Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

PLZEŇ, 2023 **Jakub Zieba**

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne ....

Poděkování

Chtěl bych tímto poděkovat panu Ing. Janu Švecovi, Ph.D. za vzorné vedení této práce.

Jakub Zieba

Abstrakt

Tato práce se zabývá analýzou využití strojového překladu v úlohách porozumění přirozenému jazyku. Prozkoumává možnosti použití českých strojových překladů anglických datasetů pro trénování modelů strojového učení a jejich aplikace na úlohu inference přirozeného jazyka. Pro překlad datasetů využívá online služeb strojového překladu. Proces trénování využívá přeneseného učení a existujících předtrénovaných modelů. Cílem práce je zhodnocení výsledků českých modelů trénovaných na přeložených datech a porovnání s výsledky anglických modelů trénovaných na ekvivalentních originálních datech.

Klíčová slova

zpracování přirozeného jazyka, porozumění přirozenému jazyku, inference přirozeného jazyka, strojové učení, přenesené učení, neuronové sítě, transformer, strojový překlad

Abstract

This paper analyzes the usage of machine translation in natural language understanding tasks. It explores the possibilities of using Czech machine translation of English datasets for training machine learning models and their application on natural language inference tasks. It uses online machine translation services for dataset translation. The training process utilizes transfer learning and existing pre-trained models. Finally, it evaluates the results of Czech models trained on translated data and compares them to the performance of English models trained on equivalent original data.

Key words

natural language processing, natural language understanding, natural language inference, machine learning, transfer learning, neural networks, transformer, machine translation

Obsah

1 Úvod

Lidé na planetě Zemi nemluví pouze jedním jazykem. Různé kultury a oblasti využívají ke komunikaci různé jazyky, často s velmi odlišnými výrazy a pravidly. Tyto jazyky se vyvíjely po značnou část lidské historie a postupně vznikaly i jejich písemné podoby. Pro člověka je běžné rozumět a být schopen aktivně používat jeden či více jazyků, i přes jejich velmi složitou a často flexibilní strukturu.

Vzhledem k důležitosti jazyka pro lidskou komunikaci není překvapivé, že po nástupu výpočetních technologií začaly snahy o vývoj algoritmů pro automatické zpracování přirozeného jazyka. Tento vývoj postupně vedl k vytvoření systémů pro automatické překlady, analýzu či generování textu, jejichž kvalita a možnosti využití v současnosti rapidně roste.

Vývoj těchto systémů je však zpravidla zaměřen na malé množství jazyků, často pouze na jeden. U více používaných jazyků je pro tento vývoj k dispozici větší množství dat a zdrojů. U málo používaných jazyků může tedy být vývoj algoritmů pro jeho strojové zpracování pracnější.

Cílem této práce je prozkoumat možnosti využití systémů strojového překladu za účelem zjednodušení procesu získávání dat pro vývoj algoritmů strojového zpracování přirozeného jazyka. Konkrétně je práce zaměřena na využití dat v anglickém jazyce pro úlohy zpracování českého jazyka s pomocí volně dostupných služeb strojového překladu.

2 Úlohy zpracování jazyka

V této kapitole je popsán princip a rozdělení klasifikačních úloh, základní problematika strojového zpracování přirozeného jazyka a následně úloha inference přirozeného jazyka, na kterou se tato práce bude zaměřovat, včetně analýzy vhodných datasetů.

2.1 Klasifikační úlohy

Klasifikace je úloha strojového učení, která se zabývá zařazením jednotlivých objektů, reprezentovaných jejich obrazem, do jedné či několika z předem definovaných tříd. Obrazem je obecně vektor, jenž může být reprezentací textu, obrázku, zvuku, atd.

Existuje mnoho různých přístupů ke klasifikaci. Zde jsou stručně popsány ty základní: [1]

- **Binární klasifikace**

  V úlohách binární klasifikace se určuje, zda daný obraz spadá do určité třídy či ne. Pokud obraz ke třídě náleží, označuje se tento případ jako „pozitivní“. V opačném případě se jedná o případ „negativní“. Výstupem je tedy pouze informace typu ano/ne.

  Příkladem takové klasifikace může být detekce spamu, padělku, nemoci, apod.

- **Klasifikace do více tříd**

  V případě klasifikace do více tříd (*multi-class classification*) se daný obraz řadí do právě jedné z mnoha definovaných tříd. Na rozdíl od binární klasifikace není jejím výstupem pouze ano/ne, ale je jím jedna hodnota (štítek) odpovídající právě jedné třídě.

  Tento typ klasifikace lze použít například pro rozpoznání obličejů, rostlin, vozů, ale také pro třídění vět nebo promluv dle nálady, významu, použitého jazyka, atd.

- **Klasifikace s více štítky**

  Na rozdíl od klasifikace do více tříd, kde výstupem byla vždy právě jedna třída, klasifikace s více štítky (*multi-label classification*) poskytuje více štítků pro daný vstupní obraz. Dá se tedy říci, že každý obraz může patřit do více tříd najednou.

  Tuto klasifikaci je možné použít pro analýzu fotografií, souvislého textu, příspěvků na sociálních sítích, apod.

Ne každá klasifikační úloha přesně odpovídá jednomu z těchto typů. Často se používají různé obměny a úpravy v závislosti na konkrétní řešené úloze. Například nevyvážená klasifikace (*imbalanced classification*) je speciální případ binární klasifikace, při které převážná většina vstupních dat spadá do pozitivní třídy, a je proto třeba při trénování využít specializovaných technik. [1]

Konkrétní úlohy se také mohou lišit podobou výstupu. K samotné informaci o zvolené třídě mohou například dodat i hodnotu jistoty (*confidence score*). Způsoby výpočtu této hodnoty jsou různé, avšak vždy se jedná o informaci popisující jak moc „jistý“ si klasifikační algoritmus je v daném rozhodnutí.

U pravděpodobnostních algoritmů (např. neuronových sítí) je možné pro každou třídu získat přímo hodnotu pravděpodobnosti, s jakou do ní určitý obraz náleží. Zvolená třída je poté ta, pro kterou je tato hodnota maximální. Informace o hodnotách pravděpodobností může být užitečná pro další zpracování a analýzu výstupu. [2]

Klasifikační algoritmy jsou užitečné v případech, kdy chceme roztřídit objekty do předem známých tříd na základě jejich podoby a vlastností. Používají se v řadě aplikací od rozpoznávání obličejů přes analýzu přirozeného textu až po detekci nemocí.

2.2 Zpracování přirozeného jazyka (NLP)

Strojové zpracování přirozeného jazyka (*Natural Language Processing*, NLP) je oblast umělé inteligence zabývající se využitím počítačů pro zpracování či porozumění přirozenému textu.

NLP využívá kombinace znalostí z několika oblastí, především z informačních technologií a lingvistiky. Jeho cílem je návrh a vývoj technologií pro plnění široké množiny užitečných jazykových úloh zahrnujících porozumění významu textu, překlad do jiného jazyka, generování textu, apod. [3]

Oblast NLP využívá metod strojového učení, které v dnešní době zahrnují především statistické modely na principu neuronových sítí. Více o těchto modelech je sepsáno v kapitole 5.

Problematiku NLP lze rozdělit na dvě hlavní oblasti: porozumění textu a generování textu. Konkrétní úloha může spadat buď pouze do jedné z nich, nebo využívat obou. [4]

2.2.1 Porozumění přirozenému jazyku (NLU)

Porozumění přirozenému jazyku (*Natural Language Understanding*, NLU) je základem oblasti NLP a zabývá se vývojem systémů pro sémantickou analýzu textu za účelem jeho převedení do významové reprezentace. Důležitou součástí této analýzy je uvažování kontextu, který může značně ovlivnit význam textu.

Existuje mnoho úloh využívajících porozumění jazyku pro odlišné účely. Zde je uvedeno několik populárních příkladů: [3, 4]

- **Analýza sentimentu**

  Jedná se o úlohu klasifikace emocionálního zbarvení textu do několika tříd, které zpravidla odpovídají pozitivní, negativní a neutrální emoci.

- **Rozpoznávání entit**

  Tato úloha se zabývá extrakcí předem definovaných entit z textu a jejich klasifikací. Může se jednat o jména lidí, názvy míst, letopočty, příkazy, apod.

- **Inference jazyka**

  Úkolem inference jazyka je rozhodnout, zda zadaný předpoklad podporuje danou hypotézu nebo ji vyvrací. Více o této úloze je v kapitole 2.3.

- **Modelování témat**

  Úloha modelování témat spadá do oblasti učení bez učitele a jejím cílem je analýza textových dokumentů za účelem zjištění jejich hlavních témat.

- **Vyhledávání informací**

  Jedná se o úlohu vyhledávání textových dokumentů v daném souboru, které jsou relevantní k zadanému požadavku.

2.2.2 Generování přirozeného jazyka (NLG)

Generování přirozeného jazyka (*Natural Language Generation*, NLG) má za cíl produkování souvislého textu, který je podobný tomu psanému člověkem. Může se jednat buď o pouhou kompozici odpovědí z předem definovaných frází, nebo o komplexnější model schopný imitovat různé styly a formáty. [4]

2.2.3 Kombinace NLU a NLG

Jak bylo již zmíněno, porozumění a generování textu nemusí být vždy dva oddělené systémy. Kombinace obou oblastí je užitečná pro plnění mnoha různých úloh, například následujících: [4]

- **Automatické doplňování**

  Systémy automatického doplňování („autocomplete“) generují pokračování daného textu na základě jeho porozumění.

- **Vedení přirozených konverzací**

  Jedná se o automatizaci jedné strany konverzace, kde druhou stranu zpravidla představuje člověk. Tito „chatboti“ mohou buď pouze poskytovat předdefinované odpovědi, nebo vést komplexní přirozené konverzace.

- **Překlad textu**

  Tato úloha se zabývá automatickým překladem textu z jednoho jazyka do druhého. Více je popsáno v kapitole 6.

- **Hlasový dialogový systém**

  NLP je důležitou součástí hlasových dialogových systémů, které slouží pro vedení přirozeného hlasového dialogu s uživatelem. Algoritmy rozpoznávání řeči poskytují textovou reprezentaci hlasu, jejíž význam je dále posouzen pomocí NLU. Pro vygenerování odpovědi je využit NLG systém, jehož textový výstup je dále převeden na zvuk pomocí generátoru řeči.

Jak lze vidět, NLP má mnoho možných praktických využití a jedná se v dnešní době o jednu z nejrychleji se vyvíjejících oblastí umělé inteligence.

Ještě na začátku tohoto století bylo NLP využíváno pouze pro experimentální systémy bez využití v praxi. Dnes je však součástí mnoha komerčních systémů, jejichž množství stále narůstá. [3]

2.3 Inference přirozeného jazyka (NLI)

Inference přirozeného jazyka (*Natural Language Inference*, NLI) je úloha oblasti NLU zahrnující klasifikaci významového vztahu dvou vět do předem daných tříd.

Jedna z vět je označena jako *premise* (předpoklad) a druhá jako *hypothesis* (hypotéza). Úkolem NLI je rozhodnout, zda daný předpoklad implikuje danou hypotézu nebo ji vyvrací.

Konkrétní třídy popisující tyto vztahy se mohou lišit dle úlohy. V některých případech se jedná o rozhodování, zda spolu věty souvisí nebo ne. V jiných úlohách můžeme zjišťovat, zda daný fakt podporuje určité tvrzení. Specifikace tříd tedy vždy závisí na konkrétním problému, jenž potřebujeme vyřešit. [5, 6]

V této práci se bude dále pracovat pouze s NLI úlohou klasifikace dvojic vět do následujících tříd:

- *entailment* - implikace, hypotéza vychází z předpokladu
- *contradiction* - protiřečení, hypotéza vyvrací předpoklad nebo popisují každý jinou situaci
- *neutral* - neutrální, obě věty popisují stejnou situaci, ale nevyvrací se a nejedná se o implikaci

Zde jsou uvedeny příklady klasifikace několika párů vět z anglického SNLI datasetu:

Předpoklad **Hypotéza**	**Třída**
A soccer game with multiple males playing. Some men are playing a sport.	entailment
A man inspects the uniform of a figure in some Asian country. The man is sleeping.	contradiction
A smiling costumed woman is holding an umbrella. A happy woman in a fairy costume holds an umbrella.	neutral

Pro první dvojici vět platí, že hypotéza obecněji popisuje situaci v předpokladu. Nijak ji nevyvrací ani nepřidává nové informace. Jedná se tedy o *entailment*. Druhá dvojice vět obsahuje hypotézu, která nemůže být pravdivá zároveň s předpokladem, pokud oba popisují stejnou situaci. Hypotéza tedy vyvrací předpoklad a jedná se o *contradiction*.

V případě poslední dvojice dodává hypotéza nové informace o situaci popsané v předpokladu. Jelikož se nevyvrací, mohou obě věty popisovat stejnou situaci. Hypotéza však nevychází pouze z daného předpokladu a dvojice bude tedy zařazena do třídy *neutral*. [6]

Schopnost řešení úlohy NLI je důležitá součást sémantického porozumění přirozenému jazyku. Autoři SNLI datasetu označují tuto úlohu za užitečný test kvality sémantické reprezentace při vývoji NLU modelů, jelikož téměř všechny otázky o významu jazyka mohou být redukovány na kontextově závislou klasifikaci hypotéz.

V této práci se tedy dále budu zabývat pouze NLI úlohou a to jak pro anglický, tak především pro český text.

2.4 Anglické NLI datasety

Pro angličtinu existuje velké množství anotovaných NLI datasetů různé velikosti a obsahujících věty z mnoha odlišných zdrojů. Ne všechny datasety se však shodnou na konkrétních třídách, do kterých dvojice vět klasifikují.

Jak bylo zmíněno v předchozí podkapitole, zvolil jsem verzi NLI úlohy, která pracuje se třídami *entailment*, *contradiction* a *neutral*. Vybíral jsem tedy pouze ty datasety, jejichž anotace odpovídají této verzi.

Vybrané datasety jsem dále analyzoval a provedl předzpracování pro další použití.

2.4.1 Zvolené datasety

Z volně dostupných anotovaných NLI datasetů odpovídajících požadavkům jsem zvolil následující:

- **SNLI**

  Stanford Natural Language Inference (SNLI) dataset je soubor cca 570 000 párů lidmi psaných vět z mnoha různých zdrojů. Byl sestaven Stanford NLP Group v roce 2015. [6]

- **MultiNLI**

  Multi-Genre Natural Language Inference (MultiNLI) dataset je „"crowd-sourced"“ soubor cca 443 000 párů vět. Je obecnější než SNLI dataset a obsahuje mimo přímo psaný text i přepisy mluvených vět. Byl sestaven týmem z New York University v roce 2018. [7]

- **SICK**

  Sentences Involving Compositional Knowledge (SICK) dataset je soubor cca 10 000 párů vět obsahující popisky obrázků a videí, které byly rozšířeny o jejich další obdobné varianty. Byl sestaven týmem z University of Trento a Fondazione Bruno Kessler v roce 2016. [8]

- **Spojený SNLI a MultiNLI dataset**

  Autoři MultiNLI datasetu doporučují jeho spojení s SNLI datasetem do jednoho velkého. Sloučil jsem je tedy a pracoval s nimi jako se čtvrtým samostatným datasetem.

2.4.2 Ostatní datasety

Nalezl jsem i několik dalších NLI datasetů, jež však nesplňují zvolené požadavky na třídy a jsou zde uvedeny pouze pro úplnost:

- **PAWS**

  Paraphrase Adversaries from Word Scrambling (PAWS) dataset je soubor cca 100 000 anotovaných a 650 000 neanotovaných párů vět. Anotované věty jsou rozděleny pouze do dvou tříd: odlišný význam a parafráze. Dataset obsahuje věty z Wikipedie a služby Quora. Byl sestaven společností Google v roce 2019.

- **SciTail**

  SciTail dataset je soubor cca 27 000 párů vět získaných ze zkouškových otázek s výběrem odpovědí. Každá z odpovědí byla spojena s otázkou a formulována jako hypotéza. Předpoklady byly extrahovány z webových textů. Páry vět jsou rozděleny pouze do tříd *entailment* a *neutral*. Byl sestaven týmem z Allen Institute for Artificial Intelligence v roce 2018.

2.4.3 Analýza a předzpracování

Provedl jsem jednoduchou analýzu zvolených datasetů. Nejprve jsem pro každý dataset zjistil počet dat v trénovací (train), validační (dev) a testovací (test) množině.

	SNLI	MultiNLI	SICK
train	550152	392708	4439
dev	10000	10000	495
test	10000	10000	4906

Tabulka 1: Počet dat v množinách datasetů

Rozdělení jednotlivých dat do těchto množin je dáno přímo konkrétním datasetem. Nijak jsem tato rozdělení dále neměnil.

Poté jsem provedl odstranění nepoužitelných či přebytečných dat. Odstranil jsem páry, u kterých chyběla jedna z vět nebo štítek určující třídu. Dále jsem vyhledal páry, které se v jednotlivých datasetech vyskytují vícekrát, a ponechal je pouze jednou.

V případě spojeného SNLI a MultiNLI datasetu jsem zkontroval vzájemné duplicity. Žádnou jsem však nenalezl. Velikost tohoto datasetu je tedy součtem velikostí SNLI a MultiNLI.

V následující tabulce lze vidět výsledný počet dat v trénovací (train), validační (dev) a testovací (test) množině jednotlivých datasetů.

	SNLI	MultiNLI	SICK	SNLI+MultiNLI
train	548739	391027	4439	939766
dev	9840	9714	495	19554
test	9824	9830	4906	19654

Tabulka 2: Počet dat v množinách datasetů po předzpracování

Můžeme vidět, že počet dat v SNLI a MultiNLI datasetech klesl o více jak tisíc. Díky velikostem těchto datasetů pohybujících se ve stovkách tisíců, nemá však tato redukce příliš velký vliv na celkový počet dat.

V případě spojeného SNLI+MultiNLI datasetu je počet dat v trénovací množině téměř 1 milion. Velikosti validační a testovací množiny zde dosahují skoro 20 tisíc. Dá se tedy předpokládat, že modely strojového učení trénované na tomto datasetu budou dosahovat výrazně větší úspěšnosti než ostatní zvolené.

3 Formulace problému

Jak bylo zjištěno v předchozí kapitole, pro angličtinu existuje několik velkých anotovaných NLI datasetů. Dá se to samé říci i pro češtinu?

Po hledání vhodných volně dostupných českých datasetů jsem dospěl k závěru, že opak je pravdou. Několik NLI datasetů sice existuje (například ČTK datasety [9]), avšak obsahují řádově méně dat než ty anglické a zároveň se mi nepodařilo najít žádný, který by bylo možné použít pro dříve definovanou verzi úlohy.

Tento problém se netýká pouze konkrétně NLI úlohy. Pro češtinu je obecně k dispozici výrazně méně NLP datasetů než pro angličtinu a zároveň obsahují výrazně méně dat. Jelikož algoritmy strojového učení potřebují velké množství dat pro dosažení aplikovatelných výsledků, je tento nedostatek významným problémem.

Problémem je v porovnání s angličtinou malé množství volně dostupných anotovaných českých datasetů pro různé úlohy zpracování přirozeného jazyka. Tvorba nových datasetů a jejich anotace je však pracná a časově náročná činnost. Neexistuje jednodušší způsob jak získat nové české datasety?

Anotovaných NLP datasetů pro angličtinu existuje výrazně větší množství než pro češtinu. Nebylo by tedy možné využít tyto anglické datasety pro vytvoření českých?

Systémy pro automatický převod textu z jednoho jazyka do jiného jsou k dispozici již mnoho let. Otázka tedy zní:

Bylo by možné využít systémů strojového překladu pro generování ekvivalentních českých datasetů z anglických?

Tato práce se bude dále zabývat testováním validity tohoto hypotetického řešení. Testování zahrnuje překlad zvolených anglických NLI datasetů pomocí dvou různých systémů strojového překladu do češtiny a jejich následné použití pro natrénování českých předtrénovaných modelů strojového učení. Nejprve bude však vhodné natrénovat anglické modely pro získání referenční úspěšnosti.

Dále se pomocí analýzy výsledků těchto modelů pokusím zodpovědět otázku, zda by toto řešení mohlo být využitelné v praxi, nebo zda automatický překlad sníží kvalitu textu příliš výrazně a nebude tedy vhodné jej použít pro trénování modelů.

4 Vektorová reprezentace textu

Vstupem modelů strojového učení pro NLP je velmi často přirozený text. Jedná se však o matematické modely, které vnitřně pracují s vektory a maticemi, a nemohou text zpracovat přímo. Je třeba navrhnout nějaký algoritmus, jenž převede přirozený text na vektor či sekvenci vektorů, které lze dále použít jako vstup pro model.

Tato kapitola se zabývá problematikou rozložení textu na menší částí a jejich převodem na vektory pro použití v modelech strojového učení.

4.1 Tokenizace textu

Prvním krokem je rozdělení textu na jednotlivá slova. Tento proces se nazývá *tokenizace* a slouží k rozložení textu na malé části, které je dále možné převést na vektory. Tyto části jsou označovány jako *tokeny*.

Tokenizace zpravidla probíhá na základě slovníku tokenů. Algoritmus postupně rozkládá text na tokeny, buď na základě jejich pořadí ve slovníku, nebo pokročilejších heuristických metod. Vstupem tokenizéru je přirozený text a výstupem sekvence tokenů.

Existuje mnoho způsobů vytvoření slovníku. Je možné jej sestavit pouze ze slov extrahovaných z textů. Dnes se však nejvíce využívá *subword tokenizace*, tedy tokenizace na základě tokenů odpovídajících částem slov. [10]

4.1.1 Byte Pair Encoding (BPE)

Algoritmus BPE je jedním z hojně používaných způsobů generování tokenizačního slovníku založený na hledání co nejefektivnější reprezentace textu. Jelikož BPE pracuje obecně s posloupností bajtů, je možné jej použít i pro kompresi dat.

Principem BPE je postupné shlukování dvojic tokenů dle jejich četnosti v korpusu tvořeném souvislým textem. Základní myšlenkou je, že pokud se některé tokeny vyskytují pouze v určitých párech, je možné tyto jednotlivé tokeny nahradit jejich páry a tím snížit velikost slovníku tokenů. Algoritmus tedy hledá slovník o minimální velikosti.

Vytvoření slovníku pomocí BPE pracuje v následujících krocích:

1. **Vytvoření počátečního slovníku**

   Nejprve je třeba vytvořit počáteční slovník tokenů, jehož velikost se algoritmus dále bude pokoušet snížit. Obvykle se na počátku volí každý unikátní znak v sekvenci slov jako jeden token. Je možné místo znaků použít každý unikátní bajt (1 znak je tvořen jedním nebo více bajty).

2. **Zjištění četností slov**

   Text v korpusu je rozdělen na jednotlivá slova, nejčastěji dle mezer. Algoritmus poté spočte, kolikrát se každé slovo v korpusu vyskytuje, a výsledný seznam seřadí od nejčastějšího po nejméně časté. Každé slovo je zakončeno speciálním tokenem `</w>`.

3. **Zjištění počtů výskytů tokenů**

   Dále algoritmus spočte, kolikrát se každý token ze slovníku v korpusu vyskytuje. Obdobně jako u slov se i zde vytvoří seznam od nejčastějšího tokenu pod nejméně častý. Do seznamu je zahrnut i token `</w>`. Pokud má některý token 0 výskytů, je ze slovníku odstraněn.

4. **Spojení nejčastějších dvojic tokenů**

   Poté se prochází seznam slov seřazený dle výskytů a hledá se nejčastěji se vyskytující dvojice tokenů. Po jejím nalezení se tato dvojice přidá do slovníku jako samostatný token.

5. **Opakování**

   Algoritmus dále pokračuje krokem 3 dokud velikost slovníku neklesne pod požadovanou hodnotu nebo není překročen daný limit na počet iterací.

Nedostatkem BPE je fakt, že některé z výsledných tokenů se mohou překrývat a pro některá slova tedy může existovat více možností rozložení na jednotlivé tokeny. Dalším problémem může být rozdělení textu na slova, protože některé jazyky neoddělují slova mezerami. [10]

4.1.2 WordPiece

Algoritmus WordPiece funguje velice podobně jako BPE. Rozdílem je kritérium pro výběr dvojice tokenů ke spojení. Jak bylo zmíněno, BPE vybírá vždy nejčastěji se vyskytující dvojici. WordPiece však vybere tu dvojici, pro kterou je hodnota věrohodnostní funkce maximální.

Výpočet věrohodnostní funkce se může lišit dle implementace. Zpravidla však bere v úvahu poměr četnosti dvojice a četnosti samostatných tokenů. I přes rozdíly od BPE algoritmu, WordPiece se potýká s podobnými nedostatky. [10]

4.1.3 SentencePiece

Nedostatkem algoritmů BPE a WordPiece je nutnost rozdělení textového korpusu na jednotlivá slova, což může být problematické pro jazyky, jež neoddělují slova mezerami. Algoritmus SentencePiece využívá BPE, ale pro generování slovníku místo rozdělení na jednotlivá slova využívá rozdělení na jednotlivé unicode symboly. Mezera je také uvažována jako symbol.

SentencePiece je díky tomuto přístupu využitelný pro libovolný jazyk podporovaný kódováním unicode. Dále se ukázalo, že je rychlejší na výpočet než původní BPE a WordPiece. SentencePiece mimo upraveného BPE algoritmu podporuje i algoritmus založený na unigramovém modelu. Tyto odlišné přístupy jsou však nad rámec této práce. [10]

4.2 Vektorová reprezentace tokenů

Procesem tokenizace jsme získali sekvenci tokenů. Každý z těchto tokenů je dále třeba převést na vektorovou reprezentaci (*vector embedding*). Důležitý požadavek na vektorové reprezentace tokenů je jejich sémantický význam. Požadujeme, aby pro tyto vektory existovala míra podobnosti, jenž bude reprezentovat jejich významové vztahy.

Nejčastěji využívanou mírou podobnosti je kosinová podobnost, která je založena na úhlu mezi vektory:

$$sim(a, b) = \frac{a \cdot b}{\|a\| \|b\|}$$

Dalším důležitým požadavkem je nízká dimenze vektorů. Čím větší dimenze, tím výpočetně náročnější jsou operace s nimi. Příliš velké dimenze vektorů by proto nebyly praktické. [11, 12]

4.2.1 One-hot vektory

Nejjednodušším způsobem vektorové reprezentace jsou takzvané one-hot vektory. Dimenze vektorů je v tomto případě rovna velikosti slovníku. Jednotlivé vektory obsahují hodnotu 1 pouze na pozici odpovídající danému tokenu ve slovníku. Zbytek vektoru je vyplněn hodnotou 0.

I když se jedná o jednoduchý způsob reprezentace, není tato metoda vhodná pro úlohy NLP, jelikož není splněn ani jeden z výše uvedených požadavků. Vektor pro každý z tokenů je ortogonální vůči vektorům všech ostatních tokenů. Kosinová podobnost by tedy byla pro libovolné dvojice vždy nulová.

Také dimenze odpovídající velikosti slovníku je nepraktická a ideálně bychom chtěli nalézt reprezentaci o řádově nižší dimenzi vektorů. [11]

4.2.2 Word2vec

Způsob vektorové reprezentace lépe odpovídající kladeným požadavkům je word2vec. Jeho základem je matice o počtu řádků odpovídajícímu velikosti slovníku. Každá řádka této matice odpovídá vektoru reprezentujícímu daný token. Proces převodu tokenu na vektor je tedy pouhým výběrem příslušného řádku.

Dimenze vektoru závisí na volbě dimenze této matice a zpravidla se volí řádově menší než velikost slovníku. Čím významově podobnější si je určitá dvojice tokenů, tím vyšší bude jejich kosinová podobnost.

Samotný proces převodu tokenu na vektorovou reprezentaci pomocí word2vec je sice jednoduchý, ale pro jeho použití musíme nejprve získat již zmíněnou převodní matici označovanou jako *embedding matrix*.

Tato váhová matice je zpravidla inicializována náhodnými hodnotami a dále trénována společně s NLP modelem, ve kterém je použita. Druhým přístupem je tuto matici nejprve natrénovat jako součást velmi jednoduchého modelu, a poté je možné ji použít jako součást jiného složitějšího modelu.

Analýzy ukazují, že vektory získané z převodních matic trénovaných jako součást modelů strojového učení poskytují kvalitní reprezentaci významových vztahů. [12]

5 Modely strojového učení

Tato kapitola se zabývá využitím modelů strojového učení, především neuronových sítí, v NLP úlohách. Dále je zde popsán princip přeneseného učení, konkrétní anglické i české modely a implementace trénování těchto modelů.

5.1 Využití neuronových sítí pro NLP

První NLP model tvořen neuronovou sítí byl navržen již v roce 2001 pro úlohu predikce následujícího slova ze vstupního textu. Jedná se o *dopřednou neuronovou síť*, jejímž vstupem je vektor reprezentující text. Výstupem je vektor o velikosti odpovídající slovníku podporovaných slov. [13]

Výstupní vrstva neuronové sítě obsahuje aktivační funkci *softmax*. Tato operace převádí hodnoty v daném vektoru do rozmezí mezi 0 až 1 při zachování jejich relativních velikostí. Pokud je libovolný prvek vektoru menší než jiný libovolný prvek, bude menší i po aplikaci této funkce.

Funkce softmax pro z_i, tedy prvek i vektoru z o délce N, je vypočtena jako:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

Výstupní vektor tedy obsahuje hodnoty mezi 0 a 1, které odpovídají pravděpodobnosti, že dané slovo ve slovníku je následujícím slovem vstupního textu. Zvoleno bude dále slovo s maximální pravděpodobností.

Nevýhodou použití dopředné neuronové sítě je omezení na délku vstupního textu. Síť zpracovává celý text současně a každý text zpracovává nezávisle. Dále se proto začaly využívat *rekurentní neuronové sítě* a *sítě s krátkodobou pamětí*. Tyto modely umožňují zpracovávat vstupní text postupně, jelikož si uchovávají informace o určitém množství předchozích kroků. Často se u tohoto přístupu využívá dvou modelů. Dopředný model zpracovává vstupní sekvenci od začátku a zpětný postupuje od konce, čímž je umožněno uvažování levého i pravého kontextu.

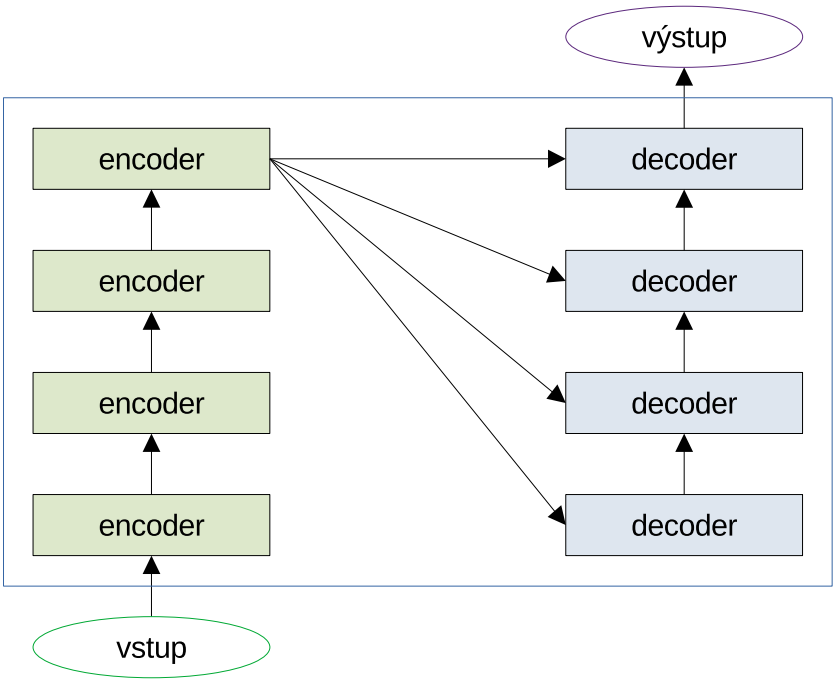
Nevýhodou tohoto přístupu je však nízká možnost paralelizace, jež vede k časově náročnému zpracování. Dalším využívaným modelem byly tedy *konvoluční neuronové sítě*, umožňující výrazně lepší paralelizaci. Vstupem této sítě je matice, jejímiž řádky jsou vektorové reprezentace slov ve vstupní sekvenci. Síť může dále zpracovávat různé části této matice současně.

Výše zmíněné přístupy však mají jeden velký nedostatek. Pracují pouze s omezeným lokálním kontextem namísto uvažování kontextu celého textu. Byly proto vyvinuty modely pro zpracování celé vstupní sekvence najednou a postupného generování nové sekvence při uvažování všech předchozích generovaných prvků. Takzvané *sequence-to-sequence* modely uvažují veškerý kontext, ukázaly se proto jako velmi vhodné pro NLP úlohy.

Existuje mnoho různých *sequence-to-sequence* architektur. Některé využívají sítí s krátkodobou pamětí, jiné obsahují konvoluční sítě nebo rekurentní sítě. Architekturou vykazující velmi kvalitní výsledky je takzvaný *transformer*, jenž bude dále popsán podrobněji. [14, 15, 16]

5.2 Transformer

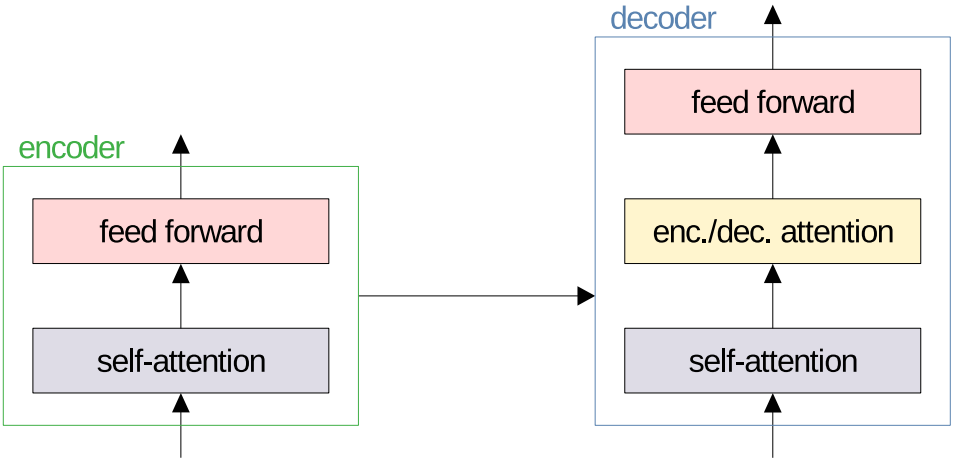
Transformer je *sequence-to-sequence* model strojového učení využívající metody zvané *attention* pro rychlejší a paralelní zpracování. V této sekci je popsána struktura a princip transformeru přizpůsobeného pro úlohy NLP.



Obrázek 1: Základní struktura transformeru

Strukturu transformeru lze rozdělit na dvě hlavní části. Vstupem je tokenizovaný text reprezentován sekvencí vektorů se zakódovanou pozicí, jenž je přiveden na *enkodér*.

Druhou část představuje *dekodér*, jehož výstupem je také sekvence vektorů, kterou lze poté převést znovu na text. Transformer pro NLP je tedy obecně model, generující nový text v závislosti na vstupním textu. [16]



Obrázek 2: Struktura jednoho bloku enkodéru a dekodéru

5.2.1 Enkodér

První část transformeru je tvořena několika identickými po sobě jdoucími bloky. Každý z těchto bloků obsahuje dvě vrstvy: *self-attention* a jednoduchou plně propojenou dopřednou neuronovou síť (*feed forward*).

Na výstupu každé vrstvy dochází k součtu s jejím vstupem (reziduální spoje) a následné normalizaci. Vstupem prvního bloku je sekvence vektorů se zakódovanou pozicí. Proces enkodéru není rekurentní, přikládáme celou sekvenci najednou. Poté je výstup každého bloku vstupem bloku následujícího a celkovým výstupem enkodéru je tedy výstup posledního bloku. [16]

5.2.2 Dekodér

Druhá část transformeru je podobně jako enkodér tvořena identickými bloky, kde výstup každého bloku je vstupem toho následujícího. Každý blok obsahuje tři vrstvy: *self-attention*, *encoder-decoder attention* a plně propojenou dopřednou neuronovou síť.

Na výstupu posledního bloku je navíc jednovrstvá neuronová síť s aktivační funkcí softmax, jejímž výstupem je vektor o délce odpovídající velikosti slovníku. Celkovým výstupem dekodéru je poté token odpovídající maximální hodnotě v tomto vektoru. V principu se jedná o klasifikaci do více tříd.

Obdobně jako u enkodéru dochází i zde na výstupu každé vrstvy k součtu s jejím vstupem a normalizaci. Účelem *encoder-decoder attention* vrstev je využití celkového výstupu enkodéru v procesu dekódování.

Jelikož dekodér pracuje rekurentně, jsou vstupem prvního bloku celkové výstupy v předchozích krocích, převedeny na jejich vektorovou reprezentaci. Transformer tedy v každém kroku bere v potaz vygenerované tokeny ze všech předchozích kroků a vygeneruje token nový. Celkovým výstupem transformeru je poté sekvence tokenů v jednotlivých krocích.

Vhodným příkladem práce dekodéru je generování textu. V prvním kroku je vygenerován první token výstupního textu. Tento token je poté přiveden na vstup a dekodér vygeneruje následující token. Dále přivedeme oba tokeny znovu na vstup a vygenerujeme třetí token. Takto pokračujeme dokud nevygenerujeme celý text. [16, 17]

5.2.3 Zakódování pozice

Jak bylo již zmíněno, vstupem transformeru je sekvence vektorových reprezentací tokenů. Jelikož čte celou vstupní sekvenci najednou, nemá model žádnou informaci o pořadí jednotlivých tokenů. Je proto třeba tuto informaci předat explicitně, tedy zakódováním pozice přímo do jejich reprezentace.

Pozice je zakódována jako vektor o stejné velikosti jako reprezentace tokenu. Způsob zakódování může být různý. V prvotním návrhu transformeru byly použity sinusoidy o frekvenci závislé na absolutní pozici.

Vektor pozice a vektor tokenu jsou poté sečteny pro získání výsledné vektorové reprezentace. Zakódování pozice se provádí nejen na vstupu enkodéru, ale i na vstupu dekodéru v každém jeho kroku. [16]

5.2.4 Attention

Metoda nazývaná *attention* je základním prvkem funkce transformeru. Princip této metody spočívá v rozložení sekvence vstupních vektorů na matice Q (query), K (key) a V (value).

Uvažujme matici X, jejíž řádky jsou jednotlivé vstupní vektory. Dále uvažujme váhové matice W^Q, W^K a W^V. Hodnoty váhových matic jsou součástí trénovatelných parametrů modelu. Matice Q, K a V poté spočteme jako:

$$Q = X \cdot W^Q$$
$$K = X \cdot W^K$$
$$V = X \cdot W^V$$

Pomocí těchto hodnot dále vypočteme attention matici Z pomocí váženého skalárního součinu. Konstanta d_k odpovídá počtu sloupců matice K.

$$Z = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V$$

Jelikož každá z trojice matic Q, K a V je spočtena ze stejného vstupu X, nazývá se tato metoda konkrétně *self-attention*.

Význam matice Z lze vysvětlit na příkladu prvního bloku enkodéru, vstupem jehož self-attention vrstvy je přímo vstupní sekvence vektorových reprezentací tokenů. Každý z řádků matice X tedy reprezentuje jednotlivý token.

Výsledkem součinu $Q \cdot K^T$ je čtvercová matice o dimenzi odpovídající délce vstupní sekvence. Hodnotou na pozici i, j je skóre určující důležitost významu tokenu i pro význam tokenu j. Operace softmax převede tato skóre na hodnoty mezi 0 a 1.

Jednotlivé řádky matice Z jsou *attention vektory* tokenů. Podobně jsou řádky matice V *value vektory* jednotlivých tokenů.

Attention vektor každého tokenu je dán váženou sumou value vektorů všech tokenů. Jednotlivé příspěvky value vektorů tokenů do attention vektoru daného tokenu jsou tedy ovlivněny hodnotou skóre, jež určuje velikost významového vztahu mezi těmito tokeny.

Zatím jsme uvažovali, že pro vstupní matici X vypočteme pouze jednu attention matici Z. Transformer model však počítá několik matic $Z_0, Z_1, ..., Z_H$, každou použitím různých váhových matic W_h^Q, W_h^K a W_h^V, kde H je počet „hlav“ modelu. Tento přístup se nazývá *multi-head attention*.

Jednotlivé matice jsou poté spojeny pomocí konkatenace a vynásobeny váhovou maticí W^O, jejíž hodnoty jsou také součástí trénovatelných parametrů, čímž získáme celkovou attention matici.

$$Z = [Z_0, Z_1, ..., Z_H] \cdot W^O$$

Tento přístup nám umožňuje zachytit více vztahů mezi tokeny než použití pouze jedné matice. Jelikož výpočty jednotlivých attention matic nejsou na sobě závislé, máme možnost je provádět paralelně, čímž dojde k výrazné redukci výpočetní náročnosti.

Mimo self-attention vrstev obsahuje transformer i *encoder-decoder attention* vrstvy. Ty pracují podobně, avšak ze vstupu X počítají pouze matici Q. Matice K a V jsou převzaty ze self-attention vrstvy posledního bloku enkodéru. Počítané skóre tedy reprezentuje významové vztahy mezi celkovým výstupem z enkodéru a vstupem dané vrstvy v dekodéru. [17]

5.2.5 Trénování

Proces trénování transformeru je obdobný trénování obecných neuronových sítí. Jedná se o proces učení s učitelem, tedy postupné předkládání trénovacích dvojic (vstup a požadovaný výstup). Modelu předložíme vstup a zjistíme jeho reálný výstup, jenž poté porovnáme s požadovaným výstupem a na základě ztrátové funkce aktualizujeme parametry modelu. Mezi trénované parametry patří i hodnoty převodní matice pro vektorovou reprezentaci.

Při trénování neuvažujeme jako reálný výstup transformeru sekvenci tokenů, ale sekvenci vektorů před jejich převedením na konkrétní tokeny určením maxima, tj. přímo po *softmaxu*. Tyto vektory o délce odpovídající velikosti slovníku obsahují hodnoty mezi 0 a 1. Jako požadovaný výstup uvažujeme sekvenci one-hot vektorů, tedy vektorů s hodnotou 1 na prvku odpovídající požadovanému tokenu a 0 všude jinde.

Rozdíl mezi reálnou a požadovanou sekvencí můžeme získat pouhým odečtením jednotlivých vektorů. V praxi se však vektory často porovnávají jako dvě pravděpodobnostní rozložení, například pomocí křížové entropie (*cross-entropy*).

Ať už je způsob výpočtu ztrátové funkce jakýkoliv, její hodnota je využita algoritmem *backpropagation* pro výpočet gradientu ztrátové funkce dle parametrů transformeru.

Důležitou hodnotou při trénování modelů je míra učení (*learning rate*). Tato hodnota určuje míru, se kterou jsou přepočítávány parametry modelu. Větší hodnota má za následek prudší změny parametrů a tím pádem rychlejší, avšak potencionálně méně stabilní, trénování.

Základní implementace algoritmu *backpropagation* upravuje parametry modelu pouze na základě velikosti gradientu ztrátové funkce a míry učení. Lze však využít některý z optimalizačních algoritmů pro větší kontrolu nad změnou parametrů.

Často používaným optimalizátorem je ADAM. Tento algoritmus upravuje míru učení pro každý parametr zvlášť v průběhu trénování na základě exponenciálního klouzavého průměru změny a její druhé mocniny. Volitelné parametry β_1 a β_2 určují míru poklesu těchto průměrů. Dalším parametrem algoritmu je velmi malé číslo ϵ pro náhradu nuly při dělení. [16, 17, 18]

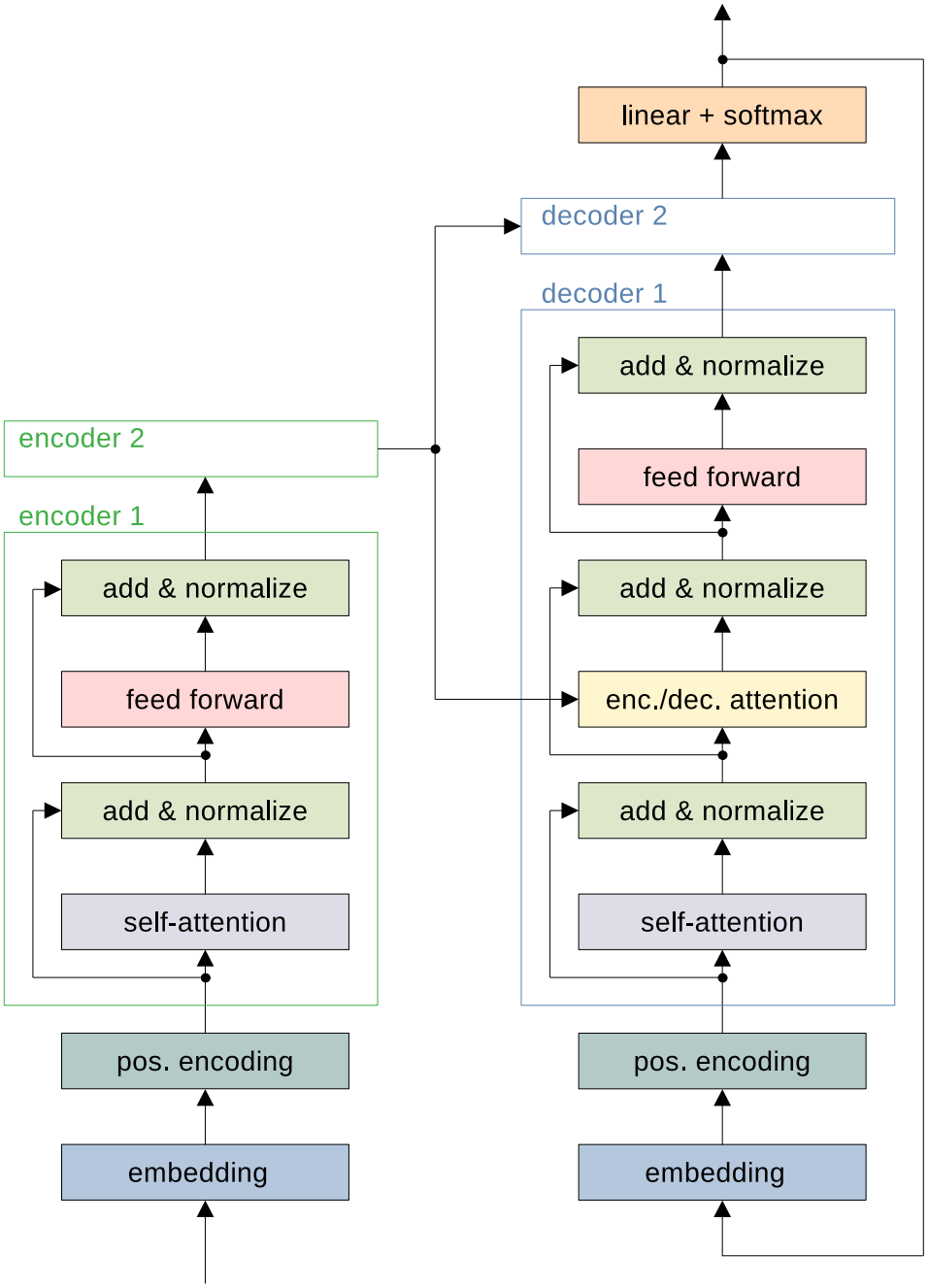
5.2.6 Výhody a využití

Transformer architektura řeší mnoho problémů spojených s použitím dřívějších přístupů. Pro rekurentní sítě je počet kroků potřebných pro zpracování sekvence úměrný její délce a není je možné počítat paralelně. Transformer zpracovává celou sekvenci najednou v konstantním čase a přináší široké možnosti paralelizace.

Možnosti paralelizace má i konvoluční síť. Ta však podobně jako rekurentní síť uvažuje pro zpracování dané části textu vždy pouze úzký kontext. Transformer dokáže díky attention metodě uvažovat vztahy mezi slovy v celém textu, což vede ke kvalitnějšímu porozumění.

Některé problémy ovšem stále přetrvávají. Délka vstupní sekvence tokenů je omezená a není proto možné zpracovat text přesahující určitý počet tokenů. Omezený je zároveň i vstup dekodéru, což vede na omezení délky výstupní sekvence.

Transformer modely je možné využít v mnoha NLP úlohách. Některé (například strojový překlad) využívají celou strukturu transformeru. Ku příkladu pro úlohy klasifikace je však potřeba pouze enkodér. Úlohy generování přirozeného jazyka využívají pouze dekodér. Základní transformer architektura se tedy zpravidla upravuje v závislosti na typu řešené úlohy. [16]



Obrázek 3: Detailní struktura transformeru

5.3 Přenesené učení

Běžný přístup k trénování modelů strojového učení zahrnuje inicializaci náhodnými parametry a následné trénování daty související s řešenou úlohou. Model tedy trénujeme vždy „od nuly“.

Metoda přeneseného učení (*transfer learning*) navrhuje místo náhodné inicializace využít jako počáteční bod model natrénovaný na podobné úloze. Jedná se o přenos znalostí z jedné úlohy na jinou, proto název „přenesené učení“.

Tento přístup lze využít v případech, kdy obecné znalosti naučené na datech jedné úlohy jsou přenositelné na řešenou úlohu. Tuto přenositelnost je většinou těžké určit a je třeba ji ověřit experimentálně.

Existují dva hlavní přístupy k přenesenému učení:

- **Vývoj modelu**

  1. **Zvolení úlohy**
     Nalezení úlohy s velkým množstvím dostupných trénovacích dat, u které lze předpokládat přenositelnost znalostí na řešenou úlohu.

  2. **Sestavení a trénování modelu**
     Sestavení modelu vhodného pro zvolenou úlohu a jeho následné natrénování na vybraných datech.

  3. **Využití modelu pro řešenou úlohu**
     Využití natrénovaného modelu jako počátečního bodu pro řešenou úlohu. Můžeme využít celý model nebo pouze jeho části.

  4. **Dotrénování modelu**
     Trénování modelu na datech souvisejících s řešenou úlohou.

- **Využití předtrénovaného modelu**

  1. **Zvolení modelu**
     Nalezení modelu předtrénovaného na obecných úlohách podobných té řešené. Existuje mnoho volně dostupných obecných modelů trénovaných na velkých datasetech. Počet vhodných modelů však závisí na konkrétní řešené úloze.

2. **Využití modelu pro řešenou úlohu**
   Využití zvoleného předtrénovaného modelu jako počátečního bodu pro řešenou úlohu. Můžeme využít celý model nebo pouze jeho části.

3. **Dotrénování modelu**
   Trénování modelu na datech souvisejících s řešenou úlohou (*fine-tuning*).

Je zřejmé, že využití již existujícího předtrénovaného modelu je méně pracné a časově náročné než vývoj vlastního. Tento přístup je proto hojně využívaný.

Využití přeneseného učení může přinést mnoho výhod. Předtrénovaný model má obecně lepší výsledky při počátku trénování než náhodně inicializovaný model. Při dobré přenositelnosti znalostí bude rychlost trénování na řešené úloze vyšší. [19]

5.4 Předtrénované modely pro NLI

Rozhodl jsem se využít výhod přeneseného učení a vyhledat několik předtrénovaných modelů pro obecné NLU. Zvolil jsem 3 anglické modely a jeden český. Tyto modely mají velice podobnou strukturu, liší se však v počtu parametrů, tokenizaci či způsobu předtrénování.

Jedná se o transformer modely obsahující pouze enkodér. Jsou proto vhodné pro klasifikaci, tedy i pro úlohu NLI, ne však pro generování textu.

5.4.1 Anglické modely

Z anglických modelů jsem zvolil následující:

- **BERT**
  Bidirectional Encoder Representations from Transformers (BERT) je model vytvořen společností Google v roce 2018. Byl předtrénován na úloze *masked language modeling* (doplnění chybějících slov do vět) a *next sentence prediction* (rozhodování, zda určitá věta dává smysl jako následující k jiné větě). Obsahuje přibližně 110 milionů trénovatelných parametrů. Pro tokenizaci využívá algoritmus WordPiece s inicializací slovníku unikátními znaky. [20]

- **BERT-Large**

  BERT-Large je rozšířená verze BERT modelu s mnohem více trénovatelnými parametry (přibližně 330 milionů). Teoreticky by měl tedy dosahovat lepších výsledků, je však výpočetně a paměťově náročnější. [20]

- **RoBERTa**

  Robustly Optimized BERT Pretraining Approach (RoBERTa) je model o stejné struktuře jako BERT. Je však předtrénován pouze na úloze *masked language modeling* a odlišných datech. Odlišný je i algoritmus tokenizace, jelikož využívá BPE s inicializací slovníku unikátními bajty. Obsahuje přibližně 120 milionů trénovatelných parametrů. [21]

5.4.2 České modely

Z českých modelů jsem zvolil pouze jediný:

- **FERNET-C5**

  FERNET-C5 je český model vytvořen v roce 2021 Katedrou kybernetiky Západočeské univerzity v Plzni. Má stejnou strukturu jako BERT, ale používá algoritmus SentencePiece pro tokenizaci. Obdobně jako BERT byl i tento model předtrénován na úloze *masked language modeling* a *next sentence prediction* pomocí vět získaných z webu. [22]

5.5 Implementace trénování

Skripty pro předzpracování datasetů, trénování modelů a analýzu výsledků jsem implementoval v programovacím jazyce Python 3. Tento jazyk jsem zvolil, jelikož jsou pro něj k dispozici kvalitní knihovny a nástroje pro trénování modelů strojového učení. Cílem bylo vytvoření skriptu pro natrénování zvolených modelů na daných předzpracovaných datasetech.

5.5.1 Služby a nástroje

Pro jednoduchost implementace trénování modelů jsem využil následujících služeb a nástrojů:

- **MetaCentrum**

  Virtuální organizace MetaCentrum, spravována sdružením CESNET, umožňuje bezplatné využití výpočetní kapacity a úložného prostoru studentům a akademickým pracovníkům. Jelikož trénování modelů je výpočetně i časově náročné, využil jsem této možnosti a spouštěl veškeré skripty vzdáleně na serverech poskytnutých touto službou. [23]

- **TensorFlow**

  Platforma TensorFlow je určena pro implementaci algoritmů strojového učení a jejich trénování. Poskytuje algoritmy pro předzpracování dat a vytváření, trénování a aplikaci modelů. TensorFlow je k dispozici pro Python a JavaScript. Umožňuje i práci s předtrénovanými modely a jednalo se tedy o vhodnou platformu pro implementaci trénování mnou zvolených modelů. [24]

- **Hugging Face**

  Společnost Hugging Face poskytuje nástroje a služby pro vývoj aplikací strojového učení. Pro mě užitečná byla její rozsáhlá volně dostupná knihovna předtrénovaných transformer modelů, která obsahuje všechny mnou zvolené modely. Využil jsem jejího *transformers* modulu pro Python ke stažení těchto modelů ve formátu určenému pro TensorFlow. [25]

5.5.2 Verze použitých modelů

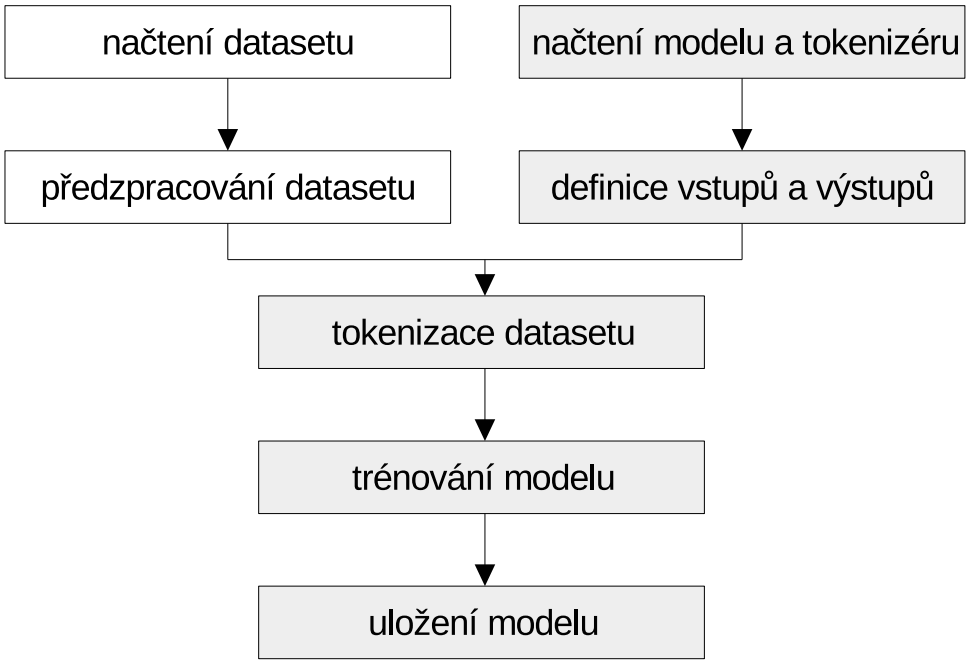
Zde jsou uvedeny konkrétní verze použitých předtrénovaných modelů:

- **BERT**: `https://huggingface.co/bert-base-uncased`
- **BERT-Large**: `https://huggingface.co/bert-large-uncased`
- **RoBERTa**: `https://huggingface.co/roberta-base`
- **FERNET-C5**: `https://huggingface.co/fav-kky/FERNET-C5`

5.5.3 Postup trénování

Pro trénování modelů jsem vytvořil obecný skript, jenž lze použít pro trénování libovolného ze zvolených modelů na libovolném ze zvolených datasetů. Některé části tohoto skriptu jsou specifické pro konkrétní modely, většina je však jednotná.

Skript je rozdělen do několika částí tvořících jednotlivé kroky procesu trénování. Vstupem je název datasetu a modelu. Výstupem je natrénovaný model uložený jako soubor a informace o procesu trénování.



Obrázek 4: Postup trénování modelů

Načtení a předzpracování datasetů bylo již popsáno v kapitole 2.4 a nebudu se proto těmto krokům zde věnovat. Zbývající kroky zahrnují:

1. **Načtení modelu a tokenizéru**

   Prvním krokem bylo načtení předtrénovaného modelu a příslušného tokenizéru. Stažení modelů a tokenizérů i jejich načítání ve skriptu jsem implementoval pomocí již zmíněného modulu *transformers*.

2. **Definice vstupů a výstupů**

   U použitých modelů je k dispozici mnoho různých vstupů a výstupů. Je tedy nejprve třeba definovat, na jaké vstupy bude přiveden tokenizovaný vstupní text a jaký výstup bude uvažován jako celkový výstup modelu.

   Jelikož jsou všechny zvolené modely založeny na stejné architektuře, použil jsem u všech stejnou podobu vstupů a výstupů. Pro NLI úlohu je u BERT architektury vhodné použít následující vstupy:

   - `input_ids`

     Jedná se o vektor s číselnými identifikátory tokenů první a druhé věty. Z důvodu dosažení konstantní velikosti pro libovolné délky vět je zbytek vektoru doplněn nulami.

   - `attention_mask`

     Attention maska je vektor s číslem 1 na pozicích odpovídajících oběma větám a číslem 0 na prázdném zbytku vektoru. Díky této masce je model schopen ignorovat nedůležitou část `input_ids` vektoru.

   - `token_type_ids`

     Třetí část vstupu je vektor s číslem 0 na pozicích odpovídajících tokenům první věty a číslem 1 na pozicích odpovídajících tokenům druhé věty. Tento vektor předává modelu informaci o tom, jaká část `input_ids` odpovídá které větě.

   Každou vstupní dvojici vět je potřeba převést na tyto tři vektory, aby ji model dokázal správně zpracovat.

   Z výstupů jsem využil pouze `pooler_output`, jenž je výstupem poslední vrstvy modelu. Jelikož NLI je klasifikační úloha, je třeba za tento výstup vložit plně propojenou vrstvu o počtu neuronů odpovídajícímu počtu tříd (v tomto případě 3). Tato vrstva je inicializována náhodnými hodnotami a není tedy předtrénována jako zbytek modelu.

   Celkovým výstupem modelu je výstup z této vložené vrstvy, na který je dále aplikována aktivační funkce softmax. Obsahuje tedy pro každou třídu hodnotu pravděpodobnosti, s jakou do ní daná dvojice vět náleží.

3. **Tokenizace datasetu**

   Dalším krokem byla tokenizace všech dvojic vět v datasetu. Jelikož každý z použitých modelů má vlastní dostupný tokenizér, nebylo nutné tokenizaci textu implementovat ručně.

   Každý token má vlastní unikátní číselný identifikátor. Výstupem tokenizéru pro každou větu je vektor tokenů reprezentovaných těmito identifikátory, doplněný o speciální tokeny označující začátek a konec sekvence. Tyto dvojice vektorů jsem poté převedl na již zmíněné vstupní vektory pro model.

4. **Trénování modelu**

   Samotné trénování modelu probíhalo na trénovací množině daného datasetu. Proces trénování je již součástí platformy TensorFlow a implementace tedy byla jednoduchá. Modelu byly postupně předkládány trénovací dvojice obsahující vstupní vektory odpovídající dvojicím vět a požadované výstupy.

   Každý model jsem trénoval v několika epochách, kdy jedna epocha odpovídá přiložení všech trénovacích dvojic z datasetu. Po skončení každé epochy jsem ověřil přesnost na validační množině daného datasetu, abych se ujistil, zda trénování probíhá správně.

5. **Uložení modelu**

   Posledním krokem bylo uložení natrénovaného modelu. Jelikož se při procesu trénování mění pouze hodnoty vah, nebylo třeba ukládat celou strukturu modelu, ale pouze všechny jeho parametry. TensorFlow umožňuje uložení parametrů modelu do souboru ve svém *checkpoint* formátu.

Tímto způsobem jsem realizoval trénování anglických i českých modelů. Více informací o trénování konkrétních modelů, jejich testování a analýze výsledků je uvedeno v kapitole 7.

6 Strojový překlad textu

Následující kapitola se věnuje problematice strojového překladu přirozeného textu. Zabývá se různými obecnými přístupy a také konkrétními online službami. Dále je zde popsán postup překladu NLI datasetů zvolených v kapitole 2.4 do češtiny a jejich analýza.

6.1 Princip a realizace

Strojový překlad je proces automatického přeložení textu z jednoho jazyka do jiného. Systém provádějící tento překlad se běžně nazývá „překladač“. Vstupem překladače je zpravidla text v daném jazyce a výstupem je text přeložený do jiného jazyka. Pokud program podporuje více jazyků, může být vstupem i specifikace vstupního a výstupního jazyka.

Existuje mnoho přístupů k realizaci strojového překladu. Mezi nejvíce rozšířené patří překlad založený na pravidlech, statistický přístup, hybridní přístup a v současnosti především překlad pomocí neuronových sítí.

6.1.1 Překlad založený na pravidlech

Nejstarší způsob strojového překladu je proces založený na pravidlech (*Rule-Based Machine Translation*, RBMT), využívající gramatické, syntaktické a sémantické informace o vstupním a výstupním jazyce pro překlad textu.

Na vstupní text je aplikována morfologická analýza, rozdělující větu na jednotlivé části na základě pravidel vstupního jazyka. Tyto části jsou poté přeloženy pomocí slovníku a předány morfologickému generátoru, jenž na základě pravidel výstupního jazyka vygeneruje výstupní text.

Nevýhodami tohoto přístupu je nutnost existence kvalitního slovníku pro překlad slov mezi požadovanými jazyky a nutnost manuálního ladění pravidel. Přeložený text také často působí nepřirozeně z důvodu malého zaměření na kontext překládaných slov, což může i vést k velkým chybám překladu.

Možným zlepšením RBMT přístupu je přeložení vstupního textu nejprve do umělého „neutrálního“ jazyka a z něj text přeložit do jazyka výstupního. Velkým problémem je však proces definice vhodného „neutrálního“ jazyka. [26]

6.1.2 Statistické metody překladu

Statistické metody překladu (*Statistical Machine Translation*, SMT) jsou založeny na modelování podmíněné hustoty pravděpodobnosti $p(e|f)$, kde f je vstupní text a e výstupní. Algoritmus zvolí takový překlad e^* z dostupného korpusu, pro který je hodnota této pravděpodobnosti maximální.

Zpravidla se $p(e|f)$ nemodeluje přímo, ale rozkládá se dle Bayesova vztahu:

$$e^* = \arg\max_e(p(e|f)) = \arg\max_e(p(f|e) \cdot p(e))$$

Zde je $p(e)$ apriorní pravděpodobností textu e určující pravděpodobnosti jeho výskytu ve výstupním jazyce a $p(f|e)$ určuje pravděpodobnost, že vstupní text f je překladem textu e.

Výhodou SMT je, že narozdíl od RBMT nepotřebuje dvojjazyčné slovníky a explicitní pravidla. Veškeré znalosti jsou získané z dvojjazyčného korpusu obsahující souvislý text a jeho překlad.

Ovšem i tento přístup má řadu nevýhod. Sestavení kvalitního korpusu je pracné a časově náročné. Dále samotné nalezení maximální pravděpodobnosti je velmi výpočetně náročné, proto je důležité použít heuristické metody prohledávání, které mohou vést ke snížení kvality. Překlady SMT se také ukázaly být méně přesné pro jazyky s velmi rozlišným slovosledem. [26]

6.1.3 Hybridní metody překladu

Spojením RBMT a SMT přístupů lze dosáhnout kvalitnějšího překladu než použitím pouze jednoho z nich. Tento hybridní přístup (*Hybrid Machine Translation*, HMT) využívá pravidla pro předzpracování vstupního textu a následně využije SMT pro samotný překlad. Některé HMT přístupy provedou překlad nejprve pomocí RBMT a použijí SMT pro následnou úpravu.

Hybridní přístup se ukázal jako mnohem flexibilnější než samotný RBMT nebo SMT. Stále však vyžaduje kvalitní slovník a korpus. [26]

6.1.4 Překlad pomocí neuronových sítí

Nejnovějším přístupem ke strojovému překladu textu je překlad pomocí neuronových sítí (*Neural Machine Translation*, NMT). Výhodou neuronových sítí je možnost učení přímo na dvojicích originálního a přeloženého textu bez nutnosti slovníku nebo náročného prohledávání korpusu.

Pro kvalitní trénování NMT modelu je však třeba velkého počtu trénovacích dvojic a samotný proces trénování je velmi výpočetně náročný. Z počátku byla významným problémem nízká spolehlivost, především u méně běžných slov a frází. Oblast NMT však za poslední roky prošla značným vývojem a mnoho těchto problémů bylo redukováno. Dnes z existujících přístupů ke strojovému překladu dosahuje NMT nejkvalitnějšího překladu izolovaných přirozených vět.

Modely strojového překladu jsou zpravidla tvořeny dvěma částmi. První část slouží pro zpracování vstupního textu a jeho převedení do takzvané „hluboké reprezentace“. Druhá část převede tuto reprezentaci na výstupní text v daném jazyce. Tato obecná reprezentace není explicitně definována, ale naučena při procesu trénování modelu.

Dříve se pro obě části používaly rekurentní neuronové sítě. Jejich významná nevýhoda, tedy neschopnost v každém kroku uvažovat kontext celého textu, vedla k přechodu na *sequence-to-sequence* modely, jejichž princip a výhody byly popsány v kapitole 5.

V procesu strojového překladu je vstupní text zpracován enkodérem a dekodér vygeneruje výstupní text. Enkodér a dekodér odpovídají výše zmíněné první a druhé části modelu. Využití *sequence-to-sequence* modelů vedlo k výraznému zlepšení schopnosti porozumění kontextu v úloze překladu. [27, 28]

6.2 Online překladače

Existuje mnoho online služeb poskytujících možnost překladu libovolného textu mezi některými z podporovaných jazyků. Tyto online překladače umožňují uživateli zadat vstupní text a zvolit vstupní a výstupní jazyk. Server poté uživateli zašle překlad generovaný některým z algoritmů strojového překladu.

Z volně dostupných online překladačů jsem pro další využití v této práci zvolil následující:

- **Google Translate**

  Online služba Google Translate umožňuje strojový překlad mezi více než 100 jazyky. První verze překladače byla vyvinuta společností Google v roce 2006 a využívala SMT přístupu s velkým množstvím gramatických nepřesností.

V roce 2016 přešel Google Translate na NMT přístup a nyní využívá vlastních *sequence-to-sequence* modelů GNMT využívajících WordPiece algoritmus pro tokenizaci. Využitím NMT přístupu se kvalita překladu výrazně zlepšila. Překladač také umožňuje automatickou identifikaci vstupního jazyka.

Google překladač jsem zvolil především pro jeho popularitu a dlouhou historii. [27, 29]

- **LINDAT Translation**

  Pro online překladač LINDAT Translation se počet podporovaných jazyků pohybuje pouze v řádu jednotek. Překladač využívá transformer modelů CUBBITT, jež byly navrženy a natrénovány týmem z Univerzity Karlovy v roce 2020.

  Dle tvrzení autorů jsou anglicko-české překlady LINDAT překladače přirozenější a podobnější lidskému překladu než v případě Google překladače, čehož dosáhli použitím velkého množství kvalitních trénovacích dat. Velké trénovací datasety byly získány doplněním lidmi přeložených korpusů o strojově přeložené korpusy. Strojově přeložené trénovací dvojice byly převráceny a obsahovaly tedy překlad jako vstup a originál jako výstup.

  LINDAT překladač jsem zvolil především z důvodu zmíněné vysoké přirozenosti anglicko-českého překladu. [28]

Při volbě překladačů pro mě byla důležitá schopnost překladu textu z angličtiny do češtiny a možnost automatizace procesu překladu většího počtu textů. Oba zvolené překladače tyto požadavky splňují.

6.3 Implementace překladu

Stejně jako u trénování modelů, jsem i skript pro překlad datasetů implementoval v jazyce Python 3. Cílem bylo vytvoření skriptu pro překlad zvolených datasetů pomocí LINDAT Translation a Google Translate.

6.3.1 Služby a nástroje

Pro implementaci překladu datasetů jsem využil následujících služeb a nástrojů:

- **deep-translator**

  Jedná se o knihovnu pro Python umožňující komunikaci s několika online překladači za účelem automatického přeložení textu. Umožňuje jednoduché přeložení textu pomocí libovolného z podporovaných překladačů bez nutnosti ruční implementace komunikace. Jelikož mezi tyto překladače patří i Google Translate, rozhodl jsem se tuto knihovnu použít. Překladač LINDAT tato knihovna nepodporuje. [30]

- **LINDAT Translation API**

  Překladač LINDAT poskytuje veřejně dostupné rozhraní pro překlad textu. Toto API není omezeno počtem žádostí ani počtem znaků a je tedy vhodné i pro překlad větších datasetů. Komunikace je realizována přes HTTPS protokol. Je možné poslat žádost obsahující text, jazyk tohoto textu a jazyk, do kterého chceme text přeložit. Rozhraní poté pošle odpověď obsahující přeložený text. [31]

6.3.2 Postup překladu

Pro přeložení datasetů jsem vytvořil obecný skript, jenž lze použít pro přeložení libovolného ze zvolených datasetů jedním ze zvolených překladačů. Vstupem je název datasetu a překladače. Výstupem je přeložený dataset uložený do souboru a informace o případných chybách v průběhu překladu.

Nejprve bylo nutné se rozhodnout, jakým způsobem budu dataset posílat danému překladači. Oba zvolené překladače jsou určeny pro překlad souvislého textu a ne dlouhého seznamu dvojic vět. Uvažoval jsem tři možné způsoby:

1. přeložit každou větu zvlášť
2. rozdělit dataset na menší části a přeložit vždy velké množství vět najednou jako souvislý text
3. přeložit zvlášť každou dvojici vět jako souvislý text

Možnost 1 se zdála jako nejjednodušší, tedy postupně překladači posílat každou větu jako samostatnou žádost a po přijetí odpovědi poslat další větu. Celkový počet dvojic vět ve všech zvolených datasetech však činí téměř 1 milion a počet nutných žádostí by v tomto případě byl téměř 2 miliony. Jelikož komunikace s překladači trvá nějaký čas, rozhodl jsem se hledat jinou vhodnou možnost s nižším počtem nutných žádostí pro snížení času potřebného k překladu všech datasetů.

Proto jsem dále uvažoval možnost 2, tedy rozdělení datasetu na menší části tak, aby velikost jedné části byla co největší s ohledem na maximální povolenou velikost jedné žádosti daného překladače. Počet nutných žádostí by v tomto případě byl nejmenší možný.

Nastává zde však problém. Jelikož věty tvoří souvislý text, je možné, že překladač bude překládat jednotlivé věty v závislosti na ostatních větách v dané části. Překlad dané věty může být závislý na okolních větách a tedy i na její pozici v datasetu. Dvojice vět v datasetu jsou uvažovány jako samostatné a nezávislé na kontextu okolních, proto tento jev není žádoucí a mohl by zhoršit kvalitu datasetu.

Zbývá kompromis ve formě možnosti 3, tedy posílat každou dvojici vět jako samostatnou žádost. Počet žádostí se tím sníží na polovinu oproti možnosti 1. Překlad jednotlivých vět může být závislý na druhé větě ve dvojici, což však není problém, jelikož nezávislost překladu na pozici dvojic v datasetu je dodržena.

Rozhodl jsem se tedy použít možnost 3. Byla však nutné zvolit vhodný symbol oddělující obě věty ve dvojici, aby bylo možné přeložený text znovu rozdělit na dvě věty. Po vyzkoušení různých symbolů při přeložení několika dvojic jsem se rozhodl věty oddělit symbolem nové řádky. Oba zvolené překladače tento symbol zachovávaly a neměnily jeho pozici v textu.

Postupně jsem překládal dvojice vět a ukládal je do seznamu. Když počet dvojic v seznamu dosáhl 1000, uložil jsem jej do souboru. Kdybych pro uložení čekal na přeložení celého datasetu, jakýkoliv případný pád programu by znamenal nutnost opakovat celý překlad znovu. Ukládání přeloženého datasetu po částech mi v tomto případě dovolilo opakovat překlad pouze od určité části.

U přeložených dvojic jsem štítky určující třídu převzal z jejich nepřeložených variant v původním datasetu.

6.4 Analýza překladu

Po sestavení postupu překladu a vytvoření skriptu jsem přeložil datasety SNLI, MultiNLI a SICK pomocí LINDAT Translation a Google Translate.

Při překládání jsem automaticky kontroloval, jestli byl zachován symbol nové řádky mezi větami ve dvojicích. Zachován byl ve všech případech, a proto se jednalo o vhodný oddělovací symbol.

Dále jsem provedl jednoduchou analýzu přeložených datasetů.

6.4.1 Nové duplicity

Jedním ze zjištěných údajů byl počet duplicitních dvojic. Překlad byl proveden na předzpracovaných datasetech bez duplicit, proto by jejich počet v přeložených datasetech by měl být také nulový. Ve všech třech datasetech se však objevil nenulový počet duplicitních dvojic:

	SNLI	MultiNLI	SICK
LINDAT	995	242	269
Google	677	79	199

Tabulka 3: Počet nových duplicit v přeložených datasetech

Kde se tyto duplicity vzaly? Vypsal jsem si několik nově vzniklých duplicit i s jejich anglickou variantou pro porovnání. Jednalo se vždy o velmi podobné anglické věty, které byly přeloženy do stejné podoby.

originál	Two adults walk across **a** street.	Two adults walk across **the** street.
překlad	Dva dospělí jdou přes ulici.	Dva dospělí jdou přes ulici.

Zde je příklad dvou takových vět. Tyto věty se liší pouze v použitém členu „a“ nebo „the“, proto jsou jejich české překlady stejné. Druhé věty v těchto dvojicích jsou shodné i v originále, byly proto shodné i jejich překlady. Nově vzniklé duplicity jsou tedy dvojice, jež nejsou zcela shodné v originále, ale mají shodný překlad.

6.4.2 Příklady překladu

Poté jsem ručně procházel náhodné části přeložených datasetů a hledal zajímavé rozdíly v překladech. Velká část vět byla přeložena oběma překladači stejně nebo velmi podobně.

originál	A girl in a flower dress is running on sand.
LINDAT	Dívka v květinových šatech běží po písku.
Google	Dívka v květinových šatech běží po písku.

Například tuto krátkou větu o dívce v květinových šatech přeložily oba překladače stejně. Zajímavější jsou však případy, kdy se překlady liší.

originál	A white dog with long **hair** jumps to catch a **red and green** toy.
LINDAT	Bílý pes s dlouhými **chlupy** skáče, aby chytil **červenozelenou** hračku.
Google	Bílý pes s dlouhými **vlasy** skáče, aby chytil **červenou a zelenou** hračku.

V této větě o bílém psu je obsaženo slovo „hair“, které může znamenat buď „chlupy“ nebo „vlasy“. Překladač LINDAT zvolil první možnost a Google překladač druhou. Lze předpokládat, že v případě psa se bude nejspíš jednat o „chlupy“ a LINDAT je proto v tomto případě přesnější.

Zajímavá je i druhá část věty, kde se Google rozhodl zachovat spojení „červenou a zelenou“, ale LINDAT tato slova spojil do výrazu „červenozelenou“. Jedná se o často se vyskytující jev, kdy Google překládá některé fráze více doslovně, ale LINDAT si je upravuje do jiných frází stejného významu.

originál	Four **young men** sit on the floor close to a television that is showing Elmo from **Sesame Street**.
LINDAT	Čtyři **mladíci** sedí na podlaze blízko televize, na které běží Elmo ze **Sezamové ulice**.
Google	Čtyři **mladí muži** sedí na podlaze blízko televize, která ukazuje Elma ze **Sesame Street**.

Zde můžeme znovu vidět, že Google přeložil „young men“ doslovně jako „mladí muži“ a LINDAT se rozhodl pro výraz „mladíci“. Zajímavější je však překlad názvu pořadu „Sesame Street“. Google jej zachoval nepřeložený, ale LINDAT použil českou variantu. Tento jev, kdy Google na rozdíl od LINDATu ponechává anglické názvy, se v překladech také vyskytoval často.

originál	This **mother and her daughter and granddaughter** are having car trouble, and the poor little girl looks **hot** out in the heat.
LINDAT	Tahle **matka a její dcera a vnučka** mají problémy s autem a chudinka holčička vypadá v tom vedru **rozpálená**.
Google	Tato **matka s dcerou a vnučkou** mají problémy s autem a nebohá holčička vypadá ve vedru **horko**.

Další příklad je zajímavý, protože konec Google překladu „vypadá ve vedru horko“ nedává příliš smysl. Oproti tomu překlad LINDATu je mnohem smysluplnější. Je zde vidět, že příliš doslovný překlad může někdy vést ke ztrátě smyslu věty.

V první části věty však tentokrát doslovnější překlad zvolil LINDAT a Google tuto frázi upravil na „matka s dcerou a vnučkou“. Neplatí tedy, že by Google překladač byl vždy více doslovný než LINDAT.

originál	At a **convention**, a **man takes a phone call** behind the booth where he is selling things while others examine his merchandise.
LINDAT	Na **kongresu si muž telefonuje** za stánkem, kde prodává věci, zatímco ostatní zkoumají jeho zboží.
Google	Na **sjezdu zavolá muž** za stánkem, kde prodává věci, zatímco ostatní zkoumají jeho zboží.

Zde se jedná o další příklad, kdy méně doslovný překlad vedl k více smysluplné větě. Jedná se znovu o častý případ, kdy Google zvolil více doslovný překlad. Zajímavý je zde i rozdílný překlad slova „convention“.

originál	A girl in a striped black and white sweater holds on to **her friend** who is bent over wearing a hooded black sweater and torn blue jeans.
LINDAT	Dívka v pruhovaném černobílém svetru se drží **své kamarádky**, která je ohnutá a má na sobě černý svetr s kapucí a roztrhané modré džíny.
Google	Dívka v pruhovaném černobílém svetru se drží **svého přítele**, který je ohnutý, má na sobě černý svetr s kapucí a roztrhané modré džíny.

Tento příklad obsahuje anglické slovo „friend“, u kterého není možné určit, zda se jedná o muže nebo ženu. Překladače tedy měly úplnou volnost, zda použít ženskou nebo mužskou variantu. LINDAT se rozhodl pro kamarádku a Google pro přítele. Nejenže se jedná o různé pohlaví, ale české slovo „přítel“ může popisovat jiný typ vztahu než „kamarádka“. Význam obou překladů je proto odlišný.

originál	**A well safety equipment biker** racing on the trail in a wooded area with number 569 on his **bike**.
LINDAT	**Motorkář s bezpečnostním vybavením studny** závodí na stezce v zalesněném areálu s číslem 569 na **kole**.
Google	**Dobře bezpečnostní vybavení biker** závodící na stezce v zalesněné oblasti s číslem 569 na **kole**.

Méně doslovný překlad nemusí být vždy výhodou. V tomto případě se jedná o chybně formulovanou anglickou větu. Místo slova „equipment“ („vybavení“) zde mělo pravděpodobně být „equipped“ („vybavený“). Google překladač tuto chybně formulovanou část přeložil slovo od slova a chyba se tedy přenesla do češtiny bez negativního dopadu na zbytek překladu. LINDAT se však pokusil tuto část přeformulovat a příslovce „well“ („dobře“) přeložil jako podstatné jméno znamenající „studna“, čímž změnil smysl celé věty.

V tomto případě je zajímavé podívat se i na druhou větu z této dvojice:

originál	A **biker** goes on to the last lap of the trail.
LINDAT	**Motorkář** pokračuje do posledního kola stezky.
Google	**Cyklista** pokračuje do posledního kola stezky.

Je vidět, že LINDAT v obou větách přeložil „biker“ jako „motorkář“. Google byl méně konzistentní a rozhodl se v první větě ponechat anglické slovo a ve druhé jej přeložit jako „cyklista“. Ani LINDAT zde však není plně konzistentní. V první větě mluví o motorkáři, ale na jejím konci zmiňuje kolo, které by se více hodilo k cyklistovi.

6.4.3 Shrnutí

Ze všech zmíněných příkladů je viditelné, že oba překladače mají své nedokonalosti a produkované překlady se často liší. Právě tato odlišnost v překladech by se mohla výrazně projevit při trénování modelů pomocí přeložených datasetů.

Budou se výsledky modelů natrénovaných na jednom překladu výrazně lišit od výsledků druhého? Povedou doslovnější překlady Google překladače k horším výsledkům v NLI úloze, nebo je naopak zlepší? Bude nekonzistence překladu vět ve stejné dvojici velkým problémem? Zanáší strojový překlad do datasetu příliš velké chyby pro jeho praktické použití?

Cílem další části této práce bude nalézt odpovědi na všechny tyto otázky.

7 Trénování a analýza výsledků

Následující kapitola se zabývá trénováním anglických modelů na zvolených datasetech, trénováním českých modelů na přeložených datasetech a jejich testováním a porovnáním.

7.1 Parametry trénování

Při trénování všech modelů jsem používal následující parametry:

- **míra učení**: 10^{-5}
- **optimalizátor**: ADAM ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$)
- **ztrátová funkce**: sparse categorical crossentropy
- **počet epoch**: 2 (5 pro SICK dataset)

Zvolil jsem optimalizátor ADAM, jelikož se jedná o velmi často používaný optimalizační algoritmus. Jeho parametry jsem ponechal na výchozích hodnotách nastavených platformou TensorFlow. Zvolenou ztrátovou funkcí je varianta křížové entropie pro úlohy klasifikace do více tříd.

Počet epoch jsem volil nízký pro zamezení zkreslení výsledků přetrénováním. Ze stejného důvodu jsem zvolil nízkou míru učení. Pro trénování na SICK datasetu bylo třeba použít více epoch kvůli jeho řádově menší velikosti.

Lze předpokládat, že pokročilejším laděním parametrů by bylo možné dosáhnout lepších výsledků. Předmětem této práce však není dosáhnout co nejlepších výsledků, ale porovnat výsledky jednotlivých modelů mezi sebou. Optimální volbu parametrů jsem proto nepovažoval za důležitou, ale pouze volbu stejných parametrů pro všechna trénování.

Jako míru přesnosti modelu na dané testovací množině jsem použil poměr počtu správně klasifikovaných dvojic vět H vůči velikosti množiny N:

$$\text{acc} = \frac{H}{N} \cdot 100\%$$

7.2 Trénování anglických modelů

Nejprve jsem trénoval zvolené předtrénované modely pro angličtinu. Pro každý předtrénovaný model jsem provedl 4 různá trénování, každé na trénovací množině jednoho ze zvolených datasetů. Získal jsem tedy 4 různé natrénované modely pro každý předtrénovaný.

Natrénované modely jsem postupně otestoval na testovacích množinách všech 4 datasetů a spočetl přesnost modelů na dané množině.

7.2.1 Výsledky BERT modelu

První trénování a následné testování proběhlo na BERT modelu. Zde jsou uvedeny výsledné přesnosti 4 natrénovaných modelů na jednotlivých testovacích množinách:

	test acc			
trén. množina	SNLI	MultiNLI	SICK	SNLI+MultiNLI
SNLI	90.6%	75.1%	57.6%	82.8%
MultiNLI	80.4%	84.1%	54.2%	82.3%
SICK	45.1%	50.6%	86.9%	47.8%
SNLI+MultiNLI	90.6%	85.3%	57.8%	87.9%

Tabulka 4: Přesnosti BERT modelů na testovacích množinách

Z tabulky lze vidět, že modely trénované na SNLI, MultiNLI a SICK dosáhly největší přesnosti na testovací množině toho samého datasetu. Jelikož je očekávatelné, že testovací množina stejného datasetu bude podobnější té trénovací než množiny jiných datasetů, není tento výsledek překvapující.

Model trénován na SNLI dosáhl při testování na MultiNLI přibližně o 15% horší přesnosti než na své testovací množině. Oproti tomu u modelu trénovaného na MultiNLI je tento rozdíl pouze okolo 4%. Toto ukazuje, že MultiNLI obsahuje obecnější znalosti než SNLI, což odpovídá tvrzení jeho autorů. [7]

Model trénován na spojeném SNLI+MultiNLI datasetu dosahuje na testovacích množinách SNLI a MultiNLI datasetů stejné nebo lepší přesnosti než samostatné modely. Toto ukazuje, že znalosti získané z SNLI datasetu lze uplatnit v MultiNLI a naopak. Jejich spojení se tedy vyplatí.

Model trénován na SICK datasetu dosahuje výrazně horších výsledků na ostatních testovacích množinách. Zároveň však ostatní modely dosahují podobně horších výsledků na testovací množině SICK datasetu. Znalosti nejsou tedy dobře přenositelné mezi SNLI/MultiNLI a SICK. Spojení všech třech datasetů by proto nemělo význam.

7.2.2 Příklady klasifikace BERT modelem

Zde je uvedeno několik příkladů správné a chybné klasifikace. Jedná se o příklady modelu trénovaného i testovaného na příslušných množinách SNLI+MultiNLI datasetu.

předpoklad	This church choir sings to the masses as they sing joyous songs from the book at a church.
hypotéza	A choir singing at a baseball game.
třída (anotace)	contradiction
třída (model)	contradiction

V tomto případě model správně určil, že pěvecký sbor nemůže zpívat zároveň v kostele i na baseballovém utkání a věty se tedy vzájemně vyvrací.

předpoklad	A woman with a green headscarf, blue shirt and a very big grin.
hypotéza	The woman is very happy.
třída (anotace)	entailment
třída (model)	entailment

Další příklad obsahuje předpoklad o ženě s širokým úsměvem. Hypotéza tvrdí, že žena je šťastná. Model správně určil, že předpoklad zde implikuje hypotézu.

předpoklad	A man, woman, and child get their picture taken in front of the mountains.
hypotéza	A family on vacation is posing.
třída (anotace)	entailment
třída (model)	neutral

Zde je předpokladem muž, žena a dítě foceni před horami. Model chybně určil, že hypotéza o rodině pózující na dovolené nevychází z tohoto předpokladu a klasifikoval vztah této dvojice jako neutrální.

předpoklad	A man is sitting on the floor, sleeping.
hypotéza	A man is lying down, sleeping.
třída (anotace)	contradiction
třída (model)	entailment

V posledním příkladě model nedokázal určit, že muž nemůže zároveň sedět i ležet. Dle anotace se jedná o protiřečení, ale model tuto dvojici klasifikoval jako implikaci.

7.2.3 Výsledky modelů BERT-Large a RoBERTa

Dále jsem stejným způsobem natrénoval modely BERT-Large a RoBERTa. Rozdíly mezi výsledky na jednotlivých testovacích množinách byly obdobné jako u BERT modelu, proto není třeba je zde uvádět všechny.

Jako míru přesnosti modelu jsem zvolil pouze přesnost dosaženou na testovací množině SNLI+MultiNLI datasetu modelem trénovaným na trénovací množině SNLI+MultiNLI datasetu. Jedná se o největší a nejobecnější trénovací i testovací množinu, proto je tato přesnost nejvíce vypovídající.

Zde jsou uvedeny přesnosti zvolených modelů:

model	**acc**
BERT	87.9%
BERT-Large	88.5%
RoBERTa	89.1%

Tabulka 5: Přesnosti modelů na testovací množině SNLI+MultiNLI

Při použití BERT-Large modelu došlo ke zlepšení přesnosti oproti BERT modelu. Zlepšení bylo dosaženo zřejmě díky většímu počtu parametrů modelu. Model RoBERTa dosahuje lepší přesnosti oproti nejen podobně velkému modelu BERT, ale i výrazně většímu modelu BERT-Large.

Tyto výsledky budou dále sloužit jako reference při testování českých modelů.

7.3 Trénování českých modelů

Trénoval jsem pouze jediný český předtrénovaný model, FERNET-C5. Protože mám dvě sady přeložených datasetů, jednu překladačem LINDAT a druhou Google překladačem, trénoval jsem model zvlášť na každé z nich.

Přesnosti i zde budou uvedeny pouze pro modely trénované a testované na příslušných množinách SNLI+MultiNLI datasetu, tentokrát však na přeložené verzi. Modely jsem trénoval třikrát. Jeden model na překladu LINDAT, druhý na Google překladu a třetí na datasetu LINDAT+Google, jenž vznikl spojením obou překladů. Tento spojený překlad byl tedy 2x větší a pro překlady SNLI+MultiNLI datasetu tvořen téměř 4 miliony dvojic vět.

7.3.1 Analýza výsledků

Každý z modelů jsem testoval na třech verzích překladu testovací množiny, tedy LINDAT, Google a LINDAT+Google. Přesnost na spojené testovací množině LINDAT+Google je pouze průměrem přesností na oddělených množinách.

	test acc		
trén. množina	LINDAT	Google	LINDAT+Google
LINDAT	86.2%	85.3%	85.8%
Google	86.2%	86.0%	86.1%
LINDAT+Google	86.3%	86.0%	86.1%

Tabulka 6: Přesnosti FERNET-C5 modelů na testovacích množinách

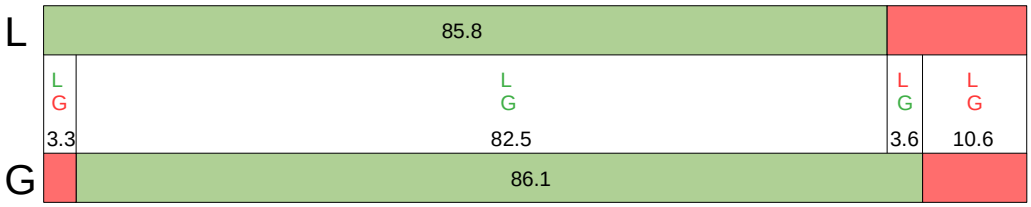
Z tabulky lze vidět, že všechny přesnosti jsou velice podobné a pohybují se kolem hodnoty 86%. Jedná se o horší výsledek než jakého dosáhl každý z anglických modelů. Rozdíl oproti BERT modelu jsou však pouze necelá 2%.

Co se konkrétních přesností týče, model trénován na LINDAT překladu dosáhl pouze nepatrně nižší přesnosti na Google překladu než na vlastním překladu. Model trénován na Google překladu dokonce dosáhl nepatrně vyšší přesnosti na LINDAT překladu než na vlastním.

Jelikož modely natrénované na jednom překladu dosáhly na tomto překladu velice podobné přesnosti jako na druhém, můžeme usoudit, že znalosti získané z jednoho překladu jsou dobře přenositelné i na druhý. Tato skutečnost je zajímavá z důvodu dříve zmíněné odlišnosti mezi překlady.

Podobný závěr nám ukazují i výsledky modelu trénovaného na spojeném LINDAT+Google datasetu. Trénováním na obou překladech jsme nedosáhli žádného výrazného zlepšení oproti trénování pouze na jednom.

Dále jsem provedl analýzu rozdílů mezi klasifikacemi modelu trénovaného na LINDAT překladu (model L) a modelu trénovaného na Google překladu (model G). Modely jsem testoval na spojené testovací množině LINDAT+Google a analyzoval rozdíly ve správnosti jejich výstupů na stejný vstup.



Obrázek 5: Porovnání správnosti klasifikací modelem trénovaným na LINDAT překladu (L) a modelem trénovaným na Google překladu (G). Hodnoty jsou uvedeny v procentech. (zelená - správně, červená - chybně)

Z obrázku lze vidět, že 82.5% množiny klasifikovaly oba modely správně a 10.6% oba chybně. Úsek, kde model L klasifikoval správně a model G špatně, tvoří pouze 3.3% množiny. Podobně malý (3.6%) je i úsek, kde model G klasifikoval správě a model L špatně. Malé úseky odlišné správnosti klasifikace mezi modely znovu ukazují, že oba modely při trénování získaly velmi podobné znalosti.

Otázkou však je, z jakého překladu dvojice vět v těchto úsecích odlišné klasifikace pocházejí. Analýza byla provedena na spojené LINDAT+Google množině, a je proto možné, že tyto úseky obsahují pouze věty toho překladu, jehož příslušný model je klasifikoval správně. Po bližší kontrole jsem však zjistil, že nejen tyto úseky, ale každý ze 4 úseků je tvořen přibližně z 50% LINDAT překladem a ze zbylých 50% Google překladem.

Závěrem této analýzy je skutečnost, že výsledky modelu trénovaného na množině přeložené LINDAT překladačem jsou srovnatelné s výsledky modelu trénovaného na množině přeložené Google překladačem, i přes dříve zmíněné a na příkladech ukázané rozdíly v překladech. Tyto rozdíly se nezdají mít žádný negativní vliv na schopnost modelu dosáhnout srovnatelných výsledků při testování na překladu druhého překladače.

7.3.2 Příklady klasifikace

Zde je uvedeno několik příkladů klasifikace dvojic vět modelem trénovaným na LINDAT překladu (model L) a modelem trénovaným na Google překladu (model G).

předpoklad	Mladá rodina si užívá pocit oceánských vln, které se jí vlní u nohou.
hypotéza	Rodina je na pláži.
třída (anotace)	entailment
třída (model L)	entailment
třída (model G)	entailment

V prvním příkladě oba modely porozuměly, že přítomnost oceánských vln implikuje pláž, a klasifikovaly dvojici vět správně.

předpoklad	Snowboardista na široké sněhové pláni.
hypotéza	Snowboardista klouzající po sněhovém poli.
třída (anotace)	neutral
třída (model L)	entailment
třída (model G)	contradiction

Zde se jedná o případ neutrálního vztahu. Snowboardista by se mohl klouzat, ale předpoklad nic takového neimplikuje. Oba modely však tuto dvojici klasifikovaly chybně, každý však jinak. Je možné, že modely zmátlo použití rozdílných slov pro stejné místo („pláň“ a „pole“).

předpoklad	Osoba s fialovou košilí maluje obraz ženy na bílou zeď.
hypotéza	Žena maluje portrét opice.
třída (anotace)	contradiction
třída (model L)	contradiction
třída (model G)	neutral

V dalším případě model L správně určil, že věty se navzájem vyvrací. Model G však chybně klasifikoval vztah této dvojice jako neutrální.

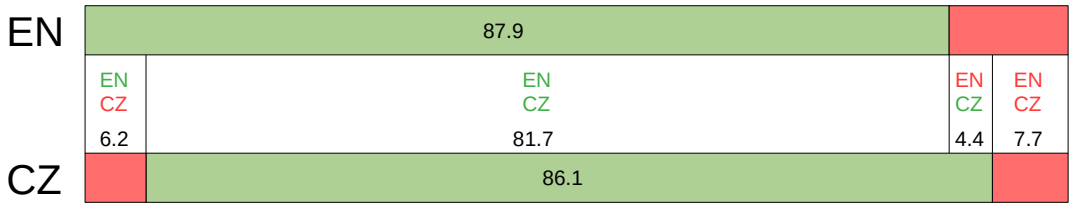
předpoklad	Před katedrálou Notre Dame přistává v noci stíhací letadlo.
hypotéza	V stíhacím letadle nikdo není.
třída (anotace)	neutral
třída (model L)	contradiction
třída (model G)	neutral

Posledním příkladem je další neutrální vztah, tentokrát však správně klasifikovaný modelem G. Chybně zde třídu určil model L.

7.4 Porovnání anglických a českých modelů

Mimo porovnání modelů trénovaných na jednotlivých překladech mezi sebou je také vhodné porovnat je s anglickými modely. Pro porovnání jsem zvolil BERT model, jelikož používá stejnou architekturu jako FERNET-C5.

Český model (CZ) byl trénovaný a testovaný na příslušných množinách LINDAT+Google překladu. Anglický model (EN) byl testován na množině obsahující odpovídající originální dvojice. Analyzoval jsem rozdíly ve správnosti jejich výstupů na odpovídající vstupy.



Obrázek 6: Porovnání správnosti klasifikací originálu BERT modelem (EN) a překladu FERNET-C5 modelem (CZ). Hodnoty jsou uvedeny v procentech. (zelená - správně, červená - chybně)

Z obrázku lze vidět, že 81.7% množiny klasifikovaly oba modely správně a 7.7% oba chybně. Úsek, kde anglický model klasifikoval správně a český model špatně, tvoří 6.2% množiny. Zbývajících 4.4% tvoří úsek, kde český model klasifikoval správně a anglický model špatně.

Úseky, kde pouze jeden model správně klasifikoval, jsou zde větší než při porovnání dvou českých modelů na obrázku 5. Rozdíly v klasifikaci mezi anglickým a českým modelem jsou tedy vyšší než mezi dvěma českými modely. Úseky rozdílné klasifikace jsou však i zde malé v porovnání s úseky stejné klasifikace. Můžeme proto usoudit, že znalosti získané anglickým modelem pro angličtinu téměř odpovídají znalostem získaným českým modelem pro český překlad.

Téměř dvouprocentní rozdíl mezi přesnostmi anglického a českého modelu může mít mnoho důvodů zahrnujících odlišnosti předtrénovaných modelů a případné chyby v překladech. Jak bylo ukázáno v tabulce 5, různé anglické předtrénované modely dosahují podobných rozdílů v přesnostech. Je tedy možné, že případný kvalitnější český model by dosáhl více srovnatelné přesnosti.

Dále je zde uvedeno několik příkladů klasifikace dvojic vět z kapitoly 6.4 anglickým modelem BERT a klasifikace jejich dvou překladů modelem FERNET-C5 trénovaným na LINDAT+Google množině.

originál	předpoklad hypotéza třída	A girl in a flower dress is running on sand. The girl is wearing a green striped dress. neutral
LINDAT	předpoklad hypotéza třída	Dívka v květinových šatech běží po písku. Dívka má na sobě zelené pruhované šaty. contradiction
Google	předpoklad hypotéza třída	Dívka v květinových šatech běží po písku. Dívka má na sobě zelené pruhované šaty. contradiction
anotace	třída	contradiction

Překlady dvojice vět o dívce v šatech byly shodné, a proto byly i stejně klasifikovány. Český model správně určil, že si věty protiřečí z důvodu rozdílného vzoru šatů. Anglický model zde však originální dvojici klasifikoval chybně.

originál	předpoklad	This mother and her daughter and granddaughter are having car trouble, and the poor little girl looks hot out in the heat.
	hypotéza	The car that belongs to the family is having issues.
	třída	entailment
LINDAT	předpoklad	Tahle matka a její dcera a vnučka mají problémy s autem a chudinka holčička vypadá v tom vedru rozpálená.
	hypotéza	Auto, které patří rodině, má problémy.
	třída	entailment
Google	předpoklad	Tato matka s dcerou a vnučkou mají problémy s autem a nebohá holčička vypadá ve vedru horko.
	hypotéza	Auto, které patří do rodiny, má problémy.
	třída	entailment
anotace	třída	entailment

Zde se jedná o případ, kdy byl konec předpokladu nesmyslně přeložen Google překladačem. Jak lze vidět, na správnost klasifikace to ovšem nemělo vliv. Tato část věty však nijak neovlivňuje vztah s hypotézou, což může být důvodem pro neovlivnění klasifikace.

originál	předpoklad	At a convention, a man takes a phone call behind the booth where he is selling things while others examine his merchandise.
	hypotéza	The man is walking outside.
	třída	contradiction
LINDAT	předpoklad	Na kongresu si muž telefonuje za stánkem, kde prodává věci, zatímco ostatní zkoumají jeho zboží.
	hypotéza	Muž jde ven.
	třída	contradiction
Google	předpoklad	Na sjezdu zavolá muž za stánkem, kde prodává věci, zatímco ostatní zkoumají jeho zboží.
	hypotéza	Muž jde ven.
	třída	entailment
anotace	třída	contradiction

Podobně jako v předchozím příkladu zde nedává Google překlad předpokladu smysl v první části věty a LINDAT překlad více odpovídá originálu. Jelikož je

tato část jedinou odlišností v překladu, lze usoudit, že se jedná o důvod chybné klasifikace Google překladu. Originál i LINDAT překlad byly klasifikovány správně. Lze tedy vidět, že případné nedostatky v překladu mohou mít za následek chybné určení třídy.

originál	předpoklad	A girl in a striped black and white sweater holds on to her friend who is bent over wearing a hooded black sweater and torn blue jeans.
	hypotéza	The girl is touching her friend.
	třída	entailment
LINDAT	předpoklad	Dívka v pruhovaném černobílém svetru se drží své kamarádky, která je ohnutá a má na sobě černý svetr s kapucí a roztrhané modré džíny.
	hypotéza	Dívka se dotýká své kamarádky.
	třída	entailment
Google	předpoklad	Dívka v pruhovaném černobílém svetru se drží svého přítele, který je ohnutý, má na sobě černý svetr s kapucí a roztrhané modré džíny.
	hypotéza	Dívka se dotýká svého přítele.
	třída	entailment
anotace	třída	entailment

Zde se jedná o případ, kdy překladače rozdílně přeložily slovo „friend“. V hypotézách však bylo přeloženo stejně jako v jednotlivých předpokladech. Vidíme, že oba překlady byly správně klasifikovány. Zajímavé však je, že jsou obě dvojice správně klasifikovány i při prohození překladu hypotéz a způsobení nekonzistence. Ukazuje se tedy, že nekonzistence mezi větami v překladu nemusí ovlivnit správnost klasifikace.

originál	předpoklad	A well safety equipment biker racing on the trail in a wooded area with number 569 on his bike.
	hypotéza	A biker goes on to the last lap of the trail.
	třída	neutral
LINDAT	předpoklad	Motorkář s bezpečnostním vybavením studny závodí na stezce v zalesněném areálu s číslem 569 na kole.
	hypotéza	Motorkář pokračuje do posledního kola stezky.
	třída	neutral
Google	předpoklad	Dobře bezpečnostní vybavení biker závodící na stezce v zalesněné oblasti s číslem 569 na kole.
	hypotéza	Cyklista pokračuje do posledního kola stezky.
	třída	neutral
anotace	třída	neutral

Nekonzistence a chyby v překladu neovlivnily správnost klasifikace ani v posledním příkladě, který obsahuje nejen rozdílný překlad slova „biker“, ale i nesmyslný začátek věty, tentokrát v LINDAT překladu. I přes tyto rozdíly byly obě české dvojice správně klasifikovány.

7.5 Kombinace modelů

V předchozích podkapitolách bylo uvedeno několik modelů natrénovaných na různých množinách a dosahujících různých výsledků. Nabízí se otázka, k jakým výsledkům by vedlo spojení těchto modelů. Jakým způsobem je ale spojit?

Jak bylo již řečeno, výstupem zvolených modelů je vektor obsahující hodnotu pravděpodobnosti pro každou třídu. Vybranou třídou je poté třída odpovídající maximální hodnotě v tomto vektoru. Pokud vektor označíme jako l, bude třída c určena jako:

$$c = \arg\max(l)$$

Nyní uvažujme dva modely. První je český model trénovaný na LINDAT překladu s výstupním vektorem l. Druhý je model trénovaný na Google překladu s výstupem g. Jednoduchý způsob spojení těchto vektorů je aritmetický průměr.

Při kombinaci těchto modelů přivedeme vstupní věty na vstupy obou modelů a výslednou třídu určíme jako:

$$c = \arg\max\left(\frac{l+g}{2}\right)$$

Jedná se o klasifikaci průměrováním výstupu dvou různých modelů. Další možností kombinace je přidání anglického modelu BERT, jehož vstupem budou odpovídající anglické věty a výstupem vektor b. Jelikož je anglický model pouze jeden, přiřadíme mu dvojnásobnou váhu.

$$c = \arg\max\left(\frac{2\cdot b+l+g}{4}\right)$$

Kombinované modely jsem otestoval na testovací množině LINDAT+Google překladu a porovnal s ostatními modely.

model	**trén. množina**	**acc**
BERT	originál	87.9%
FERNET-C5	LINDAT	85.8%
	Google	86.1%
	LINDAT+Google	86.1%
2x FERNET-C5	LINDAT, Google	86.5%
BERT + 2x FERNET-C5	originál, LINDAT, Google	88.8%

Tabulka 7: Porovnání přesností testovaných modelů

Vidíme, že průměrováním dvou FERNET-C5 modelů bylo dosaženo větší přesnosti než použitím pouze jednoho. Je však otázkou, zda takto malý přírůstek přesnosti stojí za zdvojnásobení výpočetních nároků.

Kombinace anglického modelu s českými dosáhla vyšší přesnosti než nejen samotné české modely, ale i anglický BERT model. Pro jeho použití je však nutná anglická i česká verze vstupu. Tuto kombinaci jsem testoval především ze zvědavosti a nevidím v ní možnosti praktického využití.

8 Závěr

V této práci byla nejprve popsána problematika zpracování přirozeného jazyka (NLP) a představena úloha inference přirozeného jazyka (NLI). Dále bylo uvedeno několik anglických datasetů pro tuto úlohu a poukázáno na problém absence českých datasetů. Jako řešení bylo navrženo přeložení anglických datasetů systémy strojového překladu.

V dalších částech byly popsány principy modelů strojového učení pro NLP, především *transformer* architektura, a uvedeny existující anglické a české modely vhodné pro NLI. Byl zde uveden i postup trénování těchto modelů s využitím přeneseného učení.

Další kapitolou byl úvod do problematiky strojového překladu textu a implementace překladu datasetů pomocí dvou online služeb strojového překladu. Přeložené datasety byly analyzovány a bylo poukázáno na rozdíly a nedostatky.

Poslední částí bylo trénování českých a anglických modelů s analýzou a porovnáním jejich výsledků. Výsledky českých modelů se ukázaly být srovnatelné mezi sebou a pouze o nejvýše jednotky procent horší než anglické modely.

V kapitole 3 byla položena následující otázka:

Bylo by možné využít systémů strojového překladu pro generování ekvivalentních českých datasetů z anglických?

Odpověď bohužel není jednoznačná. Z dosažených výsledků můžeme vyvodit závěr, že českým překladem anglického datasetu je možné dotrénovat český předtrénovaný model a v NLI úloze dosáhnout na přeložené verzi testovací množiny přesnosti blížící se přesnostem anglických modelů na originální testovací množině, a to i pokud je dataset přeložen jiným překladačem než trénovací množina.

Rozdíly v překladech získaných jednotlivými překladači se nezdály mít významný vliv na kvalitu natrénovaných modelů. Doslovnost překladu či jeho nekonzistence a chyby ve většině případů neovlivnily výsledek klasifikace.

Pro zhodnocení možnosti využití takto natrénovaných českých modelu pro originální český text by bylo nutné provést testování na originálním českém datasetu. Nepodařilo se mi však nalézt takový dataset pro NLI úlohu a tato otázka tedy zůstává nezodpovězena. Další nezodpovězenou otázkou je možnost využití v ostatních NLP úlohách, například v úloze generování přirozeného textu.

I přes nejednoznačnou odpověď můžou být závěry této práce užitečné pro inspiraci budoucích výzkumů a vést k podrobnějším analýzám využitelnosti přeložených datasetů pro trénování českých NLP modelů. Tyto analýzy by mohly být dále užitečné pro zjednodušení vývoje českých NLP aplikací.

Seznam použitých zdrojů

1. BROWNLEE, Jason. 4 Types of Classification Tasks in Machine Learning. *Machine Learning Mastery*. [online]. Dostupné z: https://machinelearningmastery.com/types-of-classification-in-machine-learning/.
2. WEINSHALL, Daphna a MANDELBAUM, Amit. Distance-based Confidence Score for Neural Network Classifiers. [online]. 2017. Dostupné z: https://www.arxiv-vanity.com/papers/1709.09844/.
3. CHOWDHURY, G. Natural language processing. *Annual Review of Information Science and Technology*. [online]. 2003. Dostupné z: https://strathprints.strath.ac.uk/2611/1/strathprints002611.pdf.
4. A Complete Guide to Natural Language Processing. *DeepLearning.AI*. [online]. Dostupné z: https://www.deeplearning.ai/resources/natural-language-processing/.
5. Natural language inference. *NLP-progress*. [online]. Dostupné z: http://nlpprogress.com/english/natural_language_inference.html.
6. BOWMAN, Samuel R.; ANGELI, Gabor; POTTS, Christopher a MANNING, Christopher D. A large annotated corpus for learning natural language inference. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [online]. 2015. Dostupné z: http://nlp.stanford.edu/pubs/snli_paper.pdf.
7. WILLIAMS, Adina; NANGIA, Nikita a BOWMAN, Samuel. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. [online]. 2018. Dostupné z: http://aclweb.org/anthology/N18-1101.
8. BENTIVOGLI, L.; BERNARDI, R.; MARELLI, M.; MENINI, S.; BARONI M. et al. SICK Through the SemEval Glasses. *Lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment*. [online]. 2016. Dostupné z: https://marcobaroni.org/publications/bentivogli-etal-lrej-2016.pdf.
9. ULLRICH, Herbert. Dataset for Automated Fact Checking in Czech Language. [online]. 2021. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/95430/F3-DP-2021-Ullrich-Herbert-Thesis___Ullrich.pdf.
10. HORAN, Cathal. Tokenizers: How machines read. *FloydHub Blog*. [online]. Dostupné z: https://blog.floydhub.com/tokenization-nlp/.
11. IRCING, Pavel. Přednáška: Vektorová reprezentace slov pomocí neuronových sítí. *Strojové zpracování přirozeného jazyka*. 2021.

12. ALAMMAR, Jay. The Illustrated Word2vec. [online]. Dostupné z: https://jalammar.github.io/illustrated-word2vec/.

13. BENGIO, Yoshua; DUCHARME, Réjean; VINCENT, Pascal. A Neural Probabilistic Language Model. *Proceedings of NIPS*. [online]. 2001. Dostupné z: https://papers.nips.cc/paper_files/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf.

14. RUDER, Sebastian. A Review of the Neural History of Natural Language Processing. *ruder.io*. [online]. Dostupné z: https://www.ruder.io/a-review-of-the-recent-history-of-nlp/.

15. PRIYA, Bala. Softmax Activation Function: Everything You Need to Know. [online]. Dostupné z: https://www.pinecone.io/learn/softmax-activation/.

16. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion et al. Attention Is All You Need. [online]. 2017. Dostupné z: https://arxiv.org/pdf/1706.03762.pdf.

17. ALAMMAR, Jay. The Illustrated Transformer. [online]. Dostupné z: http://jalammar.github.io/illustrated-transformer/.

18. BROWNLEE, Jason. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. *Machine Learning Mastery*. [online]. Dostupné z: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

19. BROWNLEE, Jason. A Gentle Introduction to Transfer Learning for Deep Learning. *Machine Learning Mastery*. [online]. Dostupné z: https://machinelearningmastery.com/transfer-learning-for-deep-learning/.

20. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [online]. 2019. Dostupné z: https://arxiv.org/pdf/1810.04805v2.pdf.

21. LIU, Yinhan; OTT, Myle; GOYAL, Naman; DU, Jingfei; JOSHI, Mandar et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach. [online]. 2019. Dostupné z: https://arxiv.org/pdf/1907.11692.pdf.

22. LEHEČKA, Jan; ŠVEC, Jan. Comparison of Czech Transformers on Text Classification Tasks. [online]. 2021. Dostupné z: https://arxiv.org/pdf/2107.10042.pdf.

23. O MetaCentru. *MetaCentrum (MetaVO) - virtuální organizace pro celou akademickou obec*. [online]. Dostupné z: https://metavo.metacentrum.cz/cs/about/index.html.

24. Why TensorFlow. *TensorFlow*. [online]. Dostupné z: https://www.tensorflow.org/about.

25. The AI community building the future. *Hugging Face*. [online]. Dostupné z: https://huggingface.co/.

26. OKPOR, M. D. Machine Translation Approaches: Issues and Challenges. [online]. 2014. Dostupné z: https://www.ijcsi.org/papers/IJCSI-11-5-2-159-165.pdf.

27. WU, Yonghui; SCHUSTER, Mike; CHEN, Zhifeng; LE, Quoc V.; NOROUZI, Mohammad et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. [online]. 2016. Dostupné z: https://arxiv.org/pdf/1609.08144.pdf.

28. POPEL, Martin; TOMKOVA, Marketa; TOMEK, Jakub; KAISER, Łukasz; USZKOREIT, Jakob et al. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature*. [online]. 2020. Dostupné z: https://www.nature.com/articles/s41467-020-18073-9.

29. MOLTZAU, Alex. The History of Google Translate. *Medium*. [online]. Dostupné z: https://alexmoltzau.medium.com/the-history-of-google-translate-fcbe9de3c10e.

30. deep_translator documentation. [online]. Dostupné z: https://deep-translator.readthedocs.io/en/latest/.

31. LINDAT Translation Docs. [online]. Dostupné z: https://lindat.mff.cuni.cz/services/translation/docs.