VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE A ROZPOZNÁNÍ VOLACÍCH ZNAKŮ V VHF KOMUNIKACI

CALL SIGN DETECTION AND RECOGNITION IN VHF COMMUNICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JURAJ DEDIČ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. IGOR SZŐKE, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

140505

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Dedič Juraj**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Detekce a rozpoznání volacích znaků ve VHF komunikaci**
Kategorie:	Zpracování řeči a přirozeného jazyka
Akademický rok:	2022/23

Zadání:

1. Nastudujte základy letecké komunikace - ICAO gramatiky. Soustřeďte se na volací znaky. Nastudujte základy zpracování přirozeného jazyka a dostupných nástrojů.
2. Seznamte se s dodanými daty (záznamy z ATC).
3. Navrhněte metodu, jak v automatickém přepisu identifikovat volací znaky. Implementujte ji a ověřte její úspěšnost.
4. Zdokonalte metodu a rozšiřte množství tříd, které detekujete. Změřte zlepšení vašeho přístupu.
5. Zhodnoťte výsledky a navrhněte směry dalšího vývoje.
6. Vyrobte A2 plakátek a cca 30 vteřinové video prezentující výsledky vaší práce.

Literatura:

- Zuluaga-Gomez, J. et al. Bertraffic: Bert-based joint speaker role and speaker change detection for air traffic control communications. arXiv:2110.05781 (2021).
- ATCO2 projekt, http://atco2.org
- Dále dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3 ze zadání.

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

Vedoucí práce:	**Szőke Igor, Ing., Ph.D.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	31.10.2022

Abstrakt

Práca sa zaoberá spracovaním dát z leteckej komunikácie s cieľom rozpoznania volacích znakov. Konkrétne ide o rozpoznanie volacích znakov z ľuďmi vytvorených a automatizovaných textových prepisov komunikácie medzi pilotmi a riadením letovej prevádzky.

Porovnávané sú rôzne spôsoby riešenia. Práca popisuje návrh a implementáciu systému identifikácie týchto znakov za pomoci vhodnej technológie založenej na veľkých jazykových modeloch. Jedným z výstupov je služba, ktorá dokáže v poskytnutých dátach označiť volacie znaky, čo umožní efektívnejšiu prácu s týmito údajmi.

Abstract

This work explores the processing of data from air traffic communication in order to detect and recognize the call signs it contains. Particularly it involves recognizing these call signs in human made and automated text transcripts of the communication between pilots and air traffic controllers.

The thesis compares various ways of solving this and describes their problems. It implements a system for the identification of these call signs using a suitable technology based on large language models. One of the outputs of this work is a service that is able to distinguish the call signs, which enables indexation and sorting of this data in an efficient way.

Kľúčové slová

VHF komunikácia, spracovanie prirodzeného jazyka, volacie znaky, letectvo, ATC, rozpoznanie pomenovaných entít, neurónové siete, SpaCy, BERT, GPT, transformer, jazykové modely

Keywords

VHF communication, callsigns, aviation, natural language processing, named entity recognition, neural networks, SpaCy, BERT, GPT, language models, air traffic control, ATC, aviation safety

Citácia

DEDIČ, Juraj. *Detekce a rozpoznání volacích znaků v VHF komunikaci.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Igor Szőke, Ph.D.

Detekce a rozpoznání volacích znaků v VHF komunikaci

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Igora Szőkeho, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Juraj Dedič
9. mája 2023

Poďakovanie

Ďakujem môjmu vedúcemu práce Ing. Igorovi Szőke Ph.D. za zaujímavú tému práce, ochotu a cenné rady, ktoré mi pomohli pri tvorbe tejto práce.

Obsah

1 Úvod **2**

2 Komunikácia v letectve **3**
2.1 Pravidlá prenosu 3
2.2 Kontaktné procedúry 5
2.3 Volacie znaky 5

3 Spracovanie prirodzeného jazyka **8**
3.1 Vnorenie slov (číselná reprezentácia) 8
3.2 Rozpoznanie pomenovaných entít 9
3.3 Transformery 12
3.4 BERT 16

4 Implementácia NER modelu **19**
4.1 Návrh 19
4.2 Datová sada 19
4.3 Evaluácia modelu 22
4.4 Model rozpoznania volacích znakov 22
4.5 Rozšírenie detekcie 26

5 Systém detekcie a rozpoznania volacích znakov **27**
5.1 Úloha - detekcia volacieho znaku 27
5.2 Úloha - rozpoznanie ICAO volacieho znaku 28
5.3 Konverzia datovej sady 28
5.4 Použitie enkodérových modelov 29
5.5 Naivné rozpoznanie 30
5.6 Pokrytie klasifikátorov 33
5.7 Výsledky pokrytia 34
5.8 Porovnanie modelov 38
5.9 Dosiahnuté výsledky 39
5.10 Perspektívy a možnosti rozvoja práce 41

6 Záver **42**

Literatúra **43**

A Obsah priloženého pamäťového média **46**

B Plagát **47**

Kapitola 1

Úvod

Komunikácia v doprave je dôležitým aspektom bezpečnosti, obzvlášť pri letovej a námornej preprave. Jedná sa primárne o komunikáciu medzi ATC (riadenie letovej prevádzky) a posádkou lietadla. Komunikácia je obojstranná a riadi sa vymedzenými pravidlami. Za účelom minimalizovania chýb bol navrhnutý systém dorozumievania nazývaný „letecká angličtina“ (aviation English). Vymedzuje akým spôsobom ATC informuje, dáva pokyny alebo odpovedá pilotovi. Prípadne akým spôsobom pilot informuje alebo žiada o povolenia pre rôzne úkony.

Aj napriek tomu, že tento systém má presne vymedzené pravidlá, nie vždy komunikácia prebieha úplne podľa nich. Do rádiovej komunikácie vstupujú aj faktory ako je šum alebo odlišná výslovnosť v rôznych častiach sveta.

Vzhľadom k narastajúcemu počtu letov a tým aj objemu súvisiacej rádiovej komunikácie sa zvyšuje potreba mechanizmov pre prácu s týmito dátami. Pre efektívne triedenie alebo vyhľadávanie v takejto komunikácii je vhodné využiť práve volacie znaky lietadiel.

Táto práca sa zaoberá spracovaním dát z letovej komunikácie s cieľom detekcie a rozpoznania volacích znakov. Vychádza z predpokladu, že vstupné dáta sú textové prepisy komunikácie. Jedným z cieľov je vytvoriť model, ktorý bude s dostatočnou presnosťou schopný tieto volacie znaky rozpoznať a bude mať vhodnú výpočetnú náročnosť.

Rozpoznanie tejto komunikácie je realizované vhodným NER modelom schopným tagovania slov v sekvencii. V súčasnosti sú na tento účel používané hlavne modely založené na transformeroch.

Trénovaných je viacero modelov a tieto sú následne porovnávané na základe testovacích sád. Z modelov je vybraný najvhodnejší pre nasadenie. Pre detekciu volacích znakov bude implementovaná služba, využívajúca vytvorený model. Detekované volacie znaky budú ďalej konvertované na ich ekvivalent zo zoznamu registrovaných znakov. Služba následne vráti výsledky s volacími znakmi v ICAO formáte.

V kapitole 2 sú bližšie popísané princípy leteckej komunikácie, so zameraním na použitie volacích znakov. Kapitola 3 približuje techniky používané k účelu počítačového spracovania prirodzeného jazyka a rozpoznávania špecifických prvkov. Práca ďalej v kapitole 4 zahŕňa návrh systému pre detekovanie volacích znakov a popisuje kroky potrebné k jeho implementácii. Vytvorený systém je následne vyhodnotený na testovacích dátach.

V kapitole 5 je preberaná potreba systému pre úlohy detekcie a rozpoznania volacích znakov z ľudských ale aj automatizovaných prepisov. Navrhnuté a implementované spôsoby sú vzájomne porovnané vrátane porovnania s alternatívnou technológiou.

Kapitola 2

Komunikácia v letectve

Táto kapitola približuje spôsob akým prebieha komunikácia medzi ATC (riadenie letovej prevádzky, angl. air traffic control) a posádkou lietadiel pomocou VHF (veľmi krátke vlny, angl. very high frequency) kanálov. Procedúry sú prebrané z knihy [1] a príručky [7].

Rádiová VHF komunikácia poskytuje spôsob akým môžu piloti a pozemný personál spolu komunikovať. Informácie a inštrukcie prenášané v tejto komunikácií sú veľmi dôležité pre bezpečnú a efektívnu prevádzku lietadiel. Jedným z faktorov pri incidentoch a nehodách ktoré v minulosti nastali, bolo používanie neštandardných procedúr alebo frazeológie. ICAO[1] z tohto dôvodu vyžaduje, aby bola používaná štandardizovaná frazeológia pri všetkých špecifikovaných situáciách. Jedine v prípadoch, keď štandardizovaná frazeológia nie je aplikovateľná pre daný prenos, môžu komunikujúce strany použiť prirodzený jazyk.

2.1 Pravidlá prenosu

Pravidlá prenosu sú určené na minimalizovanie rušenia od iných staníc, prípadne šumu spôsobeného prenosom alebo použitými zariadeniami. Tón reči komunikujúcich by mal byť čistý a zrozumiteľný. Rýchlosť reči by nemala presahovať 100 slov za minútu, čo je pomalšie ako bežná konverzačná rýchlosť. Niektoré inštrukcie je treba zapisovať druhou stranou, a preto by mala byť rýchlosť reči v takýchto prípadoch znížená.

Pre zrýchlenie komunikácie nie je treba používať fonetického hláskovania, pokiaľ neexistuje riziko, že tým bude ovplyvnený príjem a zrozumiteľnosť správy. V mnohých prípadoch však je potrebné používať hláskovanie a to napríklad pri prenose hodnôt ako identifikácia lietadiel, ranvejí, staníc a ďalších. Príklad rovnakej inštrukcie bez a s fonetickým hláskovaním:

> Pôvodná veta: Proceed to map grid DR98
>
> Veta s hláskovaním: Proceed to map grid Delta-Romeo-Niner-Ait

Použitie „Delta“ namiesto „D“ zabráni zámene medzi „DR98“ a „BR98“ alebo „TR98“. Pri prenášaní správy obsahujúcej označenie a druh lietadla, je potreba vyslovovať každé písmeno volacieho znaku lietadla osobitne použitím fonetického hláskovania. Slová popísané v tabuľke 2.1 majú byť používané pri tomto hláskovaní. Pokiaľ je jazyk komunikácie angličtina, čísla majú byť vyslovované podľa tabuľky 2.2.

[1] ICAO – Medzinárodná organizácia pre civilné letectvo, angl. International Civil Aviation Organization

Písmeno	**Slovo**	**Výslovnosť (anglicky)**
A	Alpha	AL FAH
B	Bravo	BRAH VOH
C	Charlie	CHAR LEE alebo SHAR LEE
D	Delta	DELL TAH
E	Echo	ECK OH
F	Foxtrot	FOKS TROT
G	Golf	GOLF
H	Hotel	HO TELL
I	India	IN DEE AH
J	Juliett	JEW LEE YET
K	Kilo	KEY LOH
L	Lima	LEE MAH
M	Mike	MIKE
N	November	NO VEM BER
O	Oscar	OSS KAH
P	Papa	PAH PAH
Q	Quebec	KEH BECK
R	Romeo	ROW ME OH
S	Sierra	SEE AIR RAH
T	Tango	TANG GO
U	Uniform	YOU NEE FORM alebo OO NEE FORM
V	Victor	VIK TAH
W	Whiskey	WISS KEY
X	X-ray	ECKS RAY
Y	Yankee	YANG KEY
Z	Zulu	ZOO LOO

Tabuľka 2.1: Fonetická abeceda pre vyslovovanie písmen.

Číslo alebo číselný prvok	**Výslovnosť (anglicky)**
0	ZE-RO
1	WUN
2	TOO
3	THREE
4	FOW-er
5	FIFE
6	SIX
7	SEV-en
8	AIT
9	NIN-er
Decimal	DAY-SEE-MAL
Hundred	HUN-dred
Thousand	THOU-SAND

Tabuľka 2.2: Výslovnosť čísel podľa fonetickej abecedy.

2.2 Kontaktné procedúry

Komunikácia z veľkej časti pozostáva z povolení alebo pokynov pre vykonanie určitého úkonu od ATC pre posádku lietadla. Niektoré prenosy majú len informatívny charakter pre podporu rozhodovania. Obsah týchto povolení siaha od detailných popisov trasy a letových hladín ku krátkym povoleniam na pristátie.

Zabezpečenie spoľahlivosti prenosu

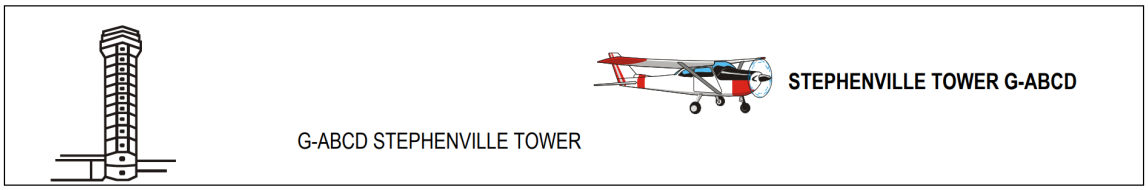
Na VHF frekvenciách prebieha aj komunikácia medzi ATC a pozemným personálom.

V niektorých prípadoch si piloti pokyny potrebujú zapísať, v takom prípade je potrebné zdeľovať informácie pomalšie aby sa predišlo zbytočnému opakovaniu pokynov. Pre zvýšenie bezpečnosti bolo zavedené opakované znovu-prečítanie (read-back). Dôležitosť znovu-prečítania je priamo spojená z vážnosťou následkov v prípade zlého porozumenia povolenia alebo pokynu. Prísne dodržiavanie procesu zaručuje nie len to, že príjem prebehol v poriadku, ale aj správne zdelenie zo strany odosielateľa. Taktiež zabezpečuje, že pokynom alebo povolením sa riadi len jedno lietadlo.

Pri väčšine inštrukcií je dostatočné aj upovedomenie druhej strany o prijatí inštrukcie alebo povolenia. Príklad znovu-prečítania povolenia na pristátie z reálnych dát:

> ATC: Hotel Golf November wind zero two zero degrees two knots runway three two cleared to land.
>
> Lietadlo: Cleared to land runway three two Hotel Golf November.

Pri nadväzovaní spojenia je potrebné, aby posádka lietadla použila plný volací znak lietadla a taktiež pozemnej stanice. Príklad je na obrázku 2.1.



Obr. 2.1: Nadviazanie spojenia, prevzaté z [7].

Pozemná stanica prípadne posádka lietadla môže taktiež kontaktovať všetky lietadlá vo svojom okolí volaním `ALL STATIONS`. Na takéto volanie sa zväčša nečaká žiadna odpoveď.

Ak sa stane, že jedna zo strán si nie je istá, či prijala správu korektne, môže použiť volanie `SAY AGAIN ...` s prípadným upresnením neporozumenej časti. Druhá strana následne zopakuje poslednú správu.

2.3 Volacie znaky

Pre jednoznačnú identifikáciu staníc, lietadiel a iných súčastí letovej prevádzky sú používané volacie znaky (angl. call sign). Príklady volacích znakov niektorých jednotiek: centrum pre riadenie oblasti má volací znak `CONTROL`, riadenie príletov – `APPROACH`, prílety na radare – `ARRIVAL`, odlety na radare – `DEPARTURE`, riadenie letiska – `TOWER`, riadenie pohybu na zemi – `GROUND`.

Volacie znaky lietadiel môžu mať pri nadväzovaní spojenia viacero podôb. Tieto dlhé alebo celé formy volacích znakov sú popísané v tabuľke 2.3.

Typ	**Príklad**
a) znaky odpovedajúce registračnému značeniu lietadla	G - ABCD alebo Cessna G - ABCD
b) rádiové označenie spoločnosti operujúcej s daným lietadlom, nasledované poslednými štyrmi znakmi registračného značenia lietadla	FASTAIR DCAB
c) rádiové označenie spoločnosti operujúcej s daným lietadlom, nasledované identifikátorom letu	FASTAIR 345

Tabuľka 2.3: Typy dlhých volacích znakov.

Po úspešnom nadviazaní spojenia a za predpokladu, že žiadne nedorozumenie by nemalo nastať, je možné skrátiť volacie znaky podľa kategórií spôsobom v tabuľke 2.4.

Typ	**Príklad**
a) prvé a najmenej dva posledné znaky registračného značenia lietadla	CD alebo Cessna CD
b) rádiové označenie spoločnosti operujúcej s daným lietadlom, nasledované aspoň poslednými dvomi znakmi registračného značenia lietadla	FASTAIR AB
c) nemá skrátenú formu	–

Tabuľka 2.4: Typy skrátených volacích znakov.

Posádka lietadla môže používať svoj skrátený volací znak až po tom, ako je týmto spôsobom kontaktovaná leteckou stanicou. Lietadlo by nemalo zmeniť typ svojej volacej značky počas letu s výnimkou prípadu, keď môže nastať nedorozumenie z dôvodu podobných volacích znakov. V takýchto prípadoch môže riadenie letovej prevádzky dať pokyn posádke lietadla k dočasnej zmene volacieho znaku.

Ďalšie typy volacích znakov

Pri komerčných operátoroch je zvyčajne používaný typ C volacieho znaku, kde je použitý volací znak daného operátora registrovaný u ICAO alebo FAA[2] v USA. Identifikácia letu je často rovnaká ako číslo letu, ale môže byť odlišná aby sa predišlo zámene v prípade viacerých letov s podobnými číslami. Použitie tohto typu volacích znakov sa líši podľa krajiny, napríklad FAA umožňuje pri použití tejto kategórie vyslovovanie čísel v celku bez vyhláskovania jednotlivých číslic. Zoznam priradených volacích znakov jednotlivých operátorov je dostupný na [23].

Volacie znaky môžu byť ďalej modifikované podľa ďalších faktorov, ako sú napríklad fyzické vlastnosti lietadla. Keď lietadlo spadá podľa hmotnosti do kategórie SUPER alebo HEAVY, musia názvy týchto kategórií nasledovať hneď za volacím znakom. Tento postup je dôležitý kvôli silnejším turbulenciám ktoré takéto lietadla za sebou zanechávajú.

[2]FAA – Federal Aviation Administration je regulačný orgán pre letectvo v Spojených štátoch amerických

Určité lety potrebujú prednostný prístup k službám letovej kontroly alebo pozemného personálu. Núdzové lety záchrannej služby majú preto taktiež volací znak pozmenený. Ten sa skladá zo slova *MEDEVAC* prípadne *HELIMED* v Spojenom Kráľovstve, za ním nasledujú ostatné časti volacieho znaku.

Cvičné lety študentov majú prefix *STUDENT*. V rôznych štátoch môžu existovať výnimky formátu volacích znakov pre určité druhy letov. Napríklad znaky policajných letov v Spojenom Kráľovstve pozostávajú z rádiového volacieho znaku *POLICE*, za ním nasledujú ďalšie časti volacieho znaku [5].

Kapitola 3

Spracovanie prirodzeného jazyka

Kapitola poskytuje úvod do spôsobov spracovania prirodzeného jazyka (NLP). NLP rieši spôsoby akými je možné počítačovo spracovávať a analyzovať veľké množstvo dát v prirodzenom jazyku. Pôvodne sa na tento účel používali symbolické a štatistické metódy. Kvôli ich výpočetnej náročnosti a relatívne nízkej efektivite pri väčšine úloh, sú dnes zväčša používané prístupy využívajúce hlboké neurónové siete. Časti tejto kapitoly čerpajú z knihy [8].

3.1 Vnorenie slov (číselná reprezentácia)

Jednou z najpodstatnejších úloh pri analyzovaní jazyka je popis významu jednotlivých slov. Každé slovo v prirodzenom jazyku má význam, ktorý dokážeme, ako ľudia popísať inými slovami, buď pomocou synoným alebo definícií.

Takýto spôsob popisu významu je pre nás, ako ľudí zväčša postačujúci. Významové vzťahy medzi slovami sú ale zložité a takýto popis je pre mnoho úloh počítačového spracovania jazyka nepostačujúci alebo neefektívny.

V štatistickom spracovaní prirodzeného jazyka bol zvolený spôsob reprezentácie slova pomocou distribučnej sémantiky. Tento systém má za úlohu zachytiť význam konkrétneho slova na základe iných slov, ktoré sa vyskytujú v jeho okolí v súbore textov pomocou matíc. Stĺpce matice spoločného výskytu „co-occurrence matrix“ reprezentujú slová zo slovnej zásoby. Hodnoty predstavujú počet spoločných výskytov slov v korpuse. Matice preto často bývajú veľmi dlhé. Zvyčajne sa tento spôsob výpočtu používa s určitým kontextovým oknom, takže spoločný výskyt je pripočítaný, len ak sa slová nachádzajú v blízkosti v rámci textu.

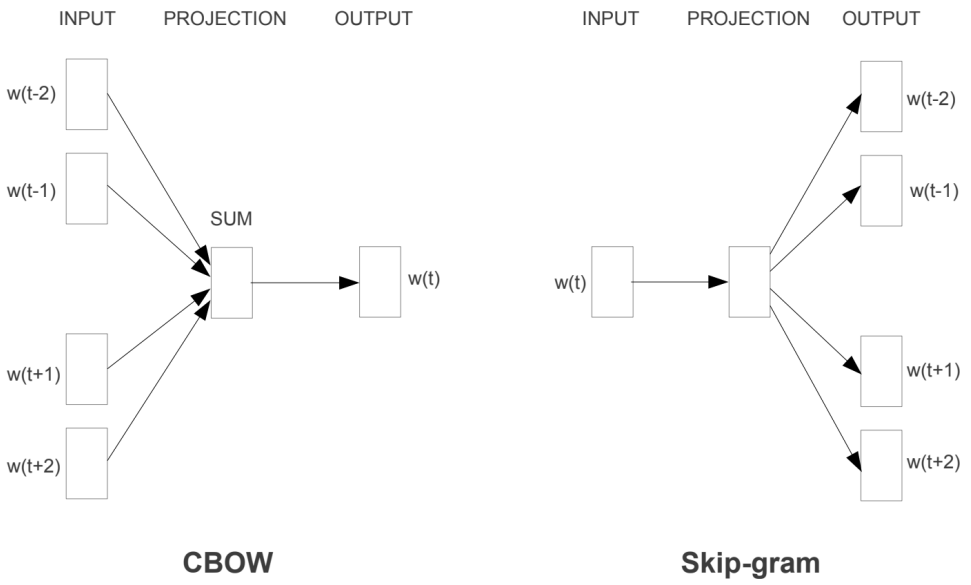
Mnoho hodnôt z takýchto vektorov bude nulových, pretože niektoré slová sa v blízkosti seba v korpuse nenachádzajú. Z tohto dôvodu sú takéto matice nazývané aj „riedke“.

Statické slovné vektory

Ďalší spôsob výpočtu vektorov, ktoré zachycujú význam slov je skip-gram a CBOW zo súboru postupov Word2Vec predstavený v článku [14]. Pre tento účel používa mnohorozmerné (zvyčajne v rádoch stoviek dimenzií) vektory. Takéto vektory sú nazývané aj „husté“ pretože neobsahujú prvok pre každé slovo zo slovnej zásoby. Preto nie je možné jednoznačne interpretovať význam jednotlivých prvkov týchto vektorov.

Takéto matice sú schopné lepšie vystihnúť rôzne sémantické vzťahy medzi slovami.

Namiesto určovania počtu spoločných výskytov slov, je trénovaný klasifikátor pre úlohu určenia pravdepodobnosti ich spoločného výskytu.



Obr. 3.1: Porovnanie úlohy klasifikátorov CBOW a Skip - gram, prevzaté z [14]. CBOW predikuje slovo na základe okolitých slov a Skip - gram opačne.

Pri algoritme CBOW je natrénovaný klasifikátor, ktorý určuje pravdepodobnosť stredového slova `w` za predpokladu kontextových slov `c`.

Algoritmus skipgram je naopak trénovaný pre úlohu predikcie kontextových slov za predpokladu stredového slova.

Pre účely porovnania slov, je potrebné vypočítať podobnosť vektorov `w` a každého slova z okna `c`, ktorá je vyjadrená ich skalárnym súčinom. Následne je treba tieto skalárne súčiny vynásobiť pomocou logistickej alebo sigmoidnej funkcie pre získanie pravdepodobnosti ich spoločného výskytu.

Každé slovo v tomto algoritme má 2 vektory, jeden je použitý keď je toto slovo tzv. stredové a druhé je pre reprezentáciu tohoto slova ako „vonkajšie slovo“ (outside word). Učenie začína náhodne vygenerovanými vektormi pre celú slovnú zásobu. Hodnoty jednotlivých vektorov iteratívne upravuje tak, aby sa znížil rozdiel medzi očakávaným a navrhovaným vektorom.

Ďalším algoritmom pre tvorbu statických vektorov je GloVe [3]. Tento model využíva globálne slovné štatistiky. Pracuje s pomermi pravdepodobností z matíc spoločného výskytu. Vektory zachytávajú syntaktické a sémantické vzťahy medzi slovami. S touto reprezentáciou slov je taktiež možné vykonávať vektorovú aritmetiku a výsledky sú blízke tomu, ako by očakávali ľudia [9].

3.2 Rozpoznanie pomenovaných entít

Jednou z úloh spracovania prirodzeného jazyka je rozpoznanie pomenovaných entít (angl. named entity recognition, NER). Tieto entity spadajú do určitých kategórií, podľa ktorých sú im priradené tagy. Môže sa jednať napríklad o určenie, ktoré slová reprezentujú osoby, organizácie, geografické lokácie a iné. Príklad textu s tagmi prevzatého z novín [18]:

> Spoločnosť [**ORG** SpaceX] získa od investorov v novom kole financovania [**MON** 750 miliónov dolárov]. Je medzi nimi firma, ktorá bola aj spoluinvestorom v [**PER** Muskovej] dohode o prevzatí [**ORG** Twitteru], či spoločnosti [**ORG** Alphabet] a [**ORG** Fidelity]. Investícia tak ohodnotí [**ORG** SpaceX] na [**MON** 137 miliárd dolárov].

Pre rôzne aplikácie je možné takéto tagovanie realizovať pre rozličné triedy entít. V zdravotníctve sa môže napríklad jednať o názvy diagnóz prípadne proteínov alebo chemických látok.

Možnosti tagovania entít

Pri realizácií tohoto tagovania je možné využiť metódy podobné z POS (angl. part of speech) tagovania, kde ide o určenie slovných druhov v rámci vety. Pri POS tagovaní nie je problém so segmentáciou, pretože každému slovu prislúcha práve jeden tag. Segmentácia môže byť problémom pri rozpoznaní pomenovaných entít, kde treba určiť či slovo patrí do nejakého tagu prípadne, ktoré slová z vety jeden tag zahŕňa.

Ďalším problémom je zaradenie slova do určitej triedy. Keď spomenieme skratku SNP, môže ísť o historickú udalosť Slovenského národného povstania, alebo aj o niektoré z mnohých miest na Slovensku pomenovaných po tejto udalosti. Každá z týchto entít môže byť zložená z viacerých slov. Tieto slová môžu patriť podľa kontextu do inej skupiny.

Jedným z prístupov označovania entít je BIO tagging. Pri tomto postupe je označený každý počiatočný token entity prefixom `B-`, pokračujúce tokeny `I-` a tokeny mimo entíty `O`. Tento prístup spôsobí nárast počtu tagov na $2n + 1$, kde n je počet druhov entít. Pri variante IO tagging je znížený počet tagov, ale zároveň je stratená informácia o začiatočnom tokene. BIOES tagging naopak pridáva koncový prefix `E-` k poslednému tokenu entity.

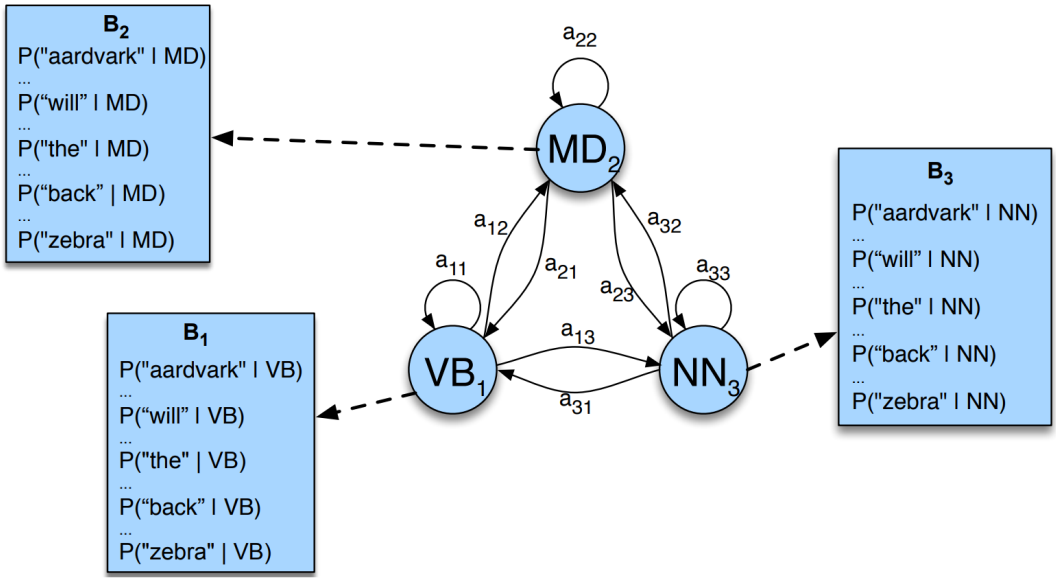
Skrytý Markovov model

Pre priradenie tagu ku každému slovu vety je možné použiť HMM (angl. Hidden Markov model). Markovov proces (angl. Markov chain) je model, ktorý zisťuje pravdepodosti ďalších stavov z nejakej postupnosti stavov. Markovov proces predpokladá, že k určeniu ďalšieho stavu, stačí poznať hodnotu aktuálneho stavu. Predošlé stavy nemajú žiaden vplyv na budúce hodnoty. Tieto stavy môžu byť reprezentované slovami vo vete a tento model v takom prípade určuje pravdepodobnosť postupnosti slov.

Tagy sa vo vetách nenachádzajú, preto sú k účelu ich rozpoznania používané Skryté Markovove modely. HMM umožňuje pracovať s pozorovateľnými javmi, ako sú slová vo vete a javmi skrytými, ako napríklad tagy, ktoré nevidno priamo.

Markovov model prvého rádu pracuje s 2 predpokladmi. Pravdepodobnosť ďalšieho stavu podľa prvého predpokladu záleží len od stavu aktuálneho. Ďalej predpokladá, že pravdepodobnosť výstupného pozorovania záleží len od stavu, ktorý vygeneroval konkrétne pozorovanie a nie od iných stavov alebo pozorovaní.

Pre priraďovanie tagov HMM využíva matice pravdepodobností A a B. Matica A obsahuje pravdepodobnosť, že nejaký tag bude nasledovať za aktuálnym tagom. Táto pravdepodobnosť je vypočítaná podielom výskytov týchto tagov v určenom poradí a celkovým výskytom aktuálneho tagu.



Obr. 3.2: Schéma HMM pre POS tagovanie, prevzaté z [8].

Pravdepodobnosti B označujú pravdepodobnosť, že zvolený tag bude priradený ku konkrétnemu slovu. Pri každom modeli so skrytými premennými akým je aj Skrytý Markovov model, nazývame úlohu zisťovania sekvencie skrytých premenných za predpokladu sekvencie pozorovaní dekódovaním. Pre dekódovanie sekvencií pomocou HMM je používaný Viterbiho algoritmus. Algoritmus najprv zostaví maticu pravdepodobností, kde sa nachádza stĺpec pre každé pozorovanie a riadok pre každý stav. Bunky tejto matice sú vypočítané, rekurzívnym prechodom po najpravdepodobnejšej ceste, ktorá povedie k tejto bunke. Algoritmus takto vypočíta stavy v čase t pomocou predošlých stavov v $t-1$.

Na obrázku 3.2 je možné vidieť ako Skrytý Markovov model určuje pravdepodobnosti tagov za predpokladu pozorovaní, tj. slov.

Podmienené náhodné polia

Skrytý Markovov model neberie v úvahu niektoré informácie akými sú napríklad okolité slová. CRF (Podmienené náhodné polia, angl. Conditional random fields) [10] rozlišujú medzi viacerými pravdepodobnými sekvenciami tagov. CRF priraďuje pravdepodobnosť celej postupnosti tagov zo všetkých možných postupností, za danej vstupnej sekvencie slov.

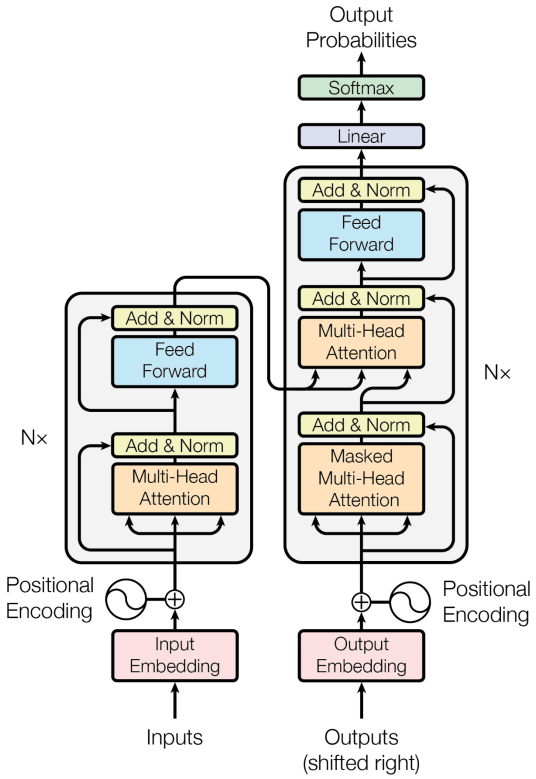
Základnou myšlienkou CRF je modelovať pravdepodobnosť sekvencie tagov y vzhľadom na sekvenciu vstupných premenných x. To sa uskutočňuje definovaním množiny príznakov $f_i(x, y)$, ktoré zachytávajú vzťahy medzi vstupnými premennými a tagmi a použitie týchto príznakov na výpočet pravdepodobnosti každej možnej postupnosti tagov.

Ak chceme použiť CRF na NER, musíme najprv definovať súbor príznakov, ktoré zachytávajú vzťahy medzi vstupnými premennými a pomenovanými entitami. Tieto príznaky môžu zahŕňať samotné slová, ako aj informácie o ich POS tagoch, použití veľkých písmen a pozícii vo vete. Nájdenie najlepšej cesty je možné u CRF realizovať podobne ako u HMM upraveným Viterbiho algoritmom.

3.3 Transformery

Transformery boli predstavené v článku Attention Is All You Need [22]. Nadväzujú na architektúru RNN[1] a LSTM[2], ale riešia problém propagácie informácií.

Na obrázku 3.3 je vidno architektúru transformeru. Tento model má 2 hlavné súčasti. Prvou je enkodér, ktorý sa nachádza v ľavej časti obrázku. Na pravo je možné vidieť blok dekodéru s výstupnou vrstvou.



Obr. 3.3: Schéma architektúry transformeru, prebraté z [22].

Vstupné embeddingy

Vstupom do modelu je sekvencia slov, ktorá je následne konvertovaná podľa slovnej zásoby na indexy konkrétnych slov. Z nich vznikne vektor podľa dĺžky vstupu.

Vrstva vstupných embeddingov priradí každému indexu vektor. Tento vektor je s počiatku inicializovaný ako náhodný, ale neskôr sú to naučené hodnoty. V pôvodnej práci sa jednalo o slovné embeddingy o dĺžke 512 prvkov. Takže výstupom tejto vrstvy je vektor embeddingov o dĺžke vstupnej sekvencie. Pre vstupnú sekvenciu môže vyzerať úprava napríklad takto:

[1]RNN (Recurrent neural network) – neurónová sieť schopná prenášať informácie sekvenčne medzi jednotlivými prvkami jednej vrstvy.

[2]LSTM (Long Short Term Memory) – druh RNN schopný ukladať informácie krátkodobo alebo dlhodobo (tisícky časových krokov).

Vstupné tokeny:	Swiss	one	five	three	turn
Vektor indexov:	[102,	244,	6568,	438,	297]
Vektor embeddingov:	[[0.23	[-0.32	[0.12	[0.93	[0.89
	0.45	0.94	-0.03	0.88	0.97
	...	...	...	...	...
	-0.11],	0.73],	0.35],	-0.59],	-0.76]]

Pri iných architektúrach ako napríklad LSTM, nie je potrebné dodatočne pridávať informácie o pozícii tokenu v rámci sekvencie pretože tieto modely spracovávanú vstupy postupne a tak vie model prirodzene rozoznať pozíciu aktuálneho slova. Transformery spracovávajú sekvenciu paralelne a tak je potrebné použiť pozičné embeddingy, s rozmerom slovných embeddingov d. V pôvodnej práci boli predstavené tieto spôsoby výpočtu pozičných embeddingov s odlišnými frekvenciami:

$$PE(pos, 2i) = sin\left(\frac{pos}{1000^{\frac{2i}{d}}}\right) \tag{3.1}$$

$$PE(pos, 2i+1) = cos\left(\frac{pos}{1000^{\frac{2i}{d}}}\right) \tag{3.2}$$

Pričom pos je pozícia tokenu a i je označenie aktuálneho rozmeru, pre ktoré je výpočet realizovaný. Pozičné embeddingy sú následne sčítané so slovnými embeddingmi a výsledný vektor môže byť následne spracovaný v bloku enkodéru.

Enkodér

Zjednodušene povedané, úlohou enkodéru je premeniť vstupnú sekvenciu vektorov na ich vnútornú reprezentáciu na základe naučených váh. Tieto vektory sú na rozdiel od statických vektorov popísaných v časti 3.1 kontextuálne, tj. viažu sa na obsah vety.

Jeden blok enkodéru sa skladá z vrstvy pozornosti. Výstupom tejto vrstvy je vektor, ktorého prvky sú upravené naučenými váhami pozornosti. Kvôli predídeniu straty informácií vrstvou pozornosti je tu tzv. „Add & Norm“ (sčítanie a normalizizácia) vrstva, ktorá sčíta výstupný vektor vrstvy pozornosti s reziduálnym spojením vstupnej reprezentácie. Vrstva ďalej normalizuje vektor, ktorý prejde do doprednej vrstvy. Výstup z doprednej vrstvy je taktiež sčítaný a normalizovaný reziduálnym spojením.

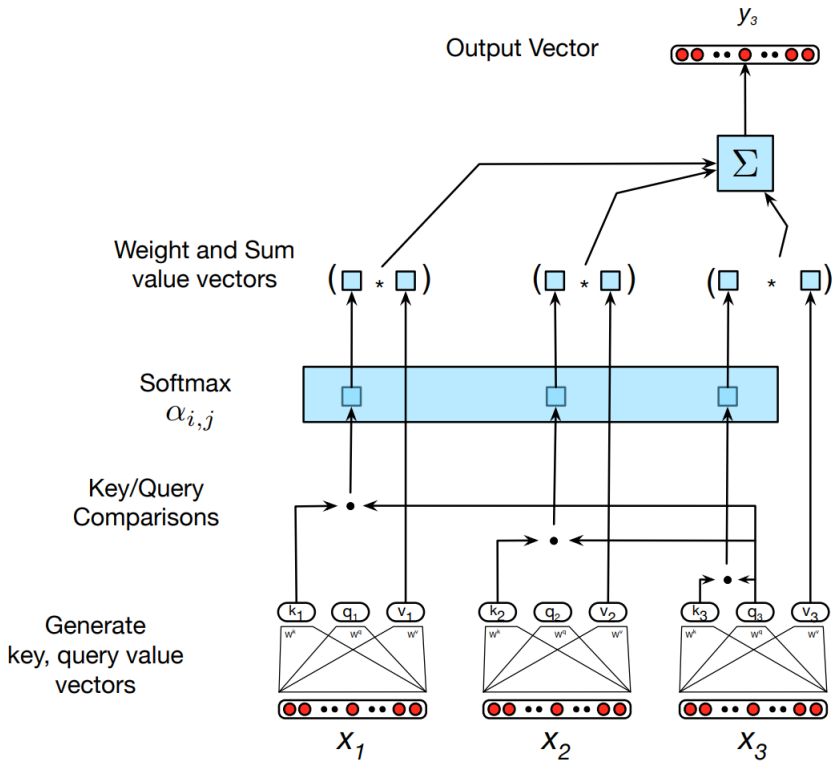
Transformery väčšinou používajú viacero blokov enkodéru. V tomto prípade sú výstupy jedného z bloku použité ako vstupy pre ďalší. Finálny výstup je braný vzhľadom k poslednému bloku.

Vrstva pozornosti

Pri prenose informácií cez sériu opakovaných prepojení v LSTM dochádza k strate relevantných informácií. Pri LSTM je taktiež problémom paralelizácia, kvôli sekvenčnému prenosu informácií.

Transformery používajú vrstvu pozornosti (angl. attention layer) znázornenú na obrázku 3.4. Mechanizmus pozornosti umožňuje zachytiť informácie z ľubovoľne veľkého kontextu, bez nutnosti presunu týchto dát cez veľké množstvo prepojov.

U spätne prepájajúcej vrstvy pozornosti má model pri spracovaní každého vstupu prístup ku všetkým vstupom po aktuálny vstup, ale nepozná ďalšie vstupy za ním.



Obr. 3.4: Výpočet hodnoty prvku za použitia pozornosti, prebraté z [8].

Tento prístup poskytuje možnosť porovnávať aktuálny vstup k ostatným vstupom a zisťovať ich dôležitosť v aktuálnom kontexte. Najjednoduchším spôsobom porovnávania prvkov vrstvy pozornosti je ich skalárny súčin. Výsledkom tejto operácie je skóre. Skalárny súčin vracia hodnoty od $-\infty$ do ∞ a so zvyšujúcou hodnotou narastá podobnosť prvkov. Výsledky týchto porovnaní sú následne normalizované použitím softmax funkcie pre vytvorenie vektoru váh, ktorý symbolizuje dôležitosť každého vstupného prvku k aktuálnemu.

$$\alpha_{ij} = \text{softmax}(\text{score}(x_{ij})) \; \forall_{j \leq i} \tag{3.3}$$

$$\alpha_{ij} = \frac{\exp(\text{score}(x_i, x_j))}{\sum_{k=1}^{i} \exp(\text{score}(x_i, x_k))} \; \forall_{j \leq i} \tag{3.4}$$

Na základe proporčného skóre α je možné vypočítať výstupnú hodnotu y_i súčtom hodnôt vstupných prvkov vynásobených ich príslušnými α hodnotami.

$$y_i = \sum_{j \leq 1} x_j \alpha_{ij} \tag{3.5}$$

Tento zjednodušený prístup v tejto podobe nepodporuje učenie a zmeny váh. Pre tento účel transformery používajú dodatočné parametre vo forme množín matíc, ktoré pracujú nad vstupmi. Vstupné dáta majú rôzne role vzhľadom na to ako sa s nimi pracuje:

- *query (Q)* nazývame vstup, ktorý bude algoritmus porovnávať so všetkými ostatnými vstupnými vektormi,

- *key (K)* pokiaľ sa jedná o prechádzajúci vstup porovnávaný s aktuálnym vektorom,
- *value (V)* v prípade, že sa jedná o aktuálnu hodnotu pomocou ktorej je počítaný výstup.

Pre zachytenie týchto rolí transformer využíva váhy W^Q, W^K a W^V. Tieto váhy budú použité pre výpočet lineárnych transformácií pre každý vstup x.

$$q_i = W_x^Q;\ k_i = W^K;\ v_i = W^V x_i$$

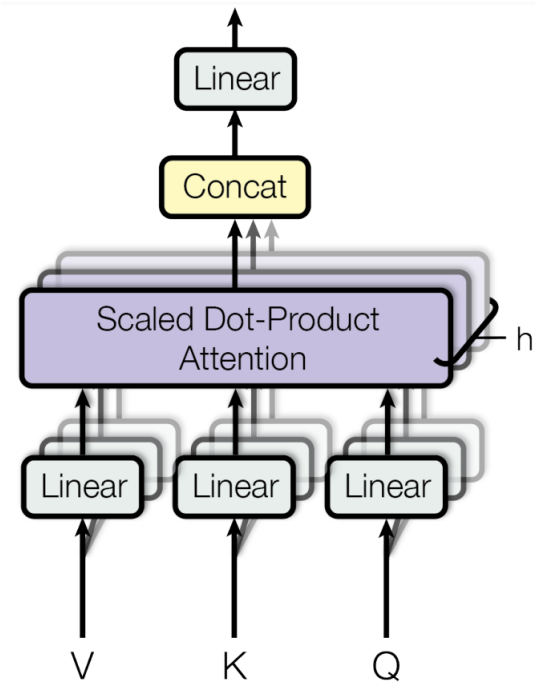
Postup výpočtu jedného výstupu s použitím mechanizmu pozornosti je vidno na obrázku 3.4. Nakoľko celý výpočet y_i je počítaný nezávisle celý tento proces je možné počítať paralelne použitím matíc. Výpočet „Scaled Dot-Product Attention“ je následne popísaný rovnicou 3.6.

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

$$SelfAttention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.6}$$

Viac-hlavá vrstva pozornosti

Rôzne slová môžu súvisieť s ostatnými mnohými spôsobmi zároveň. Pri mechanizme viac-hlavej pozornosti (angl. multiheaded attention) sa používa viacero hlavíc pozornosti, pričom každá hlava pozornosti je samostatný „pozorovateľ“ s vlastnými váhami, ktorý sa sústredí na rôzne časti vstupu. Tieto hlavy pozornosti sa potom spoja dokopy, aby sa vytvoril konečný výstup.



Obr. 3.5: Viac-hlavá vrstva pozornosti, prebrané z [22].

Pre implementovanie tejto myšlienky má každá hlava svoju množinu matíc W_i^K, W_i^Q a W_i^V. Výstup vrstvy viac-hlavej pozornosti s h hlavami pozostáva z h vektorov rovnakej dĺžky. Tieto sú následne skombinované a upravené do pôvodného rozmeru d_m. Kombinácia je docielená konkatenáciou výstupov každej z hláv a následným použitím lineárnej projekcie pre úpravu do pôvodného rozmeru d_m.

$$MultiHeadAttn(Q, K, V) = W^O(head_1 \oplus head_2 \cdots \oplus head_h) \tag{3.7}$$

$$head_i = SelfAttention(W_i^Q X, W_i^K X, W_i^V X) \tag{3.8}$$

Dekodér

Úlohou dekodéru v skratke je zo vstupnej vektorovej reprezentácie vygenerovať nový token. Vstupom do dekodéru je výstupný vektor enkodéru a tiež predošlý výstup dekodéru.

Prvou vrstvou enkodéru je kódovanie predošlého výstupu (angl. Output embeddings), ktoré prebieha podobne ako pri enkodéri.

Prvým tokenom ktorý je vstupom do dekodéru, je počiatočný token `<Start>`. Ten je následne konvertovaný na embedding s pridanými pozičnými informáciami. Tento embedding je následne spracovaný viac-hlavou maskovaciou vrstvou pozornosti. Po tomto kroku je pridané reziduálne spojenie.

Následne sú do ďalšej viac-hlavej vrstvy pozornosti predané vektory V a K z enkodéru a vektor Q z predošlej vrstvy. Po vrstve pozornosti nasleduje ešte jedna normalizovaná dopredná vrstva.

Posledným krokom je lineárna vrstva, ktorá môže mať rôzne podoby podľa účelu daného modelu. Pri tradičných jazykových modeloch sa jedná o neurónovú sieť s počtom neurónov rovným veľkosti použitej slovnej zásoby. Nasleduje softmax vrstva pre získanie pravdepodobnosti ďalšieho tokenu.

Týmto spôsobom dokáže model vygenerovať ďalší token. Autoregresívne modely[3] ako napríklad GPT využívajú architektúru dekodéru bez enkodérovej časti [16].

3.4 BERT

Model BERT (Bidirectional Encoder Representations from Transformers), je predtrénovaný jazykový model predstavený v článku [6]. Spomínaný model je predtrénovaný pomocou neoznačených dát na dvoch úlohách pomocou korpusu zloženého z 3300 miliónov slov.

Trénovanie

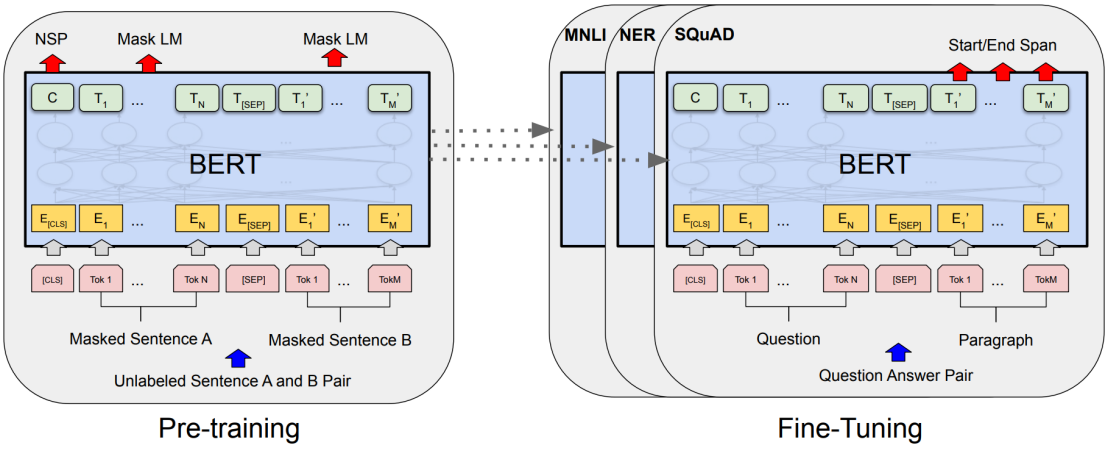
Model nebol trénovaný tradičným spôsobom zľava doprava, tj. na predikciu ďalšieho tokenu. Tento model bol trénovaný na úlohu predikcie maskovaného tokenu. Pri trénovaní preto bolo zo všetkých tokenov 15% náhodne vybraných zamaskovených a model bol učený k ich správnej predikcii. Pre tento účel BERT používa vrstvy pozornosti s obojstranným kontextom.

Ďalšou úlohou bolo predtrénovanie modelu na predikciu nasledujúcej vety. Táto úloha bola zvolená, pretože mnoho úloh spracovania prirodzeného jazyka pracuje s viacerými vetami. Preto je potrebné zaistiť, že model bude schopný rozpoznať vzťahy medzi vetami.

[3] Autoregresívny jazykový model generuje ďalší výstup na základe svojich predošlých výstupov.

Z trénovacieho korpusu boli vyberané páry po sebe nasledujúcich viