kde IR vyhľadával relevantné úryvky a klasifikátor (LSTM alebo BERT) určoval odpoveď. V biomedicínskom QA Bio-AnswerFinder sa na extrakciu viet s odpoveďou použili embeddingy BioWordVec + LSTM, po ktorých nasledovala klasifikácia typu Pravda/Nepravda na základe vopred definovaných pravidiel [12], [14]. V inom článku autori preukázali, že efektívnym prístupom je hybridný systém kombinujúci BERT s predtrénovaním na MultiNLI. Predtrénovanie na MultiNLI obohatilo BERT o schopnosť presnejšie rozpoznávať entailment a kontradikciu, čo je kľúčové pre úlohu binárnej QA. Táto hybridná metóda dosiahla accuracy 80,43 %, čím výrazne prekonala alternatívne prístupy skúmané v tomto článku a potvrdila prínos kombinovaného využitia jazykového modelu a entailmentového predtrénovania [11].

Niektoré práce rozširujú modely Transformerov o grafové neurónové siete (GNN), ktoré napomáhajú efektívnejšiemu modelovaniu vzťahov a závislostí medzi entitami. Napríklad model GEAR vytváral grafové reprezentácie na základe embeddingov získaných z modelu BERT a aplikoval ich na overovanie tvrdení v úlohe FEVER Boolean, kde dosiahol zlepšenie výkonu o 3 % [15].

Priestor na ďalší výskum

Existujúce datasety pre binárnu QA, ako napríklad BoolQ a PubMedQA, sa zvyčajne zameriavajú na jeden z dvoch typov otázok - buď na extraktívne otázky, alebo na binárne otázky. Naproti tomu nami vytvorený dataset integruje oba tieto typy, čím

kladie na modely vyššie nároky a zároveň predstavuje komplexnejšie referenčné kritérium.

Okrem toho je náš dataset automaticky generovaný prostredníctvom veľkých jazykových modelov (LLM), ktoré generujú otázky a k nim prislúchajúce správne odpovede. Tento automatizovaný prístup eliminuje nutnosť ľudskej anotácie, avšak prináša nové výzvy v podobe kontroly správnosti generovaných dát. Vďaka tomu, že náš súbor údajov obsahuje oba formáty QA, umožňuje vyhodnocovanie modelov v prostredí zmiešaných úloh, čo je viac podobné scenárom QA v reálnom svete.

Naším prínosom je generovanie otázok vo viacerých jazykoch, pričom okrem anglického jazyka dataset zahŕňa aj kontexty a otázky v nemeckom a francúzskom jazyku. Tento formát datasetu umožňuje porovnávanie modelov naprieč viacerými jazykmi a umožňuje napríklad skúmať generalizačné schopnosti modelov pri práci s rôznymi jazykovými rodinami. Týmto môže vytvorený dataset prispievať k vývoju univerzálnejších a robustnejších QA systémov.

Kapitola 3

Question Answering

Question answering (QA) je oblasť informatiky v rámci spracovania prirodzeného jazyka (NLP) a vyhľadávania informácií. Venuje sa vývoju systémov, ktoré dokážu pomocou prirodzeného jazyka odpovedať na otázky vyjadrené v prirodzenom jazyku. Tieto systémy analyzujú otázky, vyhľadávajú relevantné informácie v rozsiahlych dátach a poskytujú užívateľovi jasné a čitateľné odpovede.

S narastajúcim objemom dát a informácií sa stáva vyhľadávanie relevantných údajov čoraz komplexnejšou úlohou. Táto výzva motivovala vývoj systémov QA , ktorých cieľom je efektívnejšie sprostredkovanie prístupu k informáciám. Pred príchodom QA systémov bolo tradičné vyhľadávanie založené na identifikácii relevantných dokumentov pomocou kľúčových slov, pričom používateľ musel následne manuálne analyzovať získané zdroje s cieľom extrahovať požadované informácie.

Tradičné metódy vyhľadávania sú založené na podobnom princípe, pričom ich prínosom je automatizácia identifikácie relevantných zdrojov informácií a poskytnutie zoznamu odkazov priamo užívateľovi. Avšak interpretácia a extrakcia požadovaných údajov zostáva na samotnom používateľovi. V kontraste s tým systémy QA výrazne zefektívňujú tento proces priamym generovaním odpovede na položenú otázku, čím minimalizujú čas potrebný na vyhľadávanie a spracovanie informácií.

Rozlíšenie podľa prístupu k získavaniu informácií

Vo všeobecnosti existujú dva základné prístupy k úlohám QA. Prvým z nich je extraktívny QA, ktorý identifikuje a extrahuje odpoveď priamo z poskytnutých textov alebo zdrojov údajov. Na tento účel sa využívajú techniky, ako je rozpoznávanie pomenovaných entít (NER) a predikcia rozsahu, ktoré umožňujú lokalizovať konkrétne textové segmenty obsahujúce odpoveď na danú otázku. Napríklad extraktívny QA systém môže byť požiadaný, aby na základe dokumentu určil počet obyvateľov konkrétnej krajiny.

Druhým prístupom je generatívny QA, ktorý na rozdiel od extraktívneho QA prístupu syntetizuje odpovede, pričom využíva znalosti získané z rozsiahlych dátových súborov počas tréningu. Tieto systémy nie sú obmedzené na doslovné získavanie informácií z poskytnutých dokumentov, ale namiesto toho generujú komplexné a kontextovo prispôsobené odpovede. Často pritom využívajú veľké jazykové modely (LLM), ktoré umožňujú tvorbu prirodzených a koherentných textov. Príkladmi veľkých jazykových modelov založených na generatívnej umelej inteligencii (Generative AI) sú LLaMA (Meta), Gemini (Google), či Mistral.

Delenie podľa rozsahu znalostí s ktorými model pracuje

Ďalším kritériom klasifikácie QA systémov je rozsah znalostí, v rámci ktorého dokážu poskytovať relevantné odpovede. Open-domain QA systémy sú navrhnuté tak, aby dokázali odpovedať na otázky z rôznych tematických oblastí bez obmedzenia na konkrétnu doménu. Spoliehajú sa na rozsiahle všeobecné znalosti, efektívne vyhľadávanie a organizáciu údajov, pričom často využívajú metódy, ako napríklad ontologické modelovanie. Vďaka svojej širokej adaptabilite sú tieto systémy vhodné pre aplikácie, ktoré si vyžadujú vysokú mieru univerzálnosti, ako napríklad virtuálni asistenti alebo vyhľadávače.

Na druhej strane existujú closed-domain QA systémy, ktoré sú špecializované na konkrétne oblasti, ako je medicína, právo alebo inžinierstvo. Tieto systémy využívajú doménovo špecifické znalosti na poskytovanie presných a detailných odpovedí, prispôsobených danej oblasti. Napríklad, closed-domain QA systém v medicíne môže asistovať lekárom pri diagnostike odpovedaním na odborné diagnostické otázky pomocou informácií získaných z klinických záznamov a medicínskych databáz.

3.1 Historický vývoj

Prvé QA systémy vznikli už v 60. a 70. rokoch 20. storočia. Vývoj QA systémov môžeme rozdeliť na niekoľko fáz, pričom každý krok priniesol nové prístupy a technológie.

Prvé QA systémy boli založené na pravidlových systémoch. Jedným z prvých bol systém BASEBALL, ktorý odpovedal na otázky formulované v bežnej angličtine na základe uložených údajov o baseballových zápasoch. Využíval syntaktickú a obsahovú analýzu na získanie a spracovanie relevantných informácií [16]. Podobne, systém LUNAR bol vyvinutý na poskytovanie odpovedí na otázky týkajúce sa geologickej analýzy hornín zozbieraných počas misií Apollo [17]. Tieto systémy boli efektívne vo svojich špecifických doménach, ale ich schopnosť porozumieť širšiemu kontextu bola obmedzená.

V 90. rokoch začali QA systémy využívať techniky založené na informačnom vyhľa-

dávaní. Tieto systémy boli navrhnuté tak, aby na základe otázok formulovaných v prirodzenom jazyku identifikovali a extrahovali relevantné informácie z rozsiahlych textových korpusov. Jedným z významných systémov založených na informačnom vyhľadávaní je systém "START" vyvinutý tímom InfoLab Group pod vedením Borisa Katza, ktorý pôsobí na MIT. Tento webový QA systém poskytuje odpovede na otázky položené v prirodzenom jazyku prostredníctvom vyhľadávania a extrahovania informácií z rôznych online zdrojov [18].

S pokrokmi v oblasti strojového učenia a pokročilých neurónových sietí sa QA systémy stali sofistikovanejšími. Modely ako RNN, LSTM a GRU umožnili lepšie spracovanie sekvenčných dát a porozumenie kontextu. Pokrokové boli aj hybridné modely spájajúce viacero prístupov, ako napríklad IBM Watson. Známym sa stal v roku 2011, keď v rámci verejnej ukážky súťažil v televíznej relácii "Jeopardy!", kde porazil dvoch ľudských šampiónov. Watson využíva kombináciu prvkov IR, NLP a strojového učenia na analýzu otázok a vyhľadávanie odpovedí z obrovského množstva textových dát. Mechanizmy pozornosti ďalej zlepšili schopnosť modelov zamerať sa na relevantné časti vstupu. Tieto technické aspekty sú popísané v kapitole 4.

V posledných rokoch priniesli revolúciu do oblasti QA systémy založené na architektúre Transformer, ako sú BERT a RoBERTa. Tieto modely využívajú mechanizmus self-attention, ktorý umožňuje paralelné spracovanie dát a efektívnejšie zachytávanie vzdialených súvislostí v texte. Ich schopnosť porozumieť širšiemu kontextu a generovať texty v prirodzenom jazyku výrazne prispela k zlepšeniu presnosti a celkového výkonu QA systémov. Podrobnejší opis týchto modelov je uvedený v kapitole 4.

3.2 Datasety v QA

Vo všeobecnosti sú datasety štrukturovanými kolekciami dát organizovanými a uloženými na jednom mieste za účelmi ďalšej analýzy a spracovania [19]. V kontexte extraktívnych QA systémov zvyčajne dataset predstavuje štrukturovaný súbor dát, v ktorom sa každá vzorka skladá z kontextu, otázok formulovaných v prirodzenom jazyku týkajúcich sa kontextu a odpovedí prislúchajúcich k otázkam. Ak sú odpovede v textovej forme, musia byť obsiahnuté ako súvislý podreťazec v príslušnom kontexte, alebo môžu byť reprezentované formou počiatočného a koncového indexu odpovede v príslušnom kontexte. Príkladmi takto štrukturovaných datasetov sú SQuAD 1.1, SQuAD 2.0, Natural Questions (short answers), NewsQA, BioASQ (faktografické otázky) a WikiQA. Ich spoločnou vlastnosťou je, že umožňujú modelu učiť sa lokalizovať správny úsek textu na základe

zadanej otázky.

Datasety sú primárne používané na trénovanie modelov a hodnotenie ich výkonnosti. Často sú čerpané z konkrétnych kontextových zdrojov, ako sú dokumenty, znalostné databázy alebo štruktúrované dátové súbory. Taktiež slúžia ako referenčné hodnotiace nástroje na porovnávanie výkonnosti rôznych modelov QA. Rôzne datasety sú navrhnuté tak, aby testovali odlišné aspekty QA systémov. Niektoré overujú schopnosť modelu odpovedať na otázky pokrývajúce široké spektrum tém, zatiaľ čo iné sa zameriavajú na schopnosť modelov interpretovať zložité alebo nejednoznačné otázky.

Jednou z hlavných výziev pri vývoji QA systémov je zabezpečenie dostatočného množstva kvalitných datasetov na trénovanie modelu. Hoci existuje viacero dobre známych datasetov, ako sú SQuAD 2.0, NewsQA, WikiQA, Natural Questions či BioASQ, ktoré boli spomenuté v predchádzajúcej časti, pre robustné trénovanie veľkých jazykových modelov (LLM) sú často potrebné podstatne väčšie objemy dát. Napriek tomu, že surových textových dát je dostupné obrovské množstvo, vytváranie anotovaných QA datasetov ručnou anotáciou je časovo náročné a nákladné. Z toho dôvodu narastá záujem o automatizované prístupy, ako je generovanie syntetických datasetov a automatizované anotovanie pomocou existujúcich modelov. Jedným z perspektívnych riešení je využitie generatívnych modelov, ktoré umožňujú automatizovane generovať kontexty, otázky aj odpovede .

Podobný prístup predstavuje rámec LIQUID [20], ktorý automatizuje tvorbu QA datasetov z neanotovaných textových korpusov. Na tvorbu otázok a odpovedí využíva kombináciu sumarizácie textu a extrakcie entít, čím efektívne nahrádza potrebu ručného anotovania. Tento prístup sa ukázal ako škálovateľný a ekonomicky výhodný, pričom umožňuje kontrolu nad kvalitou generovaných dát.

Práve takýto prístup automatizovaného generovania syntetických otázok a odpovedí bol skúmaný v rámci tejto práce. Pri jeho realizácii boli využité generatívne modely založené na architektúre Transformer, ktoré umožnili vytvoriť rozsiahly tréningový dataset bez potreby manuálnej anotácie. Významným prínosom je úspora času a nákladov spojených s tradičným spracovaním dát. Kľúčovým aspektom pri tvorbe datasetu však zostáva zabezpečenie jeho kvality. Kvalita dát bola v tomto prípade overená manuálnou kontrolou náhodne vybranej podmnožiny vygenerovaných dát. Tento overovací krok slúžil na identifikáciu prípadných chýb a odchýlok pri generovaní, a zároveň poskytol spätnú väzbu pre úpravu procesu generovania.

3.3 Dodatočná klasifikácia binárnych otázok

QA systémy sa stretávajú s rôznorodými typmi otázok, ktoré si vyžadujú odlišné prístupy pri generovaní odpovedí. Zatiaľ čo niektoré otázky vyžadujú voľnejšiu alebo opisnú odpoveď, iné očakávajú jednoznačné binárne odpovede typu Pravda/Nepravda. Mnohé QA modely však nedisponujú mechanizmom, ktorý by spoľahlivo rozlíšil, aký typ odpovede je v danom prípade požadovaný. To môže viesť k nevhodne formulovaným alebo irelevantným výstupom.

Na riešenie tohto problému bol v rámci tejto práce navrhnutý hybridný systém, v ktorom model okrem poskytnutia samotnej odpovede zároveň klasifikuje typ otázky. Rozlišované sú dva základné typy: (1) extraktívna otázka, ktorá očakáva odpoveď priamo extrahovanú z kontextu, a (2) binárna otázka, ktorá vyžaduje binárnu odpoveď `Pravda` alebo `Nepravda`.

Tento mechanizmus bol implementovaný pridaním dodatočnej klasifikačnej vrstvy, ktorá využitím špeciálneho výstupu tokenu `CLS]` klasifikuje otázky na základe dvoch výstupov:

- *is_bool* - rozhoduje či ide o extraktívnu alebo binárnu otázku
- *is_true* - určuje, či je odpoveď na binárnu otázku `True` alebo `False`

Na tieto výstupy je aplikovaná sigmoidálna aktivačná funkcia, po ktorej nasleduje rozhodovanie na základe pevne stanoveného prahu. Model sa tak naučí nielen rozlišovať medzi extraktívnymi a binárnymi otázkami, ale v prípade binárnej otázky aj rozpoznávať zodpovedajúcu binárnu odpoveď.

Zavedenie tejto klasifikačnej vrstvy prispieva k zvýšeniu presnosti a konzistencie odpovedí poskytovaných QA systémom. Umožňuje zvoliť optimálnejšiu formuláciu odpovede podľa charakteru vstupnej otázky, čím zvyšuje relevantnosť odpovedí a zároveň optimalizuje využitie výpočtových zdrojov. Tento prístup môže byť obzvlášť užitočný pri aplikáciách, kde je dôležitá konzistentnosť, interpretovateľnosť a štandardizovaný formát výstupov.

Kapitola 4

Vývoj sekvenčných modelov pre QA

Spracovanie prirodzeného jazyka (NLP) je oblasť umelej inteligencie, ktorá sa zaoberá automatizovaným spracovaním, analýzou a generovaním textu v prirodzenom jazyku. Pracuje teda so sekvenčnými údajmi, pri ktorých význam slov a fráz závisí od ich kontextu vo vete alebo texte. To je obzvlášť dôležité v systémoch question answering (QA), kde na vygenerovanie presnej odpovede modely musia pochopiť otázku aj súvisiaci kontext.

4.1 Prvé sekvenčné modely a ich obmedzenia

Prvé QA systémy boli založené na pravidlových alebo vyhľadávacích metódach, avšak tie neboli schopné efektívne pracovať s kontextom. Tento problém riešili neurónové sekvenčné modely, ako sú rekurentné neurónové siete (RNNs) a Long Short-Term Memory (LSTM). Tieto modely spracúvajú text postupne, pričom si udržiavajú informácie o predchádzajúcich slovách s cieľom zachytiť závislosti v texte.

Rekurentné neurónové siete (RNN) majú svoj pôvod v práci Johna Hopfielda z rokov 1982 a 1984, v ktorej predstavil rekurentnú architektúru schopnú udržiavať vnútorný stav v čase (tzv. Hopfieldove siete) [21]. Neskôr boli predstavené ďalšie varianty, ako napríklad Jordanova sieť (1986) a Elmanova sieť (1990), ktoré sa začali využívať najmä v oblasti modelovania kognitívnych procesov a spracovania prirodzeného jazyka [22, 23].

Postupne sa RNN vyvinuli do podoby modelov určených na spracovanie sekvenčných údajov so schopnosťou zachytávať časové závislosti medzi vstupnými prvkami. Moderné formulácie RNN definujú výpočet skrytého stavu ako rekurzívnu funkciu závisiacu od aktuálneho vstupu a predchádzajúceho skrytého stavu. Pri danom vstupe $X = (x_1, \ldots, x_T)$ sa skryté stavy počítajú podľa rekurzívneho vzťahu:

$$h_t = f(W_h\ h_{t-1} + W_x\ x_t + b),$$

kde:

- h_t je skrytý stav v čase t,
- W_h, W_x sú trénovateľné matice váh,
- b je bias,
- f je nelineárna funkcia (často hyperbolický tangens alebo ReLU).

Výstup v každom kroku je možné vypočítať nasledovným vzťahom:

$$y_t = W_y\ h_t + b_y.$$

Tieto siete dokážu zachytiť časové závislosti medzi slovami v čase, čo je obzvlášť užitočné pri spracovaní prirodzeného jazyka. Napriek tomu majú RNN svoje obmedzenia v podobe problému miznúcich a explodujúcich gradientov, ktoré sťažujú učenie dlhých sekvencií a znižujú efektívnosť propagácie informácií naprieč vzdialenými časovými krokmi.

Na prekonanie obmedzení základných rekurentných sietí boli navrhnuté LSTM (Long Short-Term Memory) siete [24]. LSTM siete obsahujú pamäťový blok, ktorý je riadený troma regulačnými mechanizmami, tzv. bránami:

1. Zabúdacia brána: Rozhoduje, ktoré informácie z predošlých stavov bunky môžu byť zabudnuté

   $$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f),$$

   kde σ je sigmoidálna aktivačná funkcia.

2. Vstupná brána: Určuje, ktoré nové informácie budú uložené do pamäte

   $$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i),$$

   $$\bar{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c).$$

3. Výstupná brána: Rozhoduje aká veľká časť aktualizovanej pamäte má byť zohľadnená v skrytom stave

   $$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o),$$

   $$h_t = o_t \odot tanh(c_t).$$

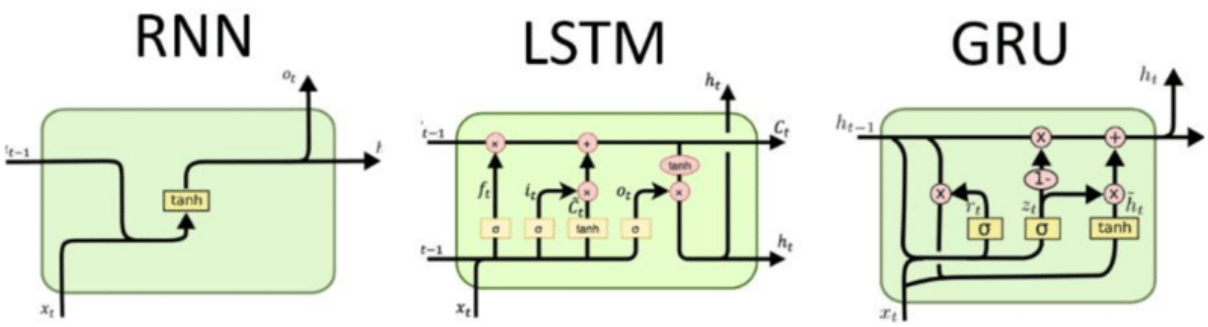
Aktualizácia stavu bunky sa vypočíta:

$$c_t = f_t \odot c_{t-1} + i_t \odot \bar{c}_t,$$

kde $\odot$ reprezentuje Hadamardov súčin.

Gated Recurrent Units (GRUs) [25] predstavujú zjednodušenú verziu LSTM, kde sú zabúdacia a vstupná brána zlúčené do jednej. Tým sa znižuje počet trénovateľných parametrov a zvyšuje sa efektivita učenia, pričom si model zachováva schopnosť pracovať so sekvenčnými dátami.

Aj napriek týmto vylepšeniam majú LSTM a GRU siete stále problémy s efektívnym zachytávaním vzdialených závislostí a obmedzenou paralelizáciou výpočtov. Tieto nedostatky viedli k vývoju mechanizmu pozornosti (attention mechanism) ako efektívnejšieho riešenia.



Obr. 4.1: Štruktúra RNN, LSTM a GRU (prevzaté z [26])

4.2 Predstavenie mechanizmu pozornosti

V sekvenčných úlohách typu sequence-to-sequence predstavuje skrytý stav s pevnou dĺžkou v RNN modeloch slabé miesto (bottle-neck), pretože núti model zakódovať všetky vstupné informácie do jediného vektora fixnej veľkosti. Tento prístup často vedie k strate informácií, najmä pri spracovaní dlhších textov. Dôvodom je, že model môže počas dekódovania „zabudnúť“ dôležité informácie z úvodu vstupu, hoci môžu byť kľúčové pre správne generovanie výstupu.

Na prekonanie týchto obmedzení je nevyhnutné, aby modely dokázali dynamicky sústrediť pozornosť na rôzne časti vstupu, namiesto toho, aby sa spoliehali na jeden komprimovaný vektor. Túto potrebu adresuje koncept mechanizmu pozornosti (attention mechanism), ktorý umožňuje modelom priraďovať rôznu dôležitosť jednotlivým častiam textu

(tokenom). Vďaka tomu dokážu zachovávať bohatší kontext a vykonávať sofistikovanejšie rozhodovanie počas dekódovania.

Jeden z prvých, a zároveň najznámejších prístupov, k implementácii pozornosti predstavili Bahdanau et al. v roku 2014 [27]. Tento mechanizmus zavádza výpočet kontextového vektora c_t ako vážený súčet skrytých stavov vstupnej sekvencie h_i, pre každý vstupný krok t, pričom váhy sú závislé od skrytého stavu dekodéra s_{t-1}:

$$e_{ti} = v^T \ tanh(W_h \ h_i + W_s \ s_{t-1}).$$

Vypočítané skóre e_{ti} je následne normalizované pomocou aktivačnej funkcie softmax, čím sa získajú váhy pozornosti α_{ti}:

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_j exp(e_{tj})}.$$

Finálny kontextový vektor sa potom vypočíta ako vážený súčet skrytých stavov:

$$c_t = \sum_i \alpha_{ti} h_i.$$

Tento kontextový vektor následne slúži ako vstup pre generovanie výstupu v kroku t.

Bahdanau-ov mechanizmus pozornosti (additive attention) [27] výrazne zlepšuje výkon modelov typu encoder-decoder tým, že eliminuje potrebu komprimovať celý vstup do jediného vektora. Umožňuje modelu dynamicky sa sústrediť na relevantné časti vstupu pri každom dekódovacom kroku, čím podporuje presnejšie spracovanie dlhších textov. Tento prístup výrazne posilňuje schopnosť modelu zachytávať vzdialené závislosti a zohľadňovať informácie z rôznych častí vstupu podľa ich dôležitosti pre aktuálny vstup. Okrem toho zlepšuje interpretovateľnosť modelu, keďže váhy pozornosti poskytujú prehľad o tom, ktoré časti vstupu mali najväčší vplyv na výstup.

Luong et al. [28] navrhli výpočtovo efektívnejší mechanizmus multiplikatívnej pozornosti (multiplicative attention), kde je skóre vypočítané pomocou skalárneho súčinu (dot product):

$$e_{ti} = s_{t-1}^T \ W \ h_i.$$

Na rozdiel od Bahdanauovej pozornosti, kde sa model učí vektor váh v, táto metóda využíva maticové násobenie, čím sa znižuje výpočtová náročnosť.

Mechanizmy pozornosti významne prispeli k zlepšeniu modelov QA tým, že umožnili selektívne sústredenie sa na relevantné časti textu, namiesto spoliehania sa na pevne zakódovanú reprezentáciu kontextu. Model BiDAF [29] využíva mechanizmus pozornosti

na zarovnanie reprezentácií otázky a kontextu, čím zvyšuje presnosť lokalizácie odpovede. Ďalší model DrQA [30] implementuje pozornosť na efektívne vyhľadávanie a extrakciu odpovedí z dokumentov. Tieto pokroky vytvorili základ pre modely založené na architektúre Transformer, ktoré mechanizmus pozornosti ďalej zdokonalili zavedením konceptu sebapozornosti (self-attention); mechanizmu, ktorý umožňuje modelu zachytiť vzťahy medzi všetkými slovami vo vstupe bez ohľadu na ich vzdialenosť.

4.3 Od mechanizmu pozornosti k Transformerom

Hoci mechanizmy pozornosti zmiernili problém pevnej dĺžky skrytého stavu v modeloch založených na rekurentných neurónových sieťach (RNNs), tieto prístupy boli naďalej závislé od rekurentného spracovania, čo obmedzovalo ich efektivitu. Sekvenčné spracovanie bránilo paralelizácii výpočtov, spomaľovalo trénovanie a sťažovalo zachytávanie vzdialených závislostí v texte. Tieto výzvy viedli k vývoju architektúry Transformer [31], ktorá úplne odstránila rekurenciu a nahradila ju mechanizmom sebapozornosti (self-attention). Výsledkom bol zásadný posun v spracovaní prirodzeného jazyka - modely majú lepšiu výkonnosť, lepšie zachytávajú globálny kontext a trénovanie je výrazne rýchlejšie vďaka paralelizácii výpočtov.

Na rozdiel od klasického mechanizmu pozornosti, ktorý sa využíva na prepojenie reprezentácií enkodéra a dekodéra, mechanizmus self-attention umožňuje modelu zachytávať vzťahy medzi všetkými tokenmi v rámci jednej vstupnej sekvencie. Vďaka tomu môže každý token adaptívne zohľadniť informácie z ostatných pozícií a vytvoriť kontextovo závislú reprezentáciu.

Základným výpočtovým prvkom je škálovaná skalárna súčinová pozornosť (scaled dot-product attention), definovaná nasledovne:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{4.1}$$

Pre pochopenie výpočtu pozornosti je potrebné pochopiť čiastkové kroky. Prvým krokom je získať matice Q, K a V. Pre vstupnú sekvenciu X dĺžky n s embedovacími vektormi dimenzie d_{model} sa tieto matice vypočítajú ako lineárne transformácie využitím trénovateľných parametrov: $Q = XW^Q, \quad K = XW^K, \quad V = XW^V,$

kde:

- $X \in \mathbb{R}^{n \times d_{model}}$ je vstupná sekvencia (napr. výstup z predchádzajúcej vrstvy),
- $W^Q, W^K, W^V \in \mathbb{R}^{d_{model} \times d_k}$ sú trénovateľné váhové matice.

Matice Q, K a V majú nasledujúce úlohy:

- Q (queries) – reprezentujú dotazy určujúce, na ktoré časti sekvencie by mal daný token upriamiť svoju pozornosť. Každý token vo vstupnej sekvencii je lineárne transformovaný na query vektor vyjadrujúci, aký druh informácie hľadá,
- K (keys) – prestavujú informáciu, ktorú tokeny ponúkajú na porovnanie s query, t.j. každý token je zároveň potenciálnou odpoveďou na otázku iného tokenu. Kľúče sa používajú pri výpočte podobnosti medzi tokenmi,
- V (values) – obsahuje skutočné informácie, ktoré sú agregované na základe vypočítanej pozornosti.

Následne sa vypočíta skalárny súčin medzi Q a K^T, čím vznikne matica skóre pozornosti pre všetky dvojice tokenov v sekvencii:

$$QK^T \in \mathbb{R}^{n \times n}. \tag{4.2}$$

Pretože vyššie dimenzie môžu viesť k príliš vysokým hodnotám skóre (4.2), čo spôsobuje numerickú nestabilitu pri aplikácii softmax funkcie, výsledné skóre sú škálované faktorom $\sqrt{d_k}$:

$$\frac{QK^T}{\sqrt{d_k}}.$$

Následne sa aplikuje aktivačná funkcia softmax, čím sa získané skóre transformujú na pravdepodobnostné rozdelenie, ktoré určuje váhu jednotlivých tokenov pre každý $q_i \in Q$. Nakoniec sa tieto váhy použijú na výpočet váženej sumy hodnôt V, čím vzniká výstupná reprezentácia (4.1).

Táto formulácia umožňuje paralelné spracovanie vstupných údajov a efektívne zachytáva vzdialené závislosti, čím adresuje hlavné obmedzenia RNN modelov. Každý token vo vstupnej sekvencii posudzuje význam ostatných tokenov na základe naučených skóre pozornosti. Tokeny relevantné pre aktuálnu pozíciu dostávajú vyššie váhy a vďaka tomu viac prispievajú do finálnej reprezentácie. Naopak, menej relevantné tokeny dostávajú nižšie váhy, čím sa minimalizuje ich vplyv. Tento mechanizmus odstraňuje sekvenčnú závislosť, umožňuje modelu spracovať všetky tokeny súbežne a tým výrazne zvyšuje výpočtovú efektivitu.

Pre zvýšenie vyjadrovacej schopnosti mechanizmu sebapozornosti (self-attention) zaviedli Vaswani et al. v pôvodnej práci [31] koncept viacnásobnej hlavy pozornosti (multi-head attention). Namiesto použitia jedinej sady váh pre výpočet pozornosti sa vstupy

paralelne premietajú do viacerých odlišných reprezentácií, pričom každá hlava (head) modeluje rôzne typy interakcií medzi tokenmi:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \ldots, head_h)W^O,$$

kde jednotlivé hlavy sú definované ako samostatné inštancie škálovanej skalárnej súčinovej pozornosti:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V).$$

Všetky váhové matice W_i^Q, W_i^K, W_i^V a výstupná projekcia W^O sú trénovateľné parametre modelu. Vďaka viacerým hlavám môže model súčasne zachytávať rôzne druhy vzťahov, napríklad syntaktické závislosti v jednej hlave a sémantické súvislosti v inej. Výsledkom je bohatšia reprezentácia každého tokenu, ktorá zohľadňuje viacero perspektív v rámci jednej vrstvy.

Keďže architektúra Transformer nepoužíva rekurenciu ani konvolučné operácie, chýba v nej prirodzený mechanizmus uchovávania informácie o poradí tokenov. Na adresovanie tejto slabiny bol zavedený koncept pozičného kódovania (positional encoding), ktorý explicitne pridáva informáciu o pozícii každého tokenu v sekvencii.

Pozičné kódy sa pripočítavajú ku vstupným vektorom tokenov pred prvou vrstvou modelu. Autori použili deterministickú formu kódovania založenú na sínusových a kosínusových funkciách s rôznymi frekvenciami:

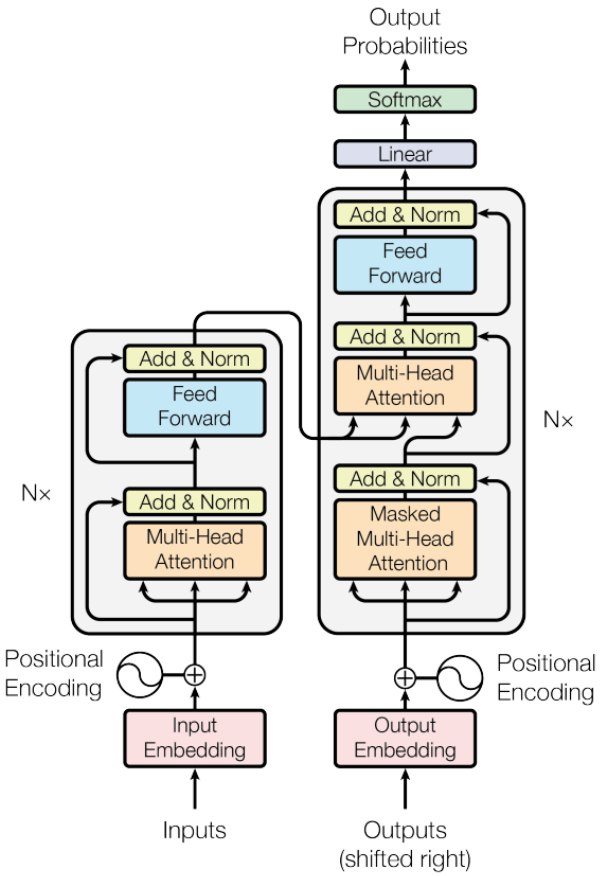
$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right),$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right),$$

kde pos označuje pozíciu tokenu v sekvencii a d_{model} predstavuje dimenziu vektorového priestoru embedovacích reprezentácií.

Použitie periodických funkcií zabezpečuje, že pozície sú kódované v hladkej, spojitej reprezentácii, ktorá umožňuje modelu odhadovať relatívne aj absolútne vzdialenosti medzi tokenmi. Vďaka tomu vie Transformer nielen rozpoznať poradie slov, ale aj zachytiť jemné vzory závislostí naprieč rôzne vzdialenými časťami textu.

Architektúra Transformer pozostáva zo štruktúry enkóder-dekóder, kde enkóder spracováva vstupnú sekvenciu a generuje kontextové reprezentácie, zatiaľ čo dekóder na ich základe postupne vytvára výstup, pričom zohľadňuje ako výstupy z enkódera, tak aj predchádzajúce generované tokeny.



Obr. 4.2: Architektúra modelu Transformer (prevzaté z [31])

Každá vrstva enkódera obsahuje self-attention mechanizmus, doprednú neurónovú sieť (feedforward network) a reziduálne prepojenia (residual connections), ktoré plynulým prechodom gradientov stabilizujú proces učenia. V dekóderi sa navyše využíva mechanizmus krížovej pozornosti (cross-attention), ktorý umožňuje modelu zamerať sa na relevantné časti výstupu z enkódera pri generovaní nasledujúceho tokenu. Vrstvenie týchto komponentov umožňuje Transformeru efektívne zachytávať kontextové informácie, vďaka čomu dosahuje veľmi dobré výsledky pri QA úlohách a ďalších aplikáciách v NLP.

Transformery sa vďaka svojej škálovateľnosti, paralelizovateľnosti a schopnosti pracovať s dlhými sekvenciami stali základom moderných jazykových modelov. Ich flexibilná architektúra umožnila vznik množstva predtrénovaných modelov, ktoré dosahujú špičkový výkon v širokom spektre NLP úloh.

Týmto sa otvára priestor na podrobnejší pohľad na proces predtrénovania týchto modelov a ich špecifiká, ktorým sa venuje nasledujúca podkapitola.

4.4 Predtrénovanie veľkých jazykových modelov

Rané modely spracovania prirodzeného jazyka (NLP) pracovali so statickými vektorovými reprezentáciami slov, ako napríklad Word2Vec [32] a GloVe [33]. Tieto modely mapovali slová na pevné vektory na základe ich štatistického výskytu v textoch. Hoci dokázali zachytiť základné sémantické podobnosti medzi slovami, každému slovu priraďovali len jedinú reprezentáciu, bez ohľadu na kontext, v ktorom sa vyskytovalo.

Tento zásadný nedostatok prekonali kontextové vektorové reprezentácie slov, ktoré umožňujú dynamicky meniť význam slov v závislosti od ich okolia (kontextu). Práve tieto reprezentácie stáli pri zrode predtrénovaných veľkých jazykových modelov (LLMs), ako sú ELMo [34], BERT [35] a GPT [36]. Vďaka spracovaniu slov v rámci celých viet sa tieto modely naučili hlboké jazykové reprezentácie, čím výrazne zvýšili úroveň porozumenia prirodzeného jazyka.

Modely LLM sú zvyčajne predtrénované na rozsiahlych korpusoch v režime self-supervised learning, teda bez potreby manuálne anotovaných dát. Vďaka tomu si dokážu osvojiť všeobecné jazykové vzory a znalosti, ktoré sa následne jemne dolaďujú (fine-tuning) na špecifické úlohy, ako napríklad úlohy (QA). Tento prechod od statických embeddingov k hlbokému predtrénovaniu predstavoval zásadný posun v oblasti NLP, ktorý výrazne zvýšil schopnosti modelov pri spracovaní zložitých jazykových úloh.

Proces predtrénovania sa najčastejšie realizuje dvoma hlavnými spôsobmi:

1. Autoregresívne jazykové modelovanie (jednosmerné, typické pre GPT),
2. Maskované jazykové modelovanie (obojstranné, typické pre BERT).

Autoregresívne jazykové modelovanie (GPT-štýl)

Autoregresívne modely predikujú nasledujúci token v sekvencii na základe predchádzajúcich tokenov. Pre danú sekvenciu $X = (x_1, \ldots, x_n)$ model maximalizuje pravdepodobnosť:

$$P(X) = \prod_{t=1}^{n} P(x_t | x_1, \ldots, x_{t-1}).$$

Tento prístup, použitý v modeloch ako GPT, umožňuje plynulé generovanie textu. Jeho hlavné obmedzenie spočíva v tom, že model nemá prístup k budúcim slovám, čo môže byť veľký nedostatok v úlohách, kde je potrebné porozumieť celkovému kontextu.

Maskované jazykové modelovanie (MLM)

Aby bolo možné modelovať obojsmerný kontext, bol zavedený prístup maskovaného jazykového modelovania (MLM). Namiesto predikcie ďalšieho tokenu model náhodne maskuje niektoré tokeny vo vstupe a učí sa ich rekonštruovať na základe okolitých slov. Cieľom je minimalizovať stratovú funkciu definovanú nad množinou maskovaných pozícií M:

$$L_{MLM} = -\sum_{i \in M} logP(x_i|X_{masked}),$$

kde: X_{masked} je vstupná sekvencia s maskovanými tokenmi.

Tento prístup, použitý v BERT, umožnil modelom naučiť sa hlboké obojsmerné reprezentácie založené na celkovom kontexte vety. Vďaka tomu sa výrazne zlepšila ich schopnosť porozumieť významovo komplexným vstupom.

Nasledujúca kapitola je zameraná na architektúru BERT, princípy jeho trénovania a vylepšenia, ktoré priniesli novšie modely ako RoBERTa a jeho viacjazyčné rozšírenia, napríklad XLM-R.

4.5 Od BERTa k RoBERTe: Pokroky v predtrénovaní Transformerov

BERT

Na éru statických embeddingov a autoregresívnych modelov nadviazal prelomový jazykový model BERT (Bidirectional Encoder Representations from Transformers), ktorý významne posunul hranice spracovania prirodzeného jazyka. BERT, predstavený Devlin et al. [35], ako prvý model umožnil hlboké obojsmerné učenie kontextu, čím výrazne zlepšil výkon naprieč širokým spektrom NLP úloh, vrátane úloh QA.

Na rozdiel od jednosmerných jazykových modelov, ktoré čítajú text buď zľava doprava alebo sprava doľava, BERT využíva maskované jazykové modelovanie (MLM), ktoré umožňuje učiť sa reprezentácie slov z oboch strán kontextu súčasne. Ako bolo opísané v predchádzajúcej podkapitole, model pri trénovaní náhodne maskuje niektoré tokeny vo vstupe a snaží sa ich predpovedať na základe ostatného okolitého kontextu. Tento prístup výrazne zvyšuje schopnosť modelu porozumieť významovým vzťahom medzi slovami.

Okrem maskovaného jazykového modelovania (MLM) BERT zahŕňa aj ďalší učebný cieľ, ktorým je predikcia nasledujúcej vety (Next Sentence Prediction, NSP). Účelom tohto predtrénovania nezávislého od konkrétnej úlohy je naučiť model zachytávať vzťahy medzi vetami, čo je kľúčové napríklad pri strojovom preklade, analýze textovej koherencie alebo

v systémoch QA. Počas trénovania model dostáva dvojicu viet S_1 a S_2 a jeho úlohou je určiť, či druhá veta v skutočnosti nadväzuje na prvú:

$$P(IsNext \mid S_1, S_2),$$

pričom v 50 % prípadov je S_2 skutočne nasledujúcou vetou, zatiaľ čo v zostávajúcich prípadoch je S_2 vybraná náhodne.

Kombináciou MLM a NSP dokázal BERT zachytiť komplexné obojsmerné kontextové vzťahy na úrovni slov aj viet, čím sa stal mimoriadne účinným riešením pre úlohy, kde kontext zohráva zásadnú úlohu.

RoBERTa: Optimalizácia tréningového procesu BERT

Hoci model BERT znamenal revolúciu v oblasti spracovania prirodzeného jazyka, neskoršie analýzy ukázali, že jeho potenciál extrahovať vzťahy počas tréningu nebol naplno využitý. Na túto skutočnosť upozornili Liu et al. [8] v práci RoBERTa: A Robustly Optimized BERT Pretraining Approach, kde demonštrovali, že zmenou tréningových nastavení je možné dosiahnuť výrazné zlepšenie výkonnosti modelu v širokej škále NLP úloh.

Model RoBERTa dosiahol lepšie výsledky než BERT najmä vďaka násobne väčšiemu množstvu tréningových dát, v ktorom boli zahrnuté aj rozsiahle korpusy ako Common Crawl a OpenWebText. Významnou zmenou bolo odstránenie samoúčebného cieľa NSP, keďže experimentálne výsledky ukázali, že tento cieľ neprináša merateľné zlepšenie v downstream úlohách. Namiesto toho sa RoBERTa sústredila výlučne na maskované jazykové modelovanie (MLM), čím sa zjednodušil tréning a zvýšila efektivita učenia.

Na rozdiel od statického maskovania použitého pri modeli BERT, RoBERTa využíva dynamické maskovanie, pri ktorom sa pozície tokenov maskovaných pri tréningu menia v každej epoche. Tento prístup znižuje pravdepodobnosť pretrénovania na konkrétnych pozíciách a vedie k lepšiemu zovšeobecneniu modelu. Ďalšími vylepšeniami boli väčšie veľkosti dávok (batch size až do 8000) a schopnosť spracovávať dlhšie vstupy až do 512 tokenov, čo posilnilo schopnosť modelu zachytávať širší kontext.

V dôsledku odstránenia úlohy NSP sa stratová funkcia RoBERTy zjednodušila na MLM základ:

$$L_{RoBERTa} = -\sum_{i \in M} logP(x_i \mid X_{masked}).$$

Táto optimalizácia umožnila modelu lepšie sa sústrediť na predikciu maskovaných tokenov a viedla k vyššej presnosti v mnohých NLP úlohách vrátane QA.

XLM-RoBERTa: Viacjazyčné rozšírenie RoBERTy

Zatiaľ čo RoBERTa priniesla významné vylepšenia v nastaveniach pre monolingválne úlohy, model XLM-RoBERTa (XLM-R), predstavený Conneau et al. [37], rozšíril tieto výhody aj na viacjazyčné úlohy NLP. XLM-R vychádza priamo z architektúry RoBERTa, no zavádza nový predtrénovací cieľ Cross-Lingual Masked Language Modeling (XLM-MLM), ktorý umožňuje modelu učiť sa jazykové reprezentácie naprieč stovkou rôznych jazykov.

Pri trénovaní XLM-R sa využíva rovnaký mechanizmus dynamického maskovania, avšak je aplikovaný na viacjazyčné korpusy, napr. CC100. Model tak získava schopnosť budovať zdieľané jazykové reprezentácie, čo mu umožňuje prenášať znalosti medzi jazykmi. Stratová funkcia má tvar:

$$L_{XLM-R} = -\sum_{i \in M} logP(x_i | X_{masked}, L),$$

kde L predstavuje jazykový embedding, ktorý informuje model o jazyku, v akom sa daný vstup nachádza.

Vďaka predtrénovaniu na rozsiahlych viacjazyčných korpusoch, XLM-R dosahuje vynikajúce výsledky v úlohách QA aj v iných ako anglických jazykoch, a výrazne prekonáva BERTa aj RoBERTu. Jeho hlavnou prednosťou je schopnosť krížovej jazykovej generalizácie, teda prenosu znalostí medzi jazykmi bez potreby jazykovo špecifického dolaďovania (fine-tuning). To je obzvlášť cenné pri viacjazyčných QA systémoch a v prípadoch, keď nie je k dispozícii dostatok anotovaných dát pre každý jazyk.

Kapitola 5

Metodológia a implementácia

V tejto kapitole predstavíme postupnosť krokov potrebných na vytvorenie dátového súboru vhodného na trénovanie a validáciu QA modelov. Dataset bol budovaný podľa zásad popísaných certifikovaným tímom výskumníkov v dokumente „Dataset Definition Standard (DDS)“ [38].

Všetky kroky spracovania dát, popísané v nasledujúcich podkapitolách, boli realizované v anglickom, nemeckom a francúzskom jazyku. Operácie, ako generovanie prirodzeného textu, otázok a odpovedí, či anotácia dát, boli aplikované použitím príkazov (promptov) v prislúchajúcom jazyku. Okrem toho boli v rámci experimentovania vytvorené aj kombinované (cross-lingual) konfigurácie, v ktorých sa text nachádza v inom jazyku ako otázka k nemu prislúchajúca (napr. anglický text a nemecká otázka). Tento prístup umožňuje testovanie viacjazyčnej robustnosti modelu a je v súlade s cieľom podporovať multilingválne aplikácie QA.

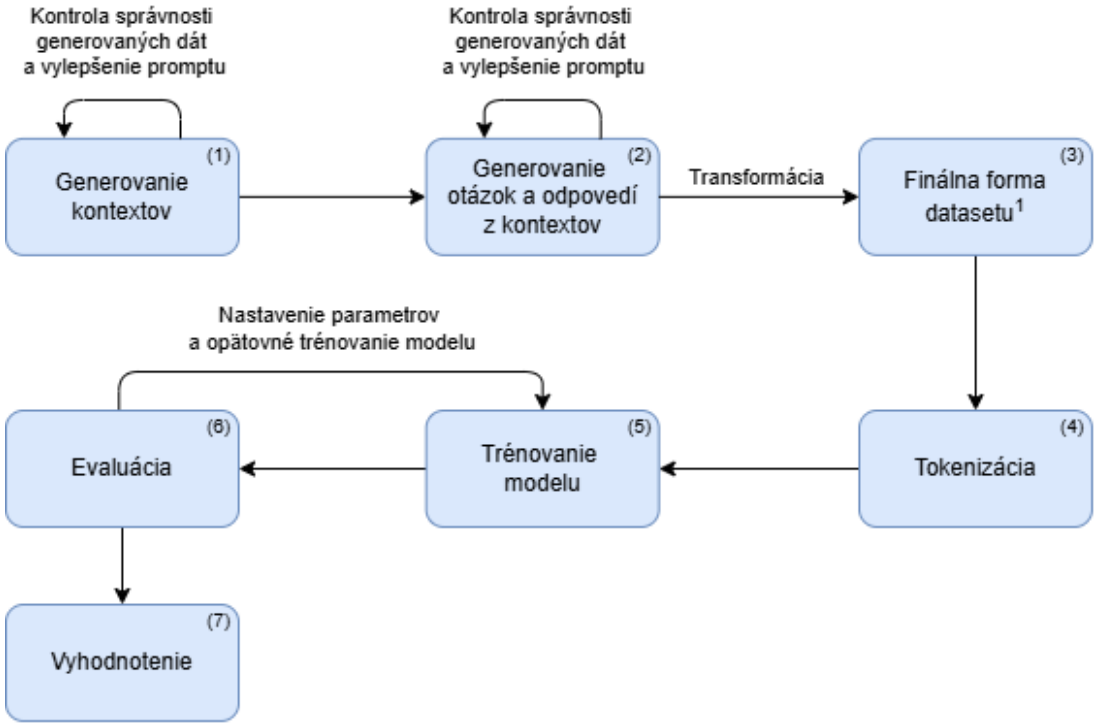
5.1 Popis základných dát

Dáta v tejto štúdii predstavujú objednávky na prepravu, ktoré sú štruktúrované v JSON formáte. Každá objednávka je reprezentovaná ako objekt, ktorý obsahuje kľúčové informácie o vyzdvihnutí, doručení a prepravených zásielkach.

Každá objednávka obsahuje nasledujúce hlavné sekcie:

1. **Pickup** - Obsahuje informácie o mieste, kde má byť tovar vyzdvihnutý. Zahŕňa názov spoločnosti, adresu, mesto, PSČ, krajinu a časový rámec, kedy sa má preprava uskutočniť.

2. **Delivery** - Obsahuje informácie o mieste doručenia tovaru. Podobne ako pri vyz-



Obr. 5.3: Postup implementácie

dvihnutí, aj tu sa nachádzajú údaje o názve spoločnosti, adrese, meste, PSČ, krajine a časovom rámci doručenia.

3. **Goods** - Táto sekcia poskytuje podrobnosti o tovare, ktorý je predmetom prepravy. Zahŕňa hmotnosť tovaru a zoznam balíkov s ich rozmermi (dĺžka, šírka, výška) a hmotnosťou.

4. **Stackable** - Tento údaj indikuje, či je možné na daný tovar ukladať ďalší tovar, čo značne ovplyvňuje spôsob, akým bude tovar naložený do vozidla a aký typ vozidla je potrebný.

5. **Required vehicle** - Uvádza požiadavky na typ vozidla, ktoré bude použité na prepravu tovaru, ako napríklad “Small lorry” (malý nákladný automobil).

6. **Special (vehicle) request** - Obsahuje špeciálne požiadavky na vybavenie vozidla, ako je požiadavka na použitie bočníc alebo iných špeciálnych zariadení.

Príklad štruktúry dát v JSON formáte vyzerá takto:

```
{
   "pickup": {
      "company_name": "COMPANY_PICKUP",
```

[1]Finálny dataset je zložený z kontextov a otázok a odpovedí ku kontextom

```
            "street_name": "STREET_PICKUP",
            "city": "CITY_PICKUP",
            "postal_code": "ZIP_PICKUP",
            "country": "COUNTRY_PICKUP",
            "datetime": "between 2021-03-04 13:30:00 and 2021-03-13 05:15:00"
        },
        "delivery": {
            "company_name": "COMPANY_DELIVERY",
            "street_name": "STREET_DELIVERY",
            "city": "CITY_DELIVERY",
            "postal_code": "ZIP_DELIVERY",
            "country": "COUNTRY_DELIVERY",
            "datetime": "between 2021-03-17 13:45:00 and 2021-03-24 18:45:00"
        },
        "goods": {
            "weight": "92 kg",
            "dimensions": [
                {
                    "length": "96m",
                    "width": "85m",
                    "height": "12m",
                    "weight": "11 kg"
                },
                {
                    "length": "86m",
                    "width": "48m",
                    "height": "78m",
                    "weight": "12 kg"
                },
                {
                    "length": "46m",
                    "width": "58m",
                    "height": "19m",
                    "weight": "69 kg"
                }
            ]
        },
        "stackable": "no",
        "required vehicle": "Small lorry",
        "special (vehicle) request": "Sideboards required"
    }
```


Všetky číselné údaje v dátach boli generované náhodne v rámci definovaných rozsahov, čím sa zabezpečila ich variabilita a vďaka tomu zodpovedajú realite. Na druhej strane, textové údaje, ako názvy spoločností, adresy, krajiny a ďalšie detaily, sú zatiaľ reprezentované zástupnými hodnotami “COMPANY_PICKUP”, “STREET_PICKUP”, atď. Tieto zástupné hodnoty uľahčia modelu prepis do prirodzeného textu a generovanie odpovedí na otázky, ako uvidíme v ďalších podkapitolách. Samozrejme, po finalizácii štruktúry datasetu budú nahradené náhodne vybratými hodnotami z vopred pripravenej množiny údajov. Každý záznam o objednávke sa riadi rovnakým formátom, čo zaručuje jednotnosť dát naprieč celou dátovou množinou a zároveň konzistenciu pri spracovaní a analýze.

Všetky tieto informácie sú následne použité na generovanie textu v prirodzenom jazyku, ktorý zhrňuje objednávku a je vhodný na použitie v rôznych aplikáciách, ako sú dopravné systémy alebo zákaznícke služby.

Štruktúra týchto dát umožňuje jednoduchú manipuláciu a analýzu, pričom poskytuje flexibilitu pre ďalšie spracovanie a prispôsobenie podľa potrieb.

5.2 Prevod základných dát do prirodzeného jazyka

Prevod štruktúrovaných dát do prirodzeného jazyka (pozri obr. 5.3 (1)) bol realizovaný pomocou jazykového modelu Llama2. Cieľom tejto transformácie bolo simulovať objednávky písané ľuďmi tak, aby pripomínali voľný konverzačný štýl namiesto štruktúrovaných dát. Presné údaje sa stále nachádzajú v texte, ale sú uvedené do formy súvislých viet, čo robí text čitateľnejším a prirodzenejším. Pri generovaní sme skúšali viacero rôznych príkazov, ktoré sme následne vylepšovali a upravovali na základe našich požiadaviek. Každý prompt bol navrhnutý tak, aby zachovával logickú štruktúru objednávok, ale zároveň modelu nechával aj istú flexibilitu pri tvorbe textov.

Nakoniec sme sa rozhodli pre použitie troch rôznych promptov, ktoré sú navrhnuté tak, aby zabezpečili rôzne štýly a čo najväčšiu rozmanitosť výstupných generovaných textov. Prvý prompt bol zameraný na vytvorenie textu čo najviac podobného ľudsky napísanému príbehu, s dôrazom na čitateľnosť a konverzačný tón. Druhý prompt obsahoval mierne upravený jazyk, pričom sme ponechali určitú voľnosť v štýle textu (vrátane použitia tabuliek alebo odstavcov pre rozdelenie údajov). Tretí prompt bol experimentálny, kde sme použili HTML prvky (ako <table>, <tr>, <td>, a ďalšie) na zlepšenie čitateľnosti a formátovania textu, čo poskytlo ešte väčšiu flexibilitu pri generovaní textu. Výsledkom týchto prístupov sú vygenerované texty vhodné na ďalšie využitie pri generovaní otázok a odpovedí v nasledujúcich krokoch procesu.

Prompty:

```
prompt1 = f"""
    {sample}
    Make natural language text from this JSON-structured transportation order.
    It should be as similar as possible to no structured (no table), human-
        written text (narrative style, aiming for readability and a~
        conversational tone to make the text stretch).
    If it is necessary for readability, use table form for dimensions value.
    Please make sure you include in the text the pickup and delivery time (not
        just the date)!
    It is the text that I will send to the forwarding company {np.random.choice(
        companies)}.
    """

prompt2 = f"""
    {sample}
    Make natural language text from this JSON-structured transportation order.
    It should be as similar as possible human-written text (narrative style,
        aiming for readability and a~conversational tone to make the text stretch
        ).
    If it is necessary for readability, you can use table/paragraph style.
    Please make sure you include in the text the pickup and delivery time (not
        just the date)!
    It is the text that I will send to the forwarding company {np.random.choice(
        companies)}.
    """

prompt3 = f"""
    {sample}
    Make natural language text from this JSON-structured transportation order.
    Add conversational tone to make the text stretch.
    Use html elements for improving readability, you can use html elements (<
        table>, <tr>, <th>, <td>, <h1> - <h6>, <br> and so~on...)
```

```
        Please make sure you include in the text the pickup and delivery time (not
            just the date)!
        It is the text that I will send to the forwarding company {np.random.choice(
            companies)}.
        """
```

Príklad výstupného textu:

```
    Sure! Here's~the natural language text based on the provided JSON-structured
        transportation order:

    "We have a~transportation order for you! We need to pick up some goods from
        COMPANY_PICKUP at STREET_PICKUP in CITY_PICKUP, ZIP_PICKUP, COUNTRY_PICKUP
        between 2021-03-04 13:30:00 and 2021-03-13 05:15:00. We'll be using a~Small
        lorry to transport these goods, which include three packages with the
        following dimensions:
    * Package 1: Length 96m, Width 85m, Height 12m, Weight 11kg
    * Package 2: Length 86m, Width 48m, Height 78m, Weight 12kg
    * Package 3: Length 46m, Width 58m, Height 19m, Weight 69kg

    We'll deliver these goods to COMPANY_DELIVERY at STREET_DELIVERY in CITY_DELIVERY
        , ZIP_DELIVERY, COUNTRY_DELIVERY between 2021-03-17 13:45:00 and 2021-03-24
        18:45:00. We've been requested to provide sideboards for the delivery, so~
        please make sure to include those in the transportation plan.

    The total weight of these goods is 92kg, and we'll need a~Small lorry to
        transport them safely. Please let us know if you have any questions or
        concerns about this order."

    I hope this text meets your needs! Let me know if there's~anything else I can do~
        for you.
```

Môžeme si všimnúť, že model okrem požadovanej odpovede poskytuje aj sprievodný text, ktorý sa nedá úplne eliminovať a je charakteristickou črtou chatbotov.

5.3 Generovanie otázok a odpovedí

Generovanie otázok a odpovedí (pozri obr. 5.3 (2)) bolo vykonané na textoch, ktoré vznikli v predchádzajúcej kapitole, v rámci ktorej sme previedli štruktúrované dáta do prirodzeného jazyka. Tento proces spočíval v experimentovaní s rôznymi príkazmi, ktoré sme opakovane upravovali a zdokonaľovali, aby sme dosiahli požadované výsledky. Rozhodli sme sa pre generovanie troch rôznych typov otázok: otázky s textovými odpoveďami, otázky s číselnými odpoveďami a otázky typu Pravda/Nepravda. Pre každý typ otázky sme vytvorili špecifický prompt, ktorý bol navrhnutý tak, aby s čo najväčšou pravdepodobnosťou vrátil správne výsledky.

Prompt 1 – Generovanie otázok s textovými odpoveďami

Tento prompt bol navrhnutý na generovanie otázok, ktoré sa týkajú špecifických informácií v texte, pričom odpovede sú vyjadrené v textovej forme. Cieľom bolo zabezpečiť, aby sa otázky týkali faktických detailov a odpovede boli priamo extrahované z textu. Pripúšťame, aby odpovede zahŕňali aj číselné údaje, avšak musia obsahovať aj dodatočný text okrem čísla.

```
    prompt1 = f"""
```

```
    {text}
    Please generate 5 questions about the text provided and also povide answers
        for the questions.
    Questions MUST be designed so~that their answers can be directly identified
        within the text.
    So~please formulate answers exatly as they are written in the text.
    Return the questions with answers in JSON. Format should be:
    {{
        "qa": [
            {{
                "q": "...",
                "a": "..."
            }},
            {{
                "q": "...",
                "a": "..."
            }},
            ...
        ]
    }}
""",
```

Príklad výstupu:

```
"question": "Where are the goods being picked up from?"
"answer": "COMPANY_PICKUP at STREET_PICKUP in CITY_PICKUP, ZIP_PICKUP,
    COUNTRY_PICKUP"
```

Môžeme si všimnúť, že údaje o odosielateľskej spoločnosti sú stále reprezentované zástupnými hodnotami, čo zjednodušuje generovanie textu modelom.

Prompt 2 – Generovanie otázok s číselnými odpoveďami

Tento prompt bol špeciálne navrhnutý pre generovanie otázok, kde odpovede sú číselné hodnoty, ako sú dátumy, hmotnosti alebo rozmerové údaje, ktoré sa nachádzajú v texte.

```
prompt2 = f"""
    {text}
    Please generate 5 questions about the text provided and also povide answers
        for the questions.
    IMPORTANT: Questions have to be formulated in a~way that answers to them are
        numeric.
    So~please formulate answers exatly as they are written in the text.
    Return the questions with answers in JSON. Format should be:
    {{
        "qa": [
            {{
                "q": "...",
                "a": "..."
            }},
            {{
                "q": "...",
                "a": "..."
            }},
            ...
        ]
    }}
""",
```

Príklad výstupu:

```
"question": "What is the weight of the goods being transported?",
"answer": "92"
```

Môžeme si všimnúť, že odpoveď je iba číselná, bez dodatočného textu.

Prompt 3 – Generovanie otázok typu Áno/Nie

Tento prompt bol použitý na generovanie otázok typu Áno/Nie, kde odpoveď je binárna: buď "True"(pravda), alebo "False"(nepravda).

```
prompt3 = f"""
    {text}
    Please generate 5 questions about the text provided and also povide answers
        for the questions.
    IMPORTNANT: Make the questions so~that the answer to each one is True/False.
    Return the questions with answers in JSON. Format should be:
    {{
        "qa": [
            {{
                "q": "...",
                "a": "True/False"
            }},
            {{
                "q": "...",
                "a": "True/False"
            }},
            ...
        ]
    }}
""",
```

Príklad výstupu:

```
"question": "Are sideboards requested for the delivery?",
"answer": "True"
```

Keďže Llama2 je jazykový model, nemusí vždy dávať správne odpovede a nie je tomu inak ani v prípade nášho použitia. Model často vrátil neplatný formát JSON, čo znemožnilo extrahovanie JSON štruktúry z response modelu. Taktiež otázky niekedy nezodpovedali požadovanému typu zadanému v prompte, a preto sme museli implementovať dodatočné overenie typu odpovede. V niektorých prípadoch pri požadovaní textovej odpovede model vrátil odpoveď, ktorá sa nenachádzala v rovnakej formulácii v texte. Overenie typu odpovedí a najčastejšie chyby pri generovaní spolu s ich riešeniami boli nasledovné:

- Model vrátil neplatný formát JSON - opätovné generovanie
- Model vrátil textovú odpoveď, ktorá sa v danej formulácii nenachádza v texte - vylúčenie odpovede
- Model vrátil číselnú odpoveď s dodatočným textom, napr.“92kg” - v tejto situácii sme klasifikovali “92kg” ako textovú odpoveď a ako číselnú odpoveď sme ponechali “92”
- Model namiesto odpovede “True/False” vrátil odpoveď “Yes/No” - nahradenie odpovede “Yes/No” jednotným formátom “True/False”

5.4 Finálny formát datasetu

Táto podkapitola sa zameriava na finálnu úpravu formátu dát (pozri obr. 5.3 (3)) tak, aby boli vhodné ako vstup pre modely hlbokého učenia založené na architektúre

transformerov. Výstup generovaný modelom Llama2 obsahuje otázky a odpovede v jednoduchom formáte, kde každá otázka má priradenú odpoveď bez dodatočných metadát. Tento formát však nie je priamo kompatibilný s bežne používanými datasetmi pre trénovanie transformerových modelov, ktoré vyžadujú podrobnejšiu štruktúru obsahujúcu identifikátory otázok, presné textové odpovede, typ odpovede a index začiatku odpovede v texte.

Zároveň sme zástupné hodnoty nahradili náhodne vybranými hodnotami z vopred definovanej množiny údajov, keďže v tomto kroku už je nevyhnutné zachovať presné umiestnenie odpovedí v texte. Ak by sme túto zmenu vykonali neskôr, mohlo by dôjsť k posunu začiatkov odpovedí, čo by ovplyvnilo správnosť anotácií. Taktiež sme využili túto komplexnejšiu štruktúru dát na zoskupenie odpovedí, ktoré sme predtým rozdelili ako samostatné záznamy. Konkrétne išlo o prípady, kde odpoveď obsahovala číselný údaj spolu s dodatočným textom – v takýchto situáciách sme číselnú časť označili ako numerickú odpoveď a celý reťazec ponechali ako textovú odpoveď.

Na dosiahnutie požadovaného formátu bolo potrebné vykonať niekoľko transformácií:

- **Pridanie identifikátorov otázok:** Každej otázke bol priradený unikátny identifikátor, aby bolo možné jednotlivé záznamy jednoznačne identifikovať.
- **Obohatenie odpovedí o metadáta:** Odpovede boli rozšírené o dodatočné informácie, ako je ich typ (napr. “extractive” alebo “boolean”) a pozícia v pôvodnom texte (index začiatku odpovede).
- **Štruktúrovanie odpovedí:** Každá otázka bola upravená tak, aby odpoveď bola zapúzdrená v poli “answers”, čo je štandardný formát datasetov pre transformerové modely.

```
"qas": [
    {
        "id": "1",
        "question": "Where are the goods being picked up from?",
        "answers": [
            {
                "text": "Wuhan Iron & Steel at Apple 1969 in Durham, 48348,
                    Austria",
                "answer_type": "string",
            }
        ]
    },
    ...
    {
        "id": "7",
        "question": "What is the length of the first item?",
        "answers": [
            {
                "text": "32 meters",
                "answer_type": "string",
                "answer_start": 767
            },
```

```
                {
                    "text": "32",
                    "answer_type": "number",
                    "answer_start": 767
                }
            ]
        },
        ...
        {
            "id": "9",
            "question": "Are sideboards requested for the delivery?",
            "answers": [
                {
                    "text": "True",
                    "answer_type": "boolean"
                }
            ]
        },
        ...
    ]
```


5.5 Tokenizácia

Tokenizácia (pozri obr. 5.3 (4)) je kľúčovým počiatočným krokom v spracovaní prirodzeného jazyka (NLP), ktorý slúži na rozdelenie súvislého textu na menšie jednotky nazývané tokeny. V závislosti od zvolenej metódy môžu tokeny reprezentovať celé slová, podslová (subwords), jednotlivé znaky alebo celé vety. Tokeny predstavujú základné stavebné prvky pre modely (NLP), keďže umožňujú prevod textu do číselnej reprezentácie, ktorú sú modely hlbokého učenia schopné spracovať.

Medzi najpoužívanejšie prístupy k tokenizácii patria napríklad segmentácia podľa medzier, použitie regulárnych výrazov, tokenizácia na úrovni slov alebo znakov, ako aj segmentácia viet. V tejto práci sa ďalej podrobne venujeme tokenizácii na úrovni podslov (subword-level), ktorá je populárna pri tvorbe moderných jazykových modelov.

Pri spracovaní vstupu pre model BERT je použitý štandardizovaný formát, ktorý kombinuje otázku a kontext do jednej sekvencie pomocou špeciálnych tokenov. Vstup má nasledujúcu štruktúru:

[CLS] Otázka [SEP] Kontext [SEP]

Token [CLS] (classification token) sa vkladá na začiatok vstupu a slúži ako agregátor globálneho kontextu celej vstupnej sekvencie. Tokeny [SEP] slúžia ako oddeľovače medzi otázkou a kontextom a taktiež na konci sekvencie. Tento formát umožňuje modelu efektívne odlíšiť jednotlivé segmenty textu a správne vyhodnotiť ich vzájomný vzťah počas predikcie odpovede.

V architektúre BERT je tento typ tokenizácie implementovaný pomocou algoritmu WordPiece. Algoritmus WordPiece bol pôvodne predstavený v roku 2016 v práci Wu et al.

[39], pričom jeho prehľadný a optimalizovaný popis je dostupný v novšej práci Song et al. z roku 2021 [40]. Vytváranie slovnej zásoby vo WordPiece začína základným slovníkom znakov a postupne vytvára nové jednotky (subwords) spájaním najčastejšie sa vyskytujúcich susedných dvojíc tokenov na základe ich pravdepodobnosti zlepšenia modelovania jazyka. To umožňuje tokenizéru efektívne spracovať zriedkavé alebo neznáme slová ich rozložením na známe podslovné jednotky, čím sa zabezpečí úplné pokrytie vstupného textu s obmedzenou veľkosťou slovníka. Tento prístup vyvažuje kompaktnosť slovníka a reprezentačnú bohatosť.

Na rozdiel od jednoduchších metód, WordPiece umožňuje efektívne pracovať aj s neznámymi (OOV – out of vocabulary) alebo zriedkavými slovami. Ak sa celé slovo nenachádza v slovníku, algoritmus ho rozdelí na menšie známe podslovné jednotky, čo zaručuje úplné pokrytie textu aj pri obmedzenej veľkosti slovníka. Postup budovania slovnej zásoby algoritmom WordPiece je popísaný v Algoritme 1, ktorého koncept bol adaptovaný z [41].

Algorithm 1: Tréning WordPiece tokenizéra

Vstup : Tréningový korpus: množina slov $\mathcal{C} = \{w_1, w_2, \ldots, w_n\}$;
Maximálna veľkosť slovníka: $|\mathcal{V}|_{\max}$;

Výstup: Finálny subword slovník $\mathcal{V}$ a pravidlá tokenizácie;

`/* 1. Inicializácia slovníka so všetkými znakmi */`

1 $\mathcal{V} \leftarrow \{c \mid c$ sa vyskytuje v $\mathcal{C}\}$;

2 Inicializuj $\mathcal{V}$ ako slovník obsahujúci všetky znaky vyskytujúce sa v slovách $\mathcal{C}$;

`/* 2. Iteratívne zlúčenie dvojíc subwords */`

3 **while** $|\mathcal{V}| < |\mathcal{V}|_{max}$ **do**

4 Spočítaj výskyty všetkých dvojíc subwords v $\mathcal{C}$ podľa aktuálneho $\mathcal{V}$;

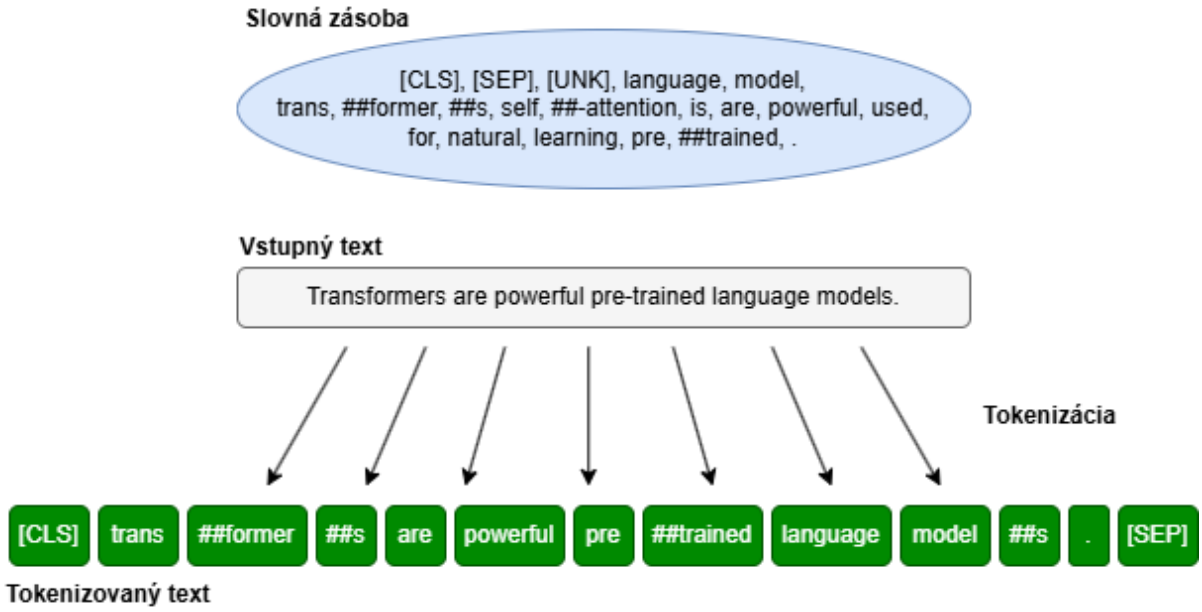
5 Vyber najčastejšiu dvojicu (a, b);

6 $\mathcal{V} \leftarrow \mathcal{V} \cup \{ab\}$ (Pridaj nový subword ab do $\mathcal{V}$);

7 **return** $\mathcal{V}$, *pravidlá tokenizácie pre nové texty*

Pri spracovaní textov presahujúcich maximálnu dĺžku vstupu podporovanú modelom BERT (512 tokenov) bol implementovaný prístup posuvného okna (sliding window). Tento prístup zabezpečuje, že tokeny presahujúce maximálnu dĺžku vstupu modelu (512 tokenov) nebudú pri predspracovaní orezané, čím sa predchádza strate informácií. Vstupný text bol rozdelený na prekrývajúce sa segmenty s veľkosťou okna 512 tokenov a prekrytím

256 tokenov medzi susednými segmentami. Prekrytie medzi oknami umožňuje zachovať kontext na hraniciach segmentov a zabezpečuje, že správna odpoveď bude prítomná v niektorom zo segmentov. Zároveň to znamená, že nie každý segment bude obsahovať správnu odpoveď, s čím sa model počas trénovania musí vysporiadať. V implementácii knižnice transformers je táto situácia štandardne ošetrená tak, že počas výpočtu straty sa zohľadňujú len segmenty obsahujúce správnu odpoveď.



Obr. 5.4: Príklad tokenizácie na úrovni podslov

5.6 Výber modelovej architektúry

Výber vhodného modelu predstavuje jeden z kľúčových krokov pri riešení úlohy QA. Hoci existuje nepreberné množstvo modelov, ktoré by túto úlohu dokázali zvládnuť so solídnymi výsledkami, je vhodné výber prispôsobiť nášmu scenáru použitia. V rámci tejto práce boli definované dve úlohy: monolingválne QA a multilingválne QA. Na ich riešenie sme starostlivo vybrali niekoľko modelov, ktoré sú vhodné na riešenie danej úlohy, pokrývajú rôzne veľkostné kategórie (malé, stredné a veľké – na základe počtu parametrov) a sú dostupné prostredníctvom knižnice HuggingFace Transformers.

Monolingválna úloha QA

Monolingválna QA úloha využíva podmnožinu datasetu vytvoreného v podkapitole 5.4, ktorá obsahuje výlučne anglické kontexty a otázky. Táto podmnožina zahŕňa 4 723 otázok. Keďže ide o dobre preskúmaný problém v oblasti spracovania prirodzeného

jazyka, v tejto časti sa zameriavame predovšetkým na adaptáciu existujúcich modelov na špecifický formát nášho datasetu.

Na riešenie tejto úlohy boli zvolené tri predtrénované modely, ktoré reprezentujú rôzne veľkostné triedy:

- **DistilBERT Base Uncased Distilled SQuAD** [42]
  Implementovaný ako `distilbert/distilbert-base-uncased-distilled-squad` v knižnici `transformers`.

  Ide o distillovaný variant BERT-base modelu, ktorý má približne 66,4 milióna parametrov, čo predstavuje zníženie o 40 % pri zachovaní viac než 95 % výkonu.

  Tokenizér: Využíva WordPiece tokenizáciu, ktorá rozkladá slová na subword jednotky na základe frekvencie výskytu. Je efektívna pri práci s OOV (Out of Vocabulary) slovami a zlepšuje kontextové porozumenie.

- **TinyRoBERTa SQuAD2.0** [43]
  Implementovaný ako `deepset/tinyroberta-squad2` v knižnici `transformers`.

  Táto distillovaná verzia modelu RoBERTa obsahuje približne 81,5 milióna parametrov a pri zachovaní porovnateľných výsledkov vo všetkých sledovaných metrikách dosahuje výrazne rýchlejší inferenčný čas.

  Tokenizér: Zdieľa rovnaký BPE tokenizér ako pôvodný RoBERTa, čím zabezpečuje kompatibilitu a efektivitu.

- **RoBERTa Base SQuAD2.0** [8]
  Implementovaný ako `deepset/roberta-base-squad2` v knižnici `transformers`.

  Model vychádza z architektúry RoBERTa-base, ktorá obsahuje približne 124 miliónov parametrov. Bol doladený na datasete SQuAD 2.0, ktorý obsahuje aj nezodpovedateľné otázky.

  Tokenizér: RoBERTa využíva Byte-Pair Encoding (BPE) tokenizáciu, ktorá rozkladá slová na časté subword jednotky, čím zlepšuje porozumenie aj pri zriedkavých alebo neznámych slovách.

Multilingválna úloha QA

Multilingválna úloha pokrýva kontexty a otázky v troch jazykoch – anglickom, nemeckom a francúzskom. Okrem extraktívnych otázok zahŕňa aj otázky binárneho typu. Dataset obsahuje 81 924 otázok, pričom 37 617 otázok je extraktívnych (odpoveď typu

string) a 44 307 otázok je binárnych. Keďže ide o zložitejšiu a menej preskúmanú úlohu, kládli sme dôraz na výber modelov, ktoré dokážu pracovať s rôznorodými jazykovými vstupmi. Pri výbere sme zohľadnili škálovateľnosť, výkonnosť a veľkostné kategórie modelov. Nasledujúce architektúry najlepšie spĺňali naše kritériá:

- **XtremeDistil l12 h384 Uncased [44]**
  Implementovaný ako `microsoft/xtremedistil-l12-h384-uncased` v knižnici `transformers`.

  Malý jazykový model s 33 miliónmi parametrov, ktorý vznikol distilovaním modelu XLM-R a použitím techniky transferu úloh (task-transfer). Jeho architektúra pozostáva z 12 vrstiev, skrytých stavov s rozmerom 384 a 12 hláv pozornosti. Má 2,7 krát rýchlejší inferenčný čas ako základný BERT a prekonáva ho takmer vo všetkých benchmarkových datasetoch, a to pri tretinovom počte parametrov.

  Tokenizér: Používa SentencePiece tokenizér s BPE slovnou zásobou, zdedený z rodičovského modelu XLM-R.

- **XLM-RoBERTa Base [45]**
  Implementovaný ako `FacebookAI/xlm-roberta-base` v knižnici `transformers`.

  Ide o základný variant multilingválneho modelu obsahujúci 279 miliónov parametrov. Jeho architektúra sa skladá z 12 vrstiev, skrytých stavov s rozmerom 768 a 8 hláv pozornosti. Model bol trénovaný na 2,5 TB dát z CommonCrawl v 100 jazykoch.

  Tokenizér: Používa SentencePiece tokenizer založený na Unigram Language Model, ktorý nevyžaduje predchádzajúcu segmentáciu slov.

- **XLM-RoBERTa Large [45]**
  Implementovaný ako `FacebookAI/xlm-roberta-large` v knižnici `transformers`. Rozšírená verzia Base modelu s približne 561 miliónmi parametrov. Jeho architektúra pozostáva z 24 vrstiev, skrytých stavov s rozmerom 1024 a 16 hláv pozornosti. Trénovaný na rovnakých multilingválnych dátach ako verzia Base.

  Tokenizér: Identický ako pri verzii Base – SentencePiece s Unigram LM.

Distilácia modelu [46] je technika prenosu znalostí (knowledge distillation), ktorá umožňuje vytvoriť menší a rýchlejší model (tzv. študent), ktorý si zachováva čo najviac výkonu pôvodného veľkého modelu (učiteľa). Počas trénovania sa študent neučí iba na základe správnych odpovedí (labelov), ale aj pomocou výstupov učiteľa, ktoré obsahujú

informácie o pravdepodobnostnom rozložení tried (tzv. soft targets). Týmto spôsobom sa dosahuje vyššia efektivita učenia a výsledný model má zvyčajne výrazne menej parametrov, kratší čas inferencie a menšie pamäťové nároky, pričom si zachováva značnú časť presnosti pôvodného modelu. Príkladmi takto vytvorených modelov sú napr. DistilBERT, ktorý má o 40 % menej parametrov než BERT-base, no dosahuje viac ako 95 % jeho výkonu.

5.7 Rozšírenie o klasifikačnú vrstvu

V rámci riešenia multilingválnej QA úlohy bol základný model doplnený o dodatočnú klasifikačnú vetvu, ktorej úlohou je predikcia typu otázky. Toto rozšírenie predstavuje vlastný prínos práce a nadväzuje na štruktúru datasetu, v ktorom sa vyskytujú tri hlavné typy otázok – extraktívne, číselné a binárne. Keďže odhaliť číselné otázky je pre jazykový model ľahká úloha (číselné odpovede obsahujú len číslice 0-9), sústredíme sa len na extraktívne a binárne otázky.

Návrh architektúry

Základom rozšírenia modelu je využitie výstupu špeciálneho tokenu `[CLS]`, ktorý sa pridáva na začiatok vstupnej sekvencie v architektúrach BERT a jeho derivátoch. Tento token po prechode celým transformerovým modelom vytvorí vlastnú reprezentáciu vstupnej sekvencie, v ktorej je kondenzovaný jej význam. Výstup `[CLS]` tokenu predstavuje vektor reálnych čísel $x_1, x_2, \ldots, x_n \in \mathbb{R}$, ktorý slúži ako globálna reprezentácia celej vstupnej vety.

Táto reprezentácia je následne odovzdaná do novopridanej plne prepojenej vrstvy s dvoma neurónmi, ktorá funguje ako klasifikačná hlava. Na výstupy tejto vrstvy sa aplikuje sigmoidálna aktivačná funkcia, čím získame dva výstupy:

- y_1 `(is_bool)` - pravdepodobnosť, že ide o binárnu otázku.

$$\textit{typ otázky} = \begin{cases} \text{extraktívna}, & \text{ak } y_1 \leq 0 \\ \text{binárna}, & \text{ak } y_1 > 0 \end{cases} \tag{5.1}$$

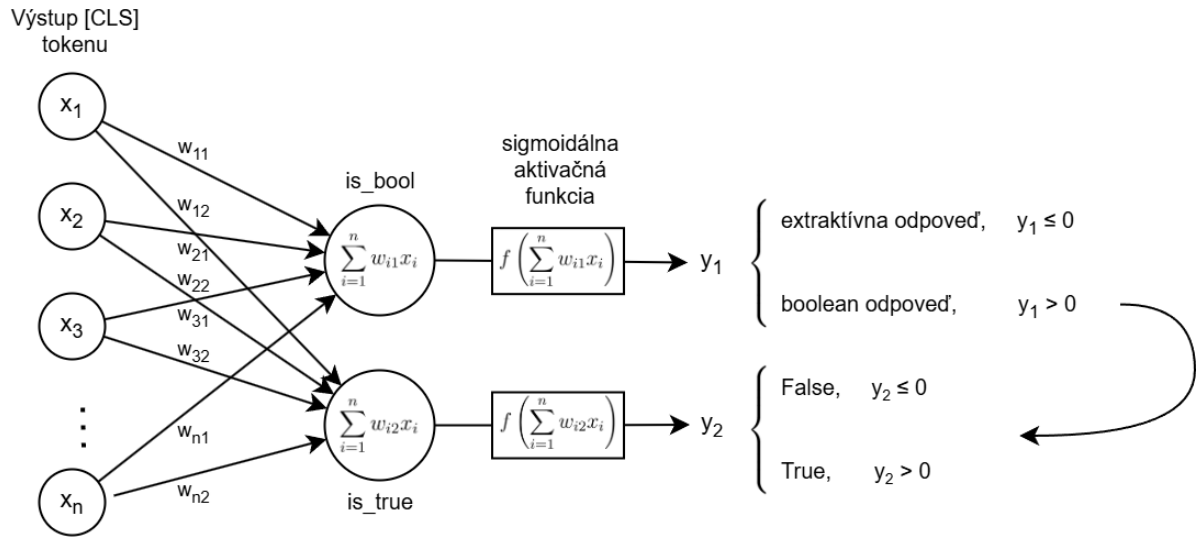
- y_2 `(is_true)` - pravdepodobnosť, že odpoveď na binárnu otázku je `True`.

$$\textit{binárna odpoveď} = \begin{cases} \texttt{False}, & \text{ak } y_2 \leq 0 \\ \texttt{True}, & \text{ak } y_2 > 0 \end{cases} \tag{5.2}$$

Výstup y_2 interpretujeme len v prípade, že predchádzajúca klasifikácia y_1 označí otázku ako binárnu.

Schému rozšírenia modelu o klasifikačnú hlavu ilustruje obrázok 5.5.

Tento klasifikačný mechanizmus je trénovaný súbežne s hlavným výstupom QA modelu, teda predikciou začiatku a konca odpovede v texte (start/end indexy). Optimalizácia modelu prebieha na základe kombinovanej stratovej funkcie, v ktorej každá zložka - extraktívna predikcia aj klasifikácia typu otázky - prispieva k celkovej hodnote straty. Tento prístup umožňuje modelu naučiť sa lokalizovať odpovede a zároveň určovať typ otázky v rámci jednej spoločnej architektúry.



Obr. 5.5: Diagram dodatočnej klasifikačnej vrstvy

Význam a prínos rozšírenia

Takýto prístup prináša niekoľko výhod. Predovšetkým zvyšuje robustnosť modelu, keďže umožňuje rozlíšiť, či otázka vyžaduje konkrétnu odpoveď z textu alebo binárnu odpoveď typu Pravda/Nepravda. To môže byť prínosom pri správnom určovaní štýlu odpovedania a spracovania otázky. Zároveň dochádza k unifikácii architektúry, pretože jediný model dokáže spracovávať rôzne typy otázok bez potreby rozvetvenia logiky na strane inferencie. Tento prístup tiež zlepšuje škálovateľnosť systému, keďže umožňuje jednoduchšie rozšírenie na ďalšie typy otázok v budúcnosti, ako sú napríklad generatívne odpovede alebo výber z viacerých možností. Rozšírenie bolo aplikované iba na multilingválne modely, keďže práve v tejto úlohe sa vyskytujú oba typy otázok. Monolingválne modely boli testované výhradne na extraktívnych otázkach, a preto takáto modifikácia v ich prípade nebola potrebná.

5.8 Nastavenie a trénovanie modelov

Táto podkapitola je venovaná popisu nastavení tréningových parametrov a analýze priebehu samotného tréningu (pozri obr. 5.3 (5)) pri experimentoch s modelmi popísanými v podkapitole 5.6. Pri návrhu tréningových nastavení sme sa sústredili na štyri kľúčové oblasti: parametre učenia (učiaci pomer, scheduler, warmup ratio), batch size a akumuláciu gradientov, regularizáciu (weight decay, dropout) a architektonické úpravy (zmrazenie vrstiev). Ostatné parametre tréningu boli ponechané na predvolených hodnotách odporúčaných knižnicou Hugging Face a budú spomenuté len stručne.

V druhej časti podkapitoly sa budeme venovať analýze priebehu učenia, konkrétne vývoju tréningovej a validačnej straty v čase, dĺžke tréningu jednotlivých modelov a identifikácii prípadných známok overfittingu alebo instability učenia.

5.8.1 Nastavenia tréningu a hyperparametrov

Monolingválna úloha

Každý z hyperparametrov uvedených v tabuľke 5.1 bol zvolený na základe kombinácie odporúčaní známych v oblasti trénovania transformerových modelov a vlastných experimentálnych testov. Kratší čas potrebný na trénovanie jednotlivých modelov nám umožnil vyskúšať viacero konfigurácií a následne vybrať také nastavenia, ktoré viedli k najlepšiemu výkonu na validačnej množine.

Stratová funkcia `eval_loss` použitá pri trénovaní modelov na extraktívnu QA je definovaná ako súčet dvoch kategoriálnych entropických stratových funkcií (*Cross-Entropy Loss*) pre predikciu začiatku a konca odpovede v texte. Formálne je daná vzťahom:

$$\text{Strata} = \text{CrossEntropy}(y_{\text{start}}, \hat{y}_{\text{start}}) + \text{CrossEntropy}(y_{\text{end}}, \hat{y}_{\text{end}}), \tag{5.3}$$

kde y_{start} a y_{end} sú skutočné indexy začiatku a konca odpovede, zatiaľ čo $\hat{y}_{\text{start}}$ a $\hat{y}_{\text{end}}$ sú pravdepodobnostné rozdelenia predikované modelom nad všetkými tokenmi v kontexte. Minimalizácia tejto stratovej funkcie vedie k zvýšeniu pravdepodobnosti správnej predikcie začiatku a konca odpovede.

Učiaci pomer bol nastavený v rozmedzí $1 \cdot 10^{-5}$ až $2 \cdot 10^{-5}$, pričom pre väčšie modely bola použitá nižšia hodnota na zabezpečenie stabilnejšej konvergencie. Ako scheduler bol zvolený `cosine` s použitím warmup fázy pokrývajúcej 10 % tréningových krokov, čo umožňuje postupné zvyšovanie učiaceho pomeru v počiatočných fázach tréningu a stabilizuje

proces optimalizácie.

Optimalizácia prebiehala pomocou algoritmu AdamW, ktorý je v súčasnosti považovaný za štandard pre trénovanie transformerových architektúr. Tento optimalizátor efektívne kombinuje adaptívne učenie s regularizáciou cez weight decay.

Batch size a akumulácia gradientov boli nastavené s ohľadom na dostupné pamäťové kapacity a požadovanú efektívnu batch size. V prípade väčších modelov s vyššími pamäťovými nárokmi bola použitá kombinácia menšieho batch size s akumuláciou gradientov. Menší batch size pri modeli `RoBERTa Base` bol použitý z dôvodu limitovanej pamäťovej kapacity servera a zároveň sa ukázalo, že priaznivo vplýval na tréning tohto modelu, keďže umožňoval častejšie aktualizácie váh a napomáhal lepšej generalizácii pri obmedzenom množstve tréningových dát.

Počet tréningových epoch bol nastavený na 10 pre menšie modely (`DistilBERT` a `TinyRoBERTa`) a na 5 pre `RoBERTa Base`. Tieto hodnoty predstavujú maximálny počet epoch, pričom tréning mohol skončiť skôr vďaka použitiu *early stopping*, ktorý ukončil učenie, ak sa validačná strata (`eval_loss`) nezlepšovala počas 3 po sebe nasledujúcich evaluácií. Modely boli evaluované priebežne po pevnom počte krokov (*steps*), čo umožnilo precíznejšie sledovanie priebehu učenia a zachytenie momentu, keď začalo dochádzať k preúčaniu. Táto stratégia prispela k efektívnejšiemu využívaniu výpočtových zdrojov a k lepšej generalizácii modelov.

Vzhľadom na obmedzenú veľkosť tréningovej množiny boli v modeloch zmrazené prvé dve až tri vrstvy, aby sa zachovali stabilné reprezentácie získané počas predtrénovania. Týmto opatrením sa minimalizovalo riziko preučenia na malom datasete a zároveň sa znížila výpočtová náročnosť tréningového procesu.

Dropout v skrytých vrstvách aj v attention mechanizmoch bol nastavený podľa veľkosti modelu. Menšie modely využívali nižšie hodnoty dropoutu (0,1–0,15), zatiaľ čo väčší `RoBERTa Base` bol trénovaný s vyšším dropoutom (0,2), aby sa zvýšila regularizácia a predišlo sa preučeniu.

Multilingválna úloha

Každý z hyperparametrov uvedených v tabuľke 5.2 bol zvolený na základe kombinácie osvedčených postupov v oblasti jemného doladenia predtrénovaných transformerových modelov a vlastných predbežných experimentov na multilingválnom datasete. Vďaka priebežnej evaluácii po pevných krokoch a nasadeniu mechanizmu *early stopping* sme mohli efektívne skúšať viacero konfigurácií a vybrať také nastavenia, ktoré priniesli najlepší kompromis medzi rýchlosťou konvergencie a generalizáciou.

Stratová funkcia `eval_loss` ostáva identická s tou definovanou vo vzťahu (5.8.1)

Parameter	`DistilBERT`	`TinyRoBERTa`	`RoBERTa Base`
Metrika evaluácie	eval_loss	eval_loss	eval_loss
Evaluačná stratégia	steps	steps	steps
Počet evaluačných krokov	50	50	100
Učiaci pomer	2e-5	1e-5	1e-5
Scheduler pre učiaci pomer	cosine	cosine	cosine
Warmup ratio	0,1	0,1	0,1
Optimalizátor	AdamW	AdamW	AdamW
Batch Size	16	16	8
Gradient accumulation	2	1	1
Effective batch size	32	16	8
Počet epoch	10	10	5
Early stopping	3	3	3
Weight decay	0,1	0,1	0,01
Zmrazené vrstvy	vrstvy 1-2	vrstvy 1-2	vrstvy 1-3
Dropout v skrytých vrstvách	0,2	0,2	0,2
Dropout v attention vrstvách	0,2	0,2	0,2

Tabuľka 5.1: Vybrané hyperparametre pre monolingválne modely

a meria súčet dvoch kategóriálnych entropických strát pre predikciu začiatku a konca odpovede. V multilingválnej úlohe sme ju zachovali ako hlavnú metriku pre výber najlepšieho modelu.

Učiaci pomer bol pri `XtremeDistil` a `XLM-RoBERTa Base` nastavený na $1 \cdot 10^{-5}$ a pri `XLM-RoBERTa Large` na $2 \cdot 10^{-5}$. Znížením veľkosti učiaceho pomeru sme úmyselne spomalili proces učenia, čím sme predišli príliš prudkým aktualizáciám váh a zabezpečili stabilnejšiu a jemnejšiu konvergenciu, najmä u väčších modelov. Ako scheduler sme použili `cosine` s warmup ratio 0,1, ktoré zodpovedá 10 % tréningových krokov a postupne uvoľňuje učiaci pomer v počiatočných fázach tréningu. Optimalizácia prebiehala pomocou štandardného algoritmu AdamW, rovnako ako v prípade monolingválnej úlohy.

Batch size a akumulácia gradientov boli zvolené s ohľadom na efektívne využitie kapacity GPU pamäte na serveri. Pre všetky modely bola zvolená kombinácia menších hodnôt `batch size` na zariadenie (8 alebo 16) a akumulácie gradientov (`gradient accumulation` = 2), čím sa dosiahla efektívna veľkosť dávky 16, resp. 32. Tento prístup umožnil trénovať aj väčšie modely ako `XLM-RoBERTa Large` bez prekročenia limitov pamäte, pričom zároveň

zabezpečil dostatočné množstvo dát na každú aktualizáciu váh a stabilitu optimalizačného procesu.

V prípade menšieho modelu `XtremeDistil` sme vedome zvolili menšiu efektívnu batch size (16), a to aj napriek tomu, že hardvérové obmedzenia by umožňovali väčší batch size. Menšia dávka umožňuje častejšie aktualizácie váh, čo môže byť výhodné pri trénovaní modelov s menším počtom parametrov na zložitejších úlohách alebo dátach s vyššou variabilitou, ako v prípade kombinovaných jazykových vstupov. Empiricky sa táto voľba ukázala ako výhodná z hľadiska priebehu straty aj generalizačných schopností modelu.

Maximálny počet epoch bol pre všetky modely stanovený na tri a evaluácia modelov prebiehala podľa stratégie *steps*, pričom pre `XtremeDistil` bola realizovaná každých 500 krokov a pre modely `XLM-RoBERTa Base` a `XLM-RoBERTa Large` každých 200, resp. 250 krokov. Takéto nastavenie umožnilo dôkladne sledovať priebeh učenia a včas detegovať náznaky preučenia, najmä pri modeli `XtremeDistil`, ktorý disponuje menším počtom parametrov a efektívne menšou veľkosťou dávky. Vo všetkých prípadoch bola použitá stratégia *early stopping*, ktorá ukončila trénovanie v prípade, že sa validačná strata nezlepšila v troch po sebe nasledujúcich hodnoteniach.

Vzhľadom na dostatočne veľkú trénovaciu množinu (približne 80 000 otázok) sme nepovažovali zmrazovanie vrstiev za potrebné, a preto boli všetky vrstvy modelov počas tréningu aktualizované.

Hodnoty dropoutu sme zvolili konzervatívne – 0,15 pre `XtremeDistil` a 0,1 pre oba XLM-RoBERTa modely – aby sme zachovali primeranú regularizáciu bez nadmerného potláčania naučených reprezentácií. Tieto hodnoty sú blízke predvoleným nastaveniam, čo sa osvedčilo pri prácach s rozsiahlymi datasetmi. Pri veľkosti nášho trénovacieho korpusu (cca 80 000 otázok) takéto mierne navýšenie dropoutu poskytuje dostatočnú ochranu proti preučeniu, pričom neznižuje schopnosť modelu efektívne zachytiť zložité jazykové vzťahy.

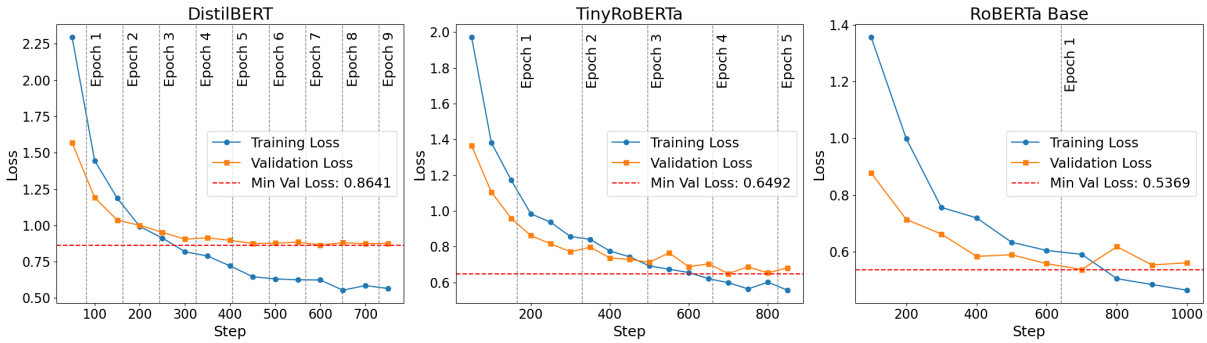
5.8.2 Priebeh a dĺžka tréningu

Monolingválna úloha

V tejto časti sme trénovali modely na anglických kontextoch a otázkach, pričom odpovede na všetky otázky boli extraktívneho typu. Preto budeme vyhodnocovať len extraktívne charakteristiky modelu. Na obrázku 5.6 sú zobrazené grafy priebehu tréningovej a validačnej straty pre jednotlivé modely na monolingválnej úlohe QA. V nasledujúcich odstavcoch popisujeme priebeh učenia a správanie každého modelu.

Parameter	`XtremeDistil`	`XLM-RoBERTa Base`	`XLM-RoBERTa Large`
Metrika evaluácie	eval_loss	eval_loss	eval_loss
Evaluačná stratégia	steps	steps	steps
Počet evaluačných krokov	500	200	250
Učiaci pomer	1e-5	1e-5	2e-5
Scheduler pre učiaci pomer	cosine	cosine	cosine
Warmup ratio	0,1	0,1	0,1
Optimalizátor	AdamW	AdamW	AdamW
Batch size (per device)	8	16	16
Gradient accumulation	2	2	2
Effective batch size	16	32	32
Počet epoch	3	3	3
Early stopping	3	3	3
Weight decay	0,01	0,01	0,01
Zmrazené vrstvy	žiadne	žiadne	žiadne
Dropout v skrytých vrstvách	0,15	0,1	0,1
Dropout v attention vrstvách	0,15	0,1	0,1

Tabuľka 5.2: Vybrané hyperparametre pre multilingválne modely



Obr. 5.6: Grafy tréningovej a evaluačnej straty pre monolingválnu úlohu QA

Priebeh tréningu modelu `DistilBERT` ukazuje, že tréningová strata plynulo klesala počas všetkých krokov a epôch. Validačná strata spočiatku nasledovala tento trend, avšak približne okolo kroku 300 až 400 sa stabilizovala a prestala sa výraznejšie znižovať. Po kroku 400 až 500 zostala validačná strata pomerne stabilná, s len miernymi osciláciami bez ďalšieho podstatného zlepšenia. Tento priebeh naznačuje, že `DistilBERT` sa učil

rýchlo, ale približne v polovici tréningu ďalšie aktualizácie váh viedli len k zlepšovaniu tréningovej straty pri veľmi malých zlepšeniach validačnej straty. Tento jav poukazuje na začínajúce preučenie modelu. Mechanizmus *early stopping* neukončil tréning, keďže nastávali veľmi malé zlepšenia pri validačnej chybe.

Model `TinyRoBERTa` vykazoval počas prvej polovice tréningu veľmi stabilné správanie. Tréningová aj validačná strata spoločne plynulo klesali, pričom trend validačnej straty veľmi blízko kopíroval priebeh tréningovej straty. Od druhej epochy začala validačná chyba oscilovať, pričom tréningová chyba ďalej plynulo klesala. Oscilácia validačnej straty znamená, že model sa stáva citlivý na malé zmeny v dátach, pretože začína prispôsobovať svoje váhy vzorom v tréningových dátach, ktoré nie sú reprezentatívne pre validačnú množinu. Inými slovami, model sa ešte nepreučuje, ale dostáva sa do fázy, kde stúpa riziko preučenia. Aj napriek fluktuáciám validačná chyba naďalej mierne klesala a v tréningu sme pokračovali ďalšie 3 epochy.

Pri modeli `RoBERTa Base` bolo možné pozorovať, že tréningová a validačná strata na začiatku tréningu klesali spoločne. Približne okolo kroku 700 až 800 sa však medzi nimi objavila viditeľná nezrovnalosť: zatiaľ čo tréningová strata pokračovala v klesajúcom trende, validačná strata stagnovala a prejavovali sa v nej viditeľné oscilácie. Napriek tomuto vývoju sa v závere tréningu obe krivky opäť priblížili a stabilizovali. Tento priebeh naznačuje, že `RoBERTa Base` sa v počiatočnej fáze učil veľmi efektívne, avšak vzhľadom na väčšiu modelovú kapacitu mal tendenciu k rýchlejšiemu preučeniu. Ku stabilizácii validačnej straty ku koncu učenia prispel kombinovaný účinok správne nastaveného učiaceho pomeru, zvoleného plánovania učenia (scheduleru), mierne zvýšeného dropoutu a vhodne nastaveného weight decay.

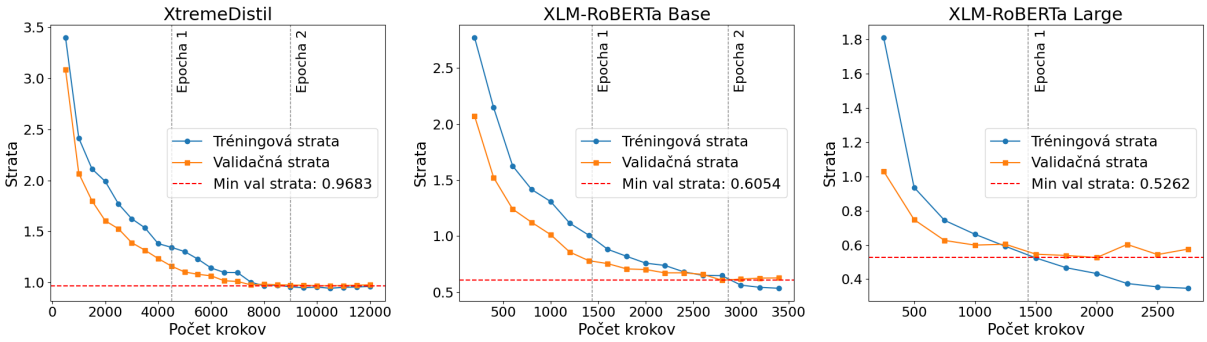
Tréning monolingválnych modelov bol časovo nenáročný vzhľadom na relatívne malé modely a veľkosť tréningovej množiny. Model `DistilBERT` bol natrénovaný za 6 minút a 18 sekúnd, `TinyRoBERTa` za 8 minút a 5 sekúnd a najväčší z tejto skupiny, `RoBERTa Base`, za 9 minút a 51 sekúnd. Mechanizmus *early_stopping* ukončil tréning, keď validačná strata začala stagnovať, konkrétne po 750 batchoch pre `DistilBERT`, 850 batchoch pre `TinyRoBERTa` a 1000 batchoch `RoBERTa Base`. Priebežné sledovanie tréningovej a validačnej straty ukázalo plynulý pokles a absenciu výraznejšieho overfittingu v prípade všetkých troch modelov. Výsledky priebehu učenia sú zobrazené na obrázku 5.6.

Model	Čas trénovania	Počet batchov	Počet dokončených epoch	Minimálna validačná strata
`DistilBERT`	6 min 18 s	750	9,26	0.8641
`TinyRoBERTa`	8 min 5 s	850	5,15	0.6492
`RoBERTa Base`	9 min 51 s	1000	1,56	0.5369

Tabuľka 5.3: Prehľad priebehu tréningu monolingválnych modelov

Multilingválna úloha

V tejto časti sme trénovali modely na multilingválnych kontextoch a otázkach, pričom dataset obsahoval kombináciu extraktívnych aj binárnych otázok. Na obrázku 5.7 sú zobrazené grafy priebehu tréningovej a validačnej straty pre jednotlivé modely na multilingválnej úlohe QA. V nasledujúcich odstavcoch popisujeme priebeh učenia a správanie každého modelu.



Obr. 5.7: Grafy tréningovej a evaluačnej straty pre multilingválnu úlohu QA

Priebeh tréningu modelu `XtremeDistil` ukazuje výrazný pokles tréningovej aj validačnej straty už počas prvej epochy. Straty plynulo klesali až do kroku 8000, kde sa hodnoty stabilizovali a ďalej klesali len mierne. Model dosiahol minimálnu validačnú stratu 0.9683. Tento priebeh naznačuje rýchlu konvergenciu v prvých fázach tréningu, čo je očakávané vzhľadom na menšiu veľkosť modelu a jednoduchšiu architektúru. Po stabilizácii validačnej straty sa model ďalej nezlepšoval, ale nevyskytli sa ani známky výraznejšieho overfittingu.

Model `XLM-RoBERTa Base` sa učil stabilne, pričom tréningová a validačná strata klesali spoločne počas prvých troch epoch. Od štvrtej epochy začala validačná strata mierne oscilovať, zatiaľ čo tréningová strata naďalej plynulo klesala. Tento jav naznačuje začína-

júce preúčanie, ktoré však bolo kompenzované mechanizmom *early stopping*. Minimálna validačná strata dosiahnutá modelom bola 0.6492, čo svedčí o dobrej schopnosti modelu prispôsobiť sa multilingválnemu zadaniu.

Pri modeli `XLM-RoBERTa Large` bolo možné pozorovať, že tréningová strata plynulo a stabilne klesala počas celého priebehu učenia, zatiaľ čo validačná strata dosiahla svoje minimum už v prvej polovici tréningu (konkrétne hodnota 0,5262) a následne začala mierne kolísať. Tento priebeh je typický pre veľké modely s vysokou kapacitou, ktoré sa rýchlo naučia vzory v tréningových dátach, avšak ich schopnosť generalizácie na validačné dáta môže byť obmedzená. Mierne oscilácie vo validačnej strate v neskorších fázach naznačujú začínajúce preučenie, ktoré však bolo do značnej miery stabilizované použitím vhodných regularizačných mechanizmov (dropout, weight decay) a aplikáciou stratégie *early stopping*.

Tréning multilingválnych modelov bol vzhľadom na väčšie architektúry časovo náročnejší než pri monolingválnej úlohe. Model `XtremeDistil` bol natrénovaný za približne 2 hodiny a 30 minút, `XLM-RoBERTa Base` za 2 hodiny a 22 minút a najväčší model `XLM-RoBERTa Large` za 4 hodín a 46 minút. Mierne predĺžený čas tréningu v prípade modelu `XtremeDistil` bol spôsobený častejšími evaluáciami (každých 500 krokov), ktoré boli zavedené s cieľom presnejšie monitorovať priebeh učenia. Tento prístup umožnil včas identifikovať moment, keď model začal vykazovať známky preučenia. V kombinácii s mechanizmom *early stopping* bolo možné ukončiť tréningový proces v optimálnom čase a efektívne predísť nadmernému využívaniu výpočtových zdrojov.

Môžeme pozorovať, že napriek vyššej výpočtovej náročnosti modelu `XLM-RoBERTa Large` nebol nárast času učenia priamo úmerný jeho veľkosti. Hoci jeho celkový čas tréningu bol približne dvojnásobný oproti základnej verzii modelu, konvergoval už pri približne 2750 krokoch, zatiaľ čo `XLM-RoBERTa Base` konvergoval až pri 3400 krokoch. Táto skutočnosť naznačuje, že väčšie modely môžu pri správnom nastavení hyperparametrov efektívnejšie využiť svoje reprezentančné schopnosti a dosiahnuť optimálny výkon s menším počtom aktualizácií váh.

Taktiež si môžeme všimnúť, že čím väčší model bol použitý, tým menej epoch bolo potrebných na dosiahnutie konvergencie - ako môžeme vidieť v tabuľke 5.4.

Model	**Čas trénovania**	**Počet batchov**	**Počet dokončených epoch**	**Minimálna validačná strata**
`XtremeDistil`	2 h 30 min	12000	2,66	0.8641
`XLM-RoBERTa Base`	2 h 22 min	3400	2,37	0.6054
`XLM-RoBERTa Large`	4 h 46 min	2750	1,92	0.5369

Tabuľka 5.4: Prehľad priebehu tréningu multilingválnych modelov

Kapitola 6

Evaluácia a vyhodnotenie

6.1 Evaluačné metriky

Pre objektívne vyhodnotenie výkonnosti navrhnutých modelov v úlohe QA je nevyhnutné zvoliť vhodné evaluačné metriky. Tie umožňujú kvantifikovať presnosť, úplnosť a celkovú spoľahlivosť odpovedí predikovaných modelom. V tejto práci využívame sadu metrík, ktoré sa aplikujú na dve odlišné úlohy – extraktívna QA a klasifikácia typu otázky. Hoci metriky ako Precision, Recall a F1-skóre sú využívané v oboch prípadoch, ich výpočtové formulácie a interpretačné rámce sa líšia v závislosti od charakteru danej úlohy.

6.1.1 Extraktívna QA úloha

V extraktívnej QA úlohe sa hodnotenie zakladá na porovnaní medzi predikovanou a referenčnou odpoveďou na úrovni tokenov. Vyhodnocované sú tieto metriky:

Exact Match (EM - presná zhoda)

Presná zhoda je binárna metrika špecifická pre extraktívne úlohy QA. Vyhodnocuje, či sa predikovaná odpoveď presne zhoduje s referenčnou odpoveďou po normalizácii (odstránení diakritiky, veľkých písmen a interpunkcie). Formálne je definovaná:

$$\mathrm{EM(x)} = \begin{cases} 1, & \text{ak } p = y \\ 0, & \text{inak} \end{cases} \tag{6.1}$$

kde p je predikovaná odpoveď a y je referenčná odpoveď.

Táto metrika je prísna a neakceptuje čiastočné zhody. Napriek tomu je často používaná, keďže odráža schopnosť modelu exaktne identifikovať odpoveď v texte.

Precision (Presnosť)

Precision vyjadruje podiel tokenov z predikcie, ktoré sú zároveň obsiahnuté v referenčnej odpovedi:

$$\text{Precision} = \frac{|\text{predikované} \cap \text{referenčné}|}{|\text{predikované}|} \tag{6.2}$$

Recall (Citlivosť)

Recall udáva podiel tokenov z referenčnej odpovede, ktoré sú obsiahnuté v predikcii:

$$\text{Recall} = \frac{|\text{predikované} \cap \text{referenčné}|}{|\text{referenčné}|} \tag{6.3}$$

F1-skóre (F1-score)

F1-skóre predstavuje harmonický priemer precision a recall:

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.4}$$

Na rozdiel od EM, F1-skóre zohľadňuje čiastočné zhody medzi odpoveďami, vďaka čomu poskytuje citlivejšie hodnotenie modelov pracujúcich s prirodzeným jazykom, kde môže existovať viacero ekvivalentných formulácií správnej odpovede.

6.1.2 Binárna QA úloha

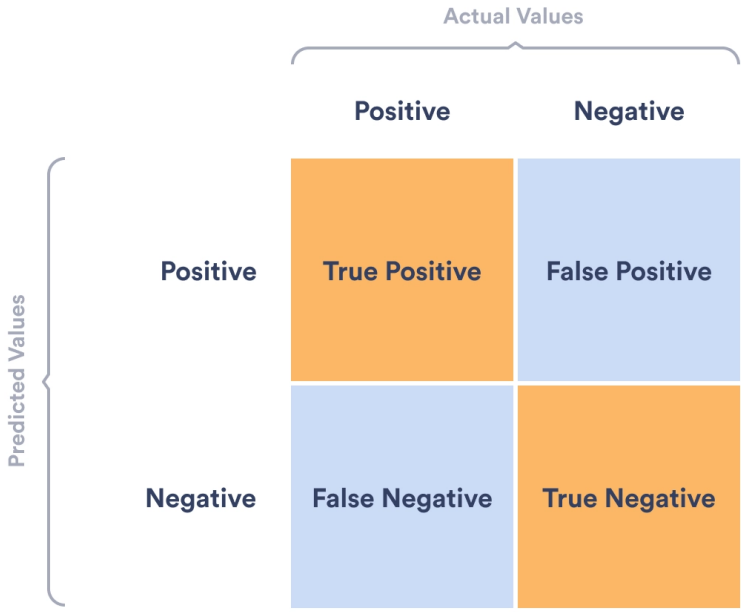
Pri klasifikácii typu otázky ide o štandardnú klasifikačnú úlohu, v ktorej model priradí každému príkladu predikovanú triedu. Využívajú sa nasledujúce metriky:

Matica zámeny (Confusion Matrix)

Matica zámeny predstavuje štandardný nástroj na analýzu výsledkov klasifikačných úloh. Ide o tabuľkové zobrazenie, v ktorom riadky reprezentujú skutočné triedy a stĺpce predstavujú triedy predikované modelom. Každá bunka matice C_{ij} vyjadruje počet prípadov, v ktorých boli príklady so skutočnou triedou i predikované ako trieda j.

V prípade binárnej klasifikácie má matica rozmery 2×2, a teda pozostáva zo štyroch základných charakteristík, ktoré sú použité na definovanie hodnotiacich metrík klasifikátora. Tieto štyri hodnoty sú:

- True Positives (TP) – správne predikované pozitívne príklady,
- False Positives (FP) – nesprávne predikované pozitívne príklady,
- False Negatives (FN) – nesprávne predikované negatívne príklady,
- True Negatives (TN) – správne predikované negatívne príklady.



Obr. 6.8: Matica zámeny pre binárnu klasifikáciu [47]

Accuracy (Správnosť klasifikácie)

Accuracy vyjadruje podiel správne predikovaných príkladov ku celkovému počtu príkladov:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{6.5}$$

Precision (Presnosť)

Precision vyjadruje počet správne klasifikovaných príkladov k celkovému počtu príkladov, ktoré model označil ako príslušné k danej triede:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.6}$$

Recall (Citlivosť)

Recall meria podiel správne identifikovaných pozitívnych prípadov k celkovému počtu skutočne pozitívnych prípadov:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.7}$$

F1-skóre (F1-score)

F1-skóre je definované rovnako ako vo vzorci (6.4).

Zvolená sada metrík umožňuje komplexné hodnotenie výkonu modelu v rôznych aspektoch. EM poskytuje konzervatívne, ale dôležité hodnotenie presnosti. F1-skóre zohľadňuje aj čiastočné zhody, čo je obzvlášť dôležité pri práci s prirodzeným jazykom.

Použitie rovnakých metrík v oboch častiach práce umožňuje porovnateľné hodnotenie výkonnosti modelov, pričom rešpektuje odlišný charakter hodnotených výstupov. Pri prezentácii výsledkov sú preto metriky explicitne oddelené podľa úlohy, v ktorej sú aplikované.

6.2 Vyhodnocovanie dlhých kontextov

Rovnako ako pri trénovaní, aj pri vyhodnocovaní modelov nastáva situácia, kedy dĺžka vstupného kontextu presahuje maximálnu kapacitu modelu. Tento limit je dôsledkom výpočtovej zložitosti self-attention mechanizmu, ktorá narastá kvadraticky s dĺžkou vstupu. Aby bolo možné vyhodnocovať dlhé texty bez straty informácie, je potrebné kontext vhodne segmentovať.

Tento problém bol v práci riešený už v kroku tvorby finálneho evaluačného datasetu (podkapitola 5.4), kde bola aplikovaná technika posuvného okna (sliding window). Text bol rozdelený na segmenty pevnej dĺžky 512 tokenov s definovaným prekryvom 256 tokenov medzi susednými oknami. Prekrytie zabezpečuje, že ak sa správna odpoveď nachádza na hranici segmentu, bude s vysokou pravdepodobnosťou zachytená aspoň v jednom z nich.

Počas evaluácie bol každý segment spracovaný modelom samostatne. Výstup modelu obsahoval

- skóre istoty modelu (confidence score), ktoré vyjadruje pravdepodobnosť správnosti predikcie,
- index začiatku odpovede (start_index),
- index konca odpovede (end_index),
- pravdepodobnosť, že ide o binárnu otázku (is_bool),
- pravdepodobnosť, že odpoveď na binárnu otázku je `True` (is_true).

Na základe výstupov zo všetkých segmentov bola vybraná odpoveď s najvyšším skóre istoty. Takýto postup umožňuje efektívne obísť obmedzenie maximálnej dĺžky vstupu bez potreby špeciálnych architektúr pre dlhé texty (napr. Longformer, BigBird), a zároveň zachováva konzistenciu s prístupom použitým počas trénovania.

6.3 Evaluácia a vyhodnotenie

Táto podkapitola sumarizuje výsledky dosiahnuté počas tréningu modelov na monolingválnej a multilingválnej úlohe QA. Uvádzame priebeh tréningovej a validačnej straty zaznamenaný počas tréningu, hodnoty evaluačných metrík, ako aj časovú náročnosť tréningu jednotlivých modelov. Podrobný opis dátových množín a použitých modelových architektúr je uvedený v podkapitole 5.6, zatiaľ čo nastavenia tréningových hyperparametrov sú popísané v podkapitole 5.8.

6.3.1 Monolingválna úloha

V monolingválnej úlohe sme riešili len extraktívnu úlohu QA, pri ktorej sme postupovali v súlade so štandardnými postupmi v tejto oblasti. Na základe pravdepodobnostného rozdelenia vypočítaného nad jednotlivými tokenmi vyberáme najpravdepodobnejší začiatok a koniec odpovede. Po určení rozsahu odpovede sú číselné reprezentácie tokenov transformované späť na ich slovné ekvivalenty. Úspešnosť predikcie odpovedí je následne vyhodnotená pomocou zvolených metrík, pričom dosiahnuté hodnoty sú uvedené v tabuľke 6.1.

Výsledky naznačujú, že všetky modely dosiahli relatívne vyvážené hodnoty precision a recall, čo naznačuje schopnosť nielen správne identifikovať odpovede, ale aj zachytiť väčšinu relevantných prípadov v dátach. S rastúcou veľkosťou a kapacitou modelov je možné pozorovať postupné zlepšovanie všetkých sledovaných metrík. Výraznejší nárast výkonu je viditeľný pri modeli `TinyRoBERTa`, ktorý v porovnaní s menším modelom `DistilBERT` dosiahol zreteľné zlepšenie v metrike EM. Model `RoBERTa Base` vykazuje v porovnaní s `TinyRoBERTa` už len marginálne zlepšenie výkonu, čo naznačuje, že ďalšie zvyšovanie počtu parametrov modelu už neprináša zásadné zlepšenie výkonu pri riešení tejto úlohy. Tento trend je v súlade s očakávaniami, keďže väčšie modely disponujú väčšou reprezentačnou schopnosťou, čo im umožňuje lepšie zachytávať komplexné vzťahy v dátach. Napriek tomu aj menšie modely `DistilBERT` a `TinyRoBERTa` preukázali robustný výkon vzhľadom na svoju redukovanú architektúru a nižší počet parametrov, čím potvrdili svoju vhodnosť pre úlohy s obmedzenými výpočtovými zdrojmi.

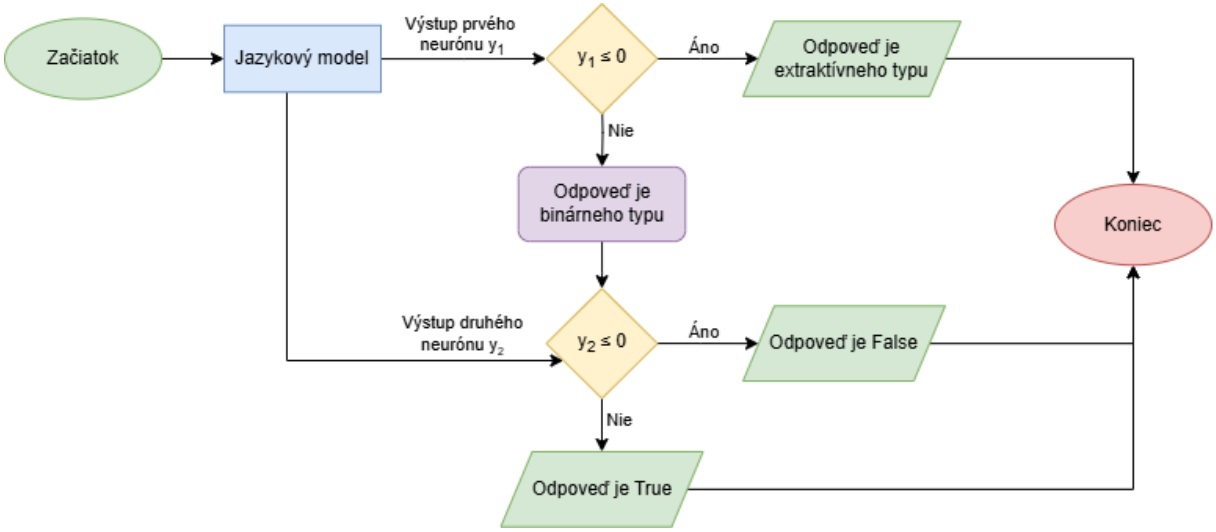
6.3.2 Multilingválna úloha

V multilingválnej úlohe QA boli otázky rozdelené do dvoch kategórií: extraktívne a binárne. Zatiaľ čo extraktívne otázky vyžadovali lokalizáciu odpovede v kontexte, bi-

Model	EM [%]	F1-skóre [%]	Precision [%]	Recall [%]
`DistilBERT`	62,82	75,82	82,96	79,85
`TinyRoBERTa`	66,28	79,03	84,35	80,86
`RoBERTa Base`	68,42	80,21	85,25	82,00

Tabuľka 6.1: Porovnanie extraktívnych metrík pre monolingválne modely

nárne otázky očakávali odpoveď v binárnej forme, t.j. True alebo False. Proces generovania odpovede začínal identifikáciou typu otázky, ktorá určovala následný spôsob predikcie odpovede. Priebeh rozhodovania je znázornený na obrázku 6.9.



Obr. 6.9: Rozhodovací diagram pre klasifikáciu typu otázky

Klasifikácia typu otázky

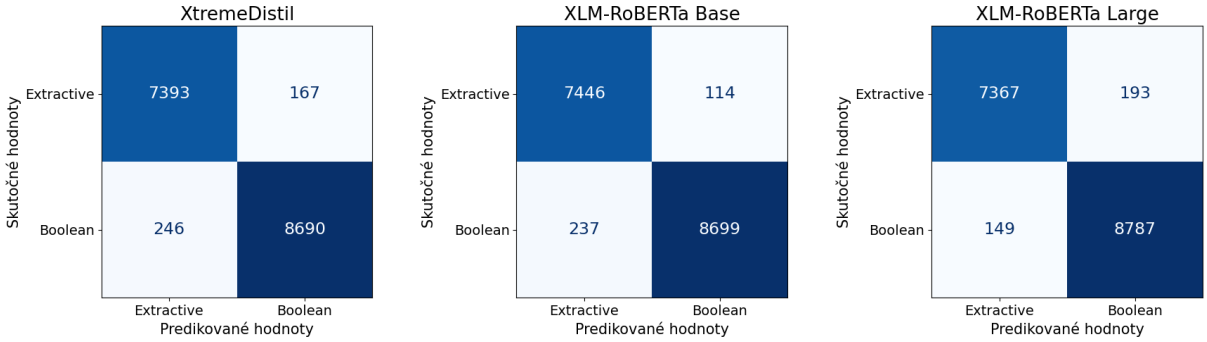
Na tento účel bol použitý model rozšírený o dodatočnú klasifikačnú hlavu, ktorý je detailne popísaný v podkapitole 5.7. Výstup prvého neurónu tejto vrstvy určoval, či ide o extraktívnu alebo binárnu otázku, pozri (5.1). V prípade, že otázka bola klasifikovaná ako binárna, druhý neurón rozhodoval o samotnej odpovedi (True/False), pozri (5.2). Výsledky úspešnosti klasifikácie typu otázky sú zhrnuté v maticiach zámeny na obrázku 6.10 a súhrnné metriky sú uvedené v tabuľke 6.2.

Výsledky naznačujú, že všetky použité modely dosiahli v klasifikácii typu otázky veľmi vysokú úspešnosť.

Modely `XtremeDistil` a `XLM-RoBERTa Base` dosiahli vyššie hodnoty precision než recall, čo naznačuje ich tendenciu k opatrnejším predikciám s nižším počtom falošne pozitívnych rozhodnutí. Naopak, model `XLM-RoBERTa Large` vykazuje mierne vyšší recall

Model	Accuracy [%]	F1-skóre [%]	Precision [%]	Recall [%]
`XtremeDistil`	97,50	97,68	98,11	97,25
`XLM-RoBERTa Base`	97,87	98,02	98,71	97,35
`XLM-RoBERTa Large`	97,93	98,09	97,85	98,33

Tabuľka 6.2: Porovnanie metrík pre klasifikáciu typu otázky v multilingválnych modeloch



Obr. 6.10: Matice zámeny klasifikácie typu otázky pre multilingválnu úlohu

než precision, čo poukazuje na schopnosť lepšie zachytávať pozitívne prípady aj za cenu mierne vyššieho počtu falošne pozitívnych predikcií.

Model `XLM-RoBERTa Base` dosiahol celkovo najvyššie hodnoty accuracy (97,87 %) a F1-skóre (98,02 %), zatiaľ čo model `XtremeDistil` napriek menšej architektúre preukázal robustný výkon, najmä v metrike precision (98,11 %). Výsledky modelu `XLM-RoBERTa Large` sú porovnateľné so základnou verziou, pričom vyšší recall môže byť dôsledkom jeho väčšej reprezentačnej kapacity.

Určenie extraktívnej odpovede

V prípade, že otázka bola klasifikovaná ako extraktívna, postupujeme analogicky ako pri monolingválnych modeloch, t.j. na základe pravdepodobnostného rozdelenia vypočítaného nad jednotlivými tokenmi vyberieme najpravdepodobnejší začiatok a koniec odpovede. Následne nahradíme číselné reprezentácie tokenov ich slovnými ekvivalentmi a úspešnosť predikcie vyhodnotíme pomocou zvolených metrík, ktorých hodnoty sú zhrnuté v tabuľke 6.3.

Model `XtremeDistil` dosiahol EM skóre 59,71 % a F1-skóre 77,20 %. Aj napriek tomu, že v metrike EM dosiahol horšie výsledky v porovnaní s robustnejšími modelmi, jeho hodnoty precision (74,77 %) a recall (79,80 %) sú vysoké a naznačujú schopnosť zachytiť väčšinu relevantných odpovedí. V porovnaní s väčšími modelmi však vykazuje menej vyvážený pomer medzi precision a recall, čo naznačuje tendenciu modelu predikovať

Model	EM [%]	F1-skóre [%]	Precision [%]	Recall [%]
`XtremeDistil`	59,71	77,20	74,77	79,80
`XLM-RoBERTa Base`	76,78	89,37	89,11	89,65
`XLM-RoBERTa Large`	81,40	91,21	91,44	90,99

Tabuľka 6.3: Porovnanie metrík pre extraktívnu QA časť multilingválnych modelov

širšie alebo menej presné úseky textu. Napriek tomu preukazuje `XtremeDistil` robustný výkon vzhľadom na svoju redukovanú architektúru a obmedzený počet parametrov.

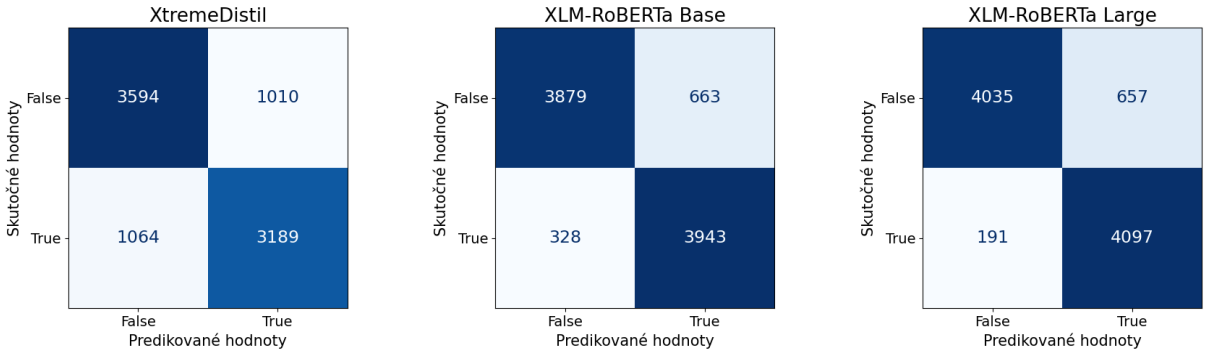
Model `XLM-RoBERTa Base` dosiahol výrazne lepšie výsledky oproti modelu `XtremeDistil` so 76,78 % EM skóre a 89,37 % F1-skóre. Vysoké a vyvážené hodnoty precision (89,11 %) a recall (89,65 %) ukazujú na vyváženú schopnosť modelu správne identifikovať a zachytiť odpovede. Tento model dosahuje vysokú robustnosť a stabilitu naprieč všetkými metrikami.

Model `XLM-RoBERTa Large` dosiahol najvyššie hodnoty vo všetkých sledovaných metrikách. Výrazne prekonáva základnú verziu modelu v metrike EM (81,40 %). F1-skóre (91,21 %), precision (91,44 %) a recall (90,99 %) len mierne prekonávajú základnú verziu modelu. Tieto výsledky potvrdzujú, že väčšia kapacita modelu prináša zlepšenie výkonu v porovnaní so základnou verziou. Pozorované výsledky naznačujú, že v tomto prípade zväčšovanie modelu alebo zvyšovanie počtu parametrov vedie k zlepšeniu, avšak s postupne klesajúcim prírastkom efektivity.

Klasifikácia binárnej odpovede

V prípade, že otázka bola v prvom kroku klasifikovaná ako binárna, rozhodnutie o odpovedi `True` alebo `False` sa uskutočňuje na základe výstupu druhého neurónu klasifikačnej hlavy, ako je uvedené v definícii (5.2). Otázky klasifikované ako extraktívne sa do tohto rozhodovacieho procesu už nezaraďujú. Takto navrhnutá binárna klasifikácia umožňuje modelu efektívne spracovávať veľké množstvo otázok aj pri vysokom počte otázok. Presnosť a stabilita predikcií sú podmienené kvalitným natrénovaním váh vstupov oboch neurónov klasifikačnej hlavy, ako aj správnym nastavením ich prahových hodnôt. Výsledky úspešnosti klasifikácie binárnych otázok sú zhrnuté v maticiach zámeny na obrázku 6.11 a súhrnné metriky sú uvedené v tabuľke 6.4.

Model `XtremeDistil` dosiahol najnižšie hodnoty vo všetkých sledovaných metrikách. Jeho matica zámeny vykazuje vyvážený výskyt falošne pozitívnych (1010) a falošne negatívnych (1064) predikcií, čo poukazuje na obmedzenú schopnosť modelu spoľahlivo odlíšiť obe triedy. Hodnoty precision (75,95 %) a recall (74,98 %) sú vzájomne vyvážené, čo naz-



Obr. 6.11: Matice zámeny klasifikácie binárnych otázok pre multilingválnu úlohu

Model	**Accuracy** [%]	**F1-skóre** [%]	**Precision** [%]	**Recall** [%]
`XtremeDistil`	76,58	75,45	75,95	74,98
`XLM-RoBERTa Base`	88,76	88,84	85,61	92,32
`XLM-RoBERTa Large`	90,56	90,62	86,18	95,55

Tabuľka 6.4: Porovnanie metrík pre klasifikáciu binárnych otázok v multilingválnych modeloch

načuje symetrickú, no suboptimálnu výkonnosť v rámci detekcie pozitívnych a negatívnych prípadov.

Výrazne lepšie výsledky dosiahol model `XLM-RoBERTa Base`, ktorého matica zámeny obsahuje výrazne menej nesprávnych predikcií (663 falošne pozitívnych a 328 falošne negatívnych). Výsledné metriky odzrkadľujú zlepšenú mieru generalizačnej schopnosti: precision 85,61 %, recall 92,32 % a F1-skóre 88,84 %. Prevaha recall nad precision naznačuje tendenciu modelu uprednostniť citlivosť (senzitivitu) pred špecifickosťou, teda maximalizovať zachytenie relevantných prípadov aj za cenu mierne zvýšeného počtu falošne pozitívnych predikcií.

Model `XLM-RoBERTa Large` dosiahol najvyššie celkové skóre, s accuracy 90,56 % a F1-skóre 90,62 %. Matica zámeny indikuje najnižší počet chybných predikcií (657 falošne pozitívnych a 191 falošne negatívnych), čo sa premieta do výnimočne vysokého recall (95,55 %). Hoci precision (86,18 %) značne zaostáva za recall, stále predstavuje solídnu hodnotu. Výsledky naznačujú, že model efektívne identifikuje väčšinu pozitívnych prípadov, pričom si zároveň zachováva akceptovateľnú mieru nesprávnych pozitívnych predikcií.

6.4 Diskusia

V tejto podkapitole porovnáme implementovaný prístup s vybranými prácami, ktoré zohrali dôležitú úlohu vo vývoji QA systémov a spracovania prirodzeného jazyka.

Prvým významným východiskom je architektúra **BERT** [35], ktorá priniesla zásadný prelom v oblasti strojového porozumenia textu. Model BERT bol navrhnutý pre monolingválne anglické úlohy a zaznamenal vysoký výkon v extraktívnej úlohe QA, konkrétne dosiahol hodnoty 80,8 % v EM-skóre a 88,5 % v F1-skóre na datasete SQuAD 1.1. V rámci tejto práce boli použité modely vychádzajúce z architektúry BERT, pričom najlepšie výsledky v monolingválnej úlohe zaznamenal model `RoBERTa Base`, ktorý dosiahol EM-skóre 68,4 % a F1-skóre 80,2 %. Treba však zdôrazniť, že trénovanie monolingválnych modelov nebolo hlavným cieľom práce, ale slúžilo ako doplnkové porovnanie a bolo realizované na pomerne malom datasete obsahujúcom 4723 otázok, ktorý predstavuje podmnožinu kompletnej dátovej sady.

Na architektúru BERT nadväzuje práca uvádzajúca model **RoBERTa** [8], ktorá pri trénovaní modelu odstraňuje úlohu predikcie ďalšej vety a využíva rozsiahlejšie dátové sady. V tejto práci bol v monolingválnej úlohe použitý model `RoBERTa Base`, ktorý prekonal menšie modely `DistilBERT` a `TinyRoBERTa`, čím potvrdzujeme trend z pôvodnej práce [8], že výkon rastie so zložitosťou modelu. Prínos našej práce však spočíva aj v presune pozornosti do multilingválneho prostredia, kde bol použitý model `XLM-RoBERTa Large`. Ten dosiahol 90,6 % accuracy v binárnej klasifikácii a 91,2 % F1-skóre v extraktívnej úlohe, čím sa výkonnostne priblížil hodnotám udávaným pre RoBERTa na anglickom datasete SQuAD 2.0 (F1: 94,6 %).

Tématicky príbuzný našej práci je aj dataset **BoolQ** [11], ktorý je zameraný výhradne na binárne otázky s odpoveďou Áno/Nie. Naša práca rozširuje binárnu úlohu QA do multilingválneho prostredia a zavádza dvojstupňový mechanizmus klasifikácie typu otázky. Ten najprv určuje typ otázky (extraktívna alebo binárna) a následne, v prípade, že otázka je klasifikovaná ako binárna, predikuje odpoveď vo forme `Pravda/Nepravda`. Predtrénovaný model `XLM-RoBERTa Large`, ktorý bol jemne doladený v rámci našej práce na nami vytvorený dataset, dosiahol v tejto úlohe accuracy 90,6 %, čo výrazne prevyšuje výsledky modelu `BERT-Large` jemne doladeného na datasete BoolQ, ktorý dosiahol accuracy približne 80 %. Tento rozdiel možno pripísať robustnejšej architektúre nami použitého modelu, rozdielnym charakteristikám dát použitých na jemné doladenie a odlišnému návrhu klasifikačného mechanizmu.

Najvýznamnejším porovnateľným datasetom v oblasti extraktívneho QA je dataset

SQuAD [48], ktorý predstavuje etablovaný štandard pre hodnotenie schopnosti modelov porozumieť prirodzenému textu. Obsahuje výlučne anglické kontexty a otázky, pričom anotácia odpovedí prebiehala manuálne s dôrazom na kvalitu a presnosť. V kontraste s týmto prístupom bola v rámci predkladanej práce vytvorená rozsiahla dátová sada s využitím generatívnych jazykových modelov. Táto databáza pokrýva tri jazyky (anglický, nemecký a francúzsky) a kombinuje extraktívne, binárne a číselné otázky. V porovnaní s [48] sme použili vyšší stupeň automatizácie pri anotácii odpovedí, čo redukuje čas a náklady potrebné na túto anotáciu.

Napokon, táto práca má spoločné črty s rámcom **LIQUID** [20], ktorý navrhuje spôsob automatizovaného generovania a anotácie dát. Zatiaľ čo LIQUID využíva modulárny systém s explicitnou kontrolou kvality, v našej práci bol proces generovania a anotácie dát zverený veľkým jazykovým modelom. Tým sa výrazne zjednodušuje implementácia a zvyšuje sa škálovateľnosť riešenia.

Môžeme teda konštatovať, že naša práca sleduje súčasné výskumné trendy, a to nielen na úrovni použitých architektúr modelov, ale aj pri tvorbe datasetu. Okrem toho prináša viacero vlastných riešení, napríklad kombináciu typov otázok, multilingválny rozsah a dvojstupňovú klasifikáciu, ktoré boli v rámci nej aj implementované.

Kapitola 7

Záver

V tejto diplomovej práci sme sa zamerali na analýzu výkonnosti modelov Transformerov pri riešení úloh QA. Naším cieľom bolo vytvoriť rozsiahly dataset, ktorý kombinuje extraktívne a binárne otázky a zároveň pokrýva tri jazyky (anglický, nemecký a francúzsky). Okrem toho sme navrhli rozšírenie modelovej architektúry o klasifikačnú hlavu, ktorá umožňuje efektívne rozlíšiť typ otázky a v prípade binárnych otázok určiť odpoveď vo forme Pravda/Nepravda.

Významným praktickým prínosom tejto práce je aj automatizovanie procesu generovania kontextov a anotácie otázok s využitím pokročilých generatívnych jazykových modelov. Tento prístup výrazne eliminuje prácne manuálne anotovanie dát a výrazne zvyšuje škálovateľnosť celého riešenia. Vytvorený dataset, ako aj návrh klasifikačnej hlavy, boli úspešne využité pri trénovaní a evaluácii modelov v monolingválnej aj multilingválnej konfigurácii.

Výsledky experimentálnej časti preukázali, že väčšie modely (napríklad `XLM-RoBERTa Large`) prekonávajú menšie modely (napríklad `XtremeDistil`) vo všetkých sledovaných metrikách, pričom rozdiely boli v niektorých prípadoch výrazné. Tieto výsledky naznačujú, že vyššia kapacita modelu umožňuje lepšiu generalizáciu a spoľahlivejšiu klasifikáciu odpovedí a to najmä v multilingválnej konfigurácii.

Všetky stanovené ciele tejto práce boli úspešne splnené. Podarilo sa nám navrhnúť funkčné rozšírenie Transformer architektúry, vytvoriť kvalitný multilingválny dataset a experimentálne overiť výkonnosť modelov rôznych veľkostí a výpočtových nárokov.

Do budúcnosti existuje viacero možností na ďalší rozvoj tejto práce. Ako perspektívne smerovanie sa ponúka rozšírenie datasetu o ďalšie jazyky či jazykové rodiny a ich následné porovnanie. Rovnako je možné zaviesť nové typy otázok, experimentovať s tokenizáciou dát alebo využiť augmentáciu na zvýšenie rozmanitosti vstupných textov.

Jedným zo zaujímavých smerov výskumu je formulácia úlohy QA ako úlohy predikcie začiatku a konca odpovede v rámci daného textu. Túto formuláciu by bolo možné ako pri extraktívnych, tak aj pri binárnych otázkach. Takýto prístup by mohol viesť k zjednoteniu modelovej architektúry a zvýšeniu flexibility systému pri práci s rôznorodými typmi otázok.

Zoznam použitej literatúry

[1] Kowsher, M. et al., 2022. Bangla-BERT: Transformer-Based Efficient Model for Transfer Learning and Language Understanding. In: *IEEE Access*. Vol. 10, p. 91855–91870. ISSN: 2169-3536.

[2] JONES, K. S., 1972. A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL. In: *Journal of Documentation*. Vol. 28, no. 1, p. 11–21. ISSN: 0022-0418.

[3] Robertson, S. et al., 1994. Okapi at TREC-3. In: *Proceedings of the Third Text REtrieval Conference (TREC-3)*. National Institute of Standards a Technology (NIST). s. 109–126. ISBN: 978-0-7881-2945-2.

[4] Abdallah, A. et al., 2025. *From Retrieval to Generation: Comparing Different Approaches*. arXiv preprint arXiv:2502.20245. URL: `http://arxiv.org/abs/2502.20245`.

[5] Hambarde, K. A. A. a Proenca, H., 2023. Information Retrieval: Recent Advances and Beyond. In: *IEEE ACCESS*. Vol. 11, p. 76581–76604. ISSN: 2169-3536.

[6] Lu, M., Chen, C. a Eickhoff, C., 2025. *Cross-Encoder Rediscovers a Semantic Variant of BM25*. arXiv preprint arXiv:2502.04645. URL: `http://arxiv.org/abs/2502.04645`.

[7] Cuevas, J. P. et al., 2024. MédicoBERT: A Medical Language Model for Spanish Natural Language Processing Tasks with a Question-Answering Application Using Hyperparameter Optimization. In: *Applied Sciences (Switzerland)*. Vol. 14, no. 16. ISSN: 2076-3417.

[8] Liu, Y. et al., 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. URL: `https://arxiv.org/abs/1907.11692`.

[9] Koto, F. et al., 2020. IndoLEM and IndoBERT: A Benchmark Dataset and Pretrained Language Model for Indonesian NLP. In: *Proceedings of the 28th International Conference on Computational Linguistics (COLING 2020)*. s. 757–770.

[10] Campos, M. M. a Couto, F. M., 2021. Post-processing BioBERT And Using Voting Methods for Biomedical Question Answering. In: *Conference and Labs of the Evaluation Forum, September 21–24, 2021*. Bucharest, Romania.

[11] Clark, C. et al., 2019. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics. s. 2924–2936.

[12] Rakotoson, L. et al., 2022. Extractive-Boolean Question Answering for Scientific Fact Checking. In: *MAD 2022 - Proceedings of the 1st International Workshop on Multimedia AI against Disinformation*. Association for Computing Machinery, Inc. s. 27–34. ISBN: 978-1-4503-9242-6.

[13] Jin, Q. et al., 2019. PubMedQA: A Dataset for Biomedical Research Question Answering. In: *2019 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING AND THE 9TH INTERNATIONAL JOINT CONFERENCE ON NATURAL LANGUAGE PROCESSING (EMNLP-IJCNLP 2019): PROCEEDINGS OF THE CONFERENCE*. s. 2567–2577. ISBN: 978-1-950737-90-1.

[14] Ozyurt, I. B., Bandrowski, A. a Grethe, J. S., 2020. Bio-AnswerFinder: A system to find answers to questions from biomedical texts. In: *Database*. Vol. 2020. ISSN: 1758-0463.

[15] Zhou, J. et al., 2019. GEAR: Graph-based Evidence Aggregating and Reasoning for Fact Verification. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics. s. 892–901.

[16] Green, B. F. et al., 1961. Baseball: an automatic question-answerer. In: *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*. Association for Computing Machinery. s. 219–224. ISBN: 978-1-4503-7872-7.

[17] Woods, W. A., Kaplan, R. M. a Webber, B. L., 1972. *The Lunar Sciences Natural Language Information System: Final Report*: výskumná správa. Cambridge, MA: Bolt Beranek a Newman Inc. Dostupné na: URL: `https://ntrs.nasa.gov/citations/19720022936`.

[18] Katz, B., 1993. *START: Natural Language Question Answering System*. [online] [cit. 05. 04. 2025]. Dostupné na: URL: `https://start.csail.mit.edu/index.php`.

[19] Databricks, 2024. *What is a Dataset?* [online] [cit. 29. 04. 2025]. Dostupné na: URL: `https://www.databricks.com/glossary/what-is-dataset`.

[20] Lee, S., Kim, H. a Kang, J., 2023. LIQUID: A Framework for List Question Answering Dataset Generation. In: *37th AAAI Conference on Artificial Intelligence*. Vol. 37, no. 11. s. 13014–13024. ISSN: 2159-5399.

[21] Hopfield, J. J., 1982. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. In: *Proceedings of the National Academy of Science*. Vol. 79, no. 8, p. 2554–2558. ISSN: 0027-8424.

[22] Jordan, M. I., 1990. Attractor dynamics and parallelism in a connectionist sequential machine. In: *Artificial Neural Networks: Concept Learning*. Los Alamitos, CA, USA: IEEE Press. s. 112–127. ISBN: 978-0-8186-2015-7.

[23] Elman, J. L., 1990. Finding structure in time. In: *Cognitive Science*. Vol. 14, no. 2, p. 179–211. ISSN: 0364-0213.

[24] Hochreiter, S. a Schmidhuber, J., 1997. Long Short-Term Memory. In: *Neural Computation*. Vol. 9, no. 8, p. 1735–1780. ISSN: 0899-7667.

[25] Cho, K. et al., 2014. Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics. s. 1724–1734. ISBN: 978-1-937284-96-1.

[26] Toharudin, T. et al., 2023. Employing Long Short-Term Memory and Facebook Prophet Model in Air Temperature Forecasting. In: *Communications in Statistics - Simulation and Computation*. Vol. 52, no. 2, p. 279–290. ISSN: 0361-0918.

[27] Bahdanau, D., Cho, K. a Bengio, Y., 2014. *Neural Machine Translation by Jointly Learning to Align and Translate*. arXiv preprint arXiv:1409.0473. URL: `https://arxiv.org/abs/1409.0473`.

[28] Luong, T., Pham, H. a Manning, C. D., 2015. Effective Approaches to Attention-based Neural Machine Translation. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal: Association for Computational Linguistics. s. 1412–1421. ISBN: 978-1-941643-32-7.

[29] Seo, M. et al., 2016. *Bidirectional Attention Flow for Machine Comprehension.* arXiv preprint arXiv:1611.01603. URL: https://arxiv.org/abs/1611.01603.

[30] Chen, D. et al., 2017. Reading Wikipedia to Answer Open-Domain Questions. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Vancouver, Canada: Association for Computational Linguistics. s. 1870–1879.

[31] Vaswani, A. et al., 2017. Attention Is All You Need. In: *Advances in Neural Information Processing Systems 30 (NIPS 2017).* Vol. 30. ISSN: 1049-5258.

[32] Mikolov, T. et al., 2013. *Efficient Estimation of Word Representations in Vector Space.* Presented at ICLR 2013 Workshop. URL: https://arxiv.org/abs/1301.3781.

[33] Pennington, J., Socher, R. a Manning, C. D., 2014. GloVe: Global Vectors for Word Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Doha, Qatar: Association for Computational Linguistics. s. 1532–1543. ISBN: 978-1-937284-96-1.

[34] Peters, M. E. et al., 2018. Deep Contextualized Word Representations. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).* New Orleans, Louisiana: Association for Computational Linguistics. s. 2227–2237.

[35] Devlin, J. et al., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* Minneapolis, Minnesota: Association for Computational Linguistics. s. 4171–4186. ISBN: 978-1-950737-13-0.

[36] Radford, A. et al., 2018. *Improving Language Understanding by Generative Pre-Training*: výskumná správa. San Francisco: OpenAI. Dostupné na: URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[37] Conneau, A. et al., 2020. Unsupervised Cross-lingual Representation Learning at Scale. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020).* Online (hosted in Seattle, Washington, USA): Association for Computational Linguistics. s. 8440–8451.

[38] Cappi, C. et al., 2021. *Dataset Definition Standard (DDS)*. arXiv preprint arXiv:2101.03020. URL: https://arxiv.org/abs/2101.03020.

[39] Wu, Y. et al., 2016. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. arXiv preprint arXiv:1609.08144. URL: https://arxiv.org/abs/1609.08144.

[40] Song, X. et al., 2021. Fast WordPiece Tokenization. In: *2021 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING (EMNLP 2021)*. s. 2089–2103. ISBN: 978-1-955917-09-4.

[41] Hugging Face, 2024. *LLM Course, Chapter 6 - Fine-tuning*. [online] [cit. 05. 04. 2025]. Dostupné na: URL: https://huggingface.co/learn/llm-course/en/chapter6/6.

[42] Sanh, V. et al., 2019. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. arXiv preprint arXiv:1910.01108. URL: https://arxiv.org/abs/1910.01108.

[43] Jiao, X. et al., 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online a Punta Cana, Dominican Republic: Association for Computational Linguistics. s. 4163–4174.

[44] Mukherjee, S., Awadallah, A. H. a Gao, J., 2021. *XtremeDistilTransformers: Task Transfer for Task-agnostic Distillation*. URL: https://arxiv.org/abs/2106.04563.

[45] Conneau, A. et al., 2020. Unsupervised Cross-lingual Representation Learning at Scale. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics. s. 8440–8451. ISBN: 978-1-952148-25-5.

[46] Hinton, G. E., Vinyals, O. a Dean, J., 2015. *Distilling the Knowledge in a Neural Network*. arXiv preprint arXiv:1503.02531. URL: https://arxiv.org/abs/1503.02531.

[47] Seraydarian, L., 2022. *What Is a Confusion Matrix And How to Read It?* [online] [cit. 18. 04. 2025]. Dostupné na: URL: https://plat.ai/blog/confusion-matrix-in-machine-learning/.

[48] Rajpurkar, P. et al., 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics. s. 2383–2392.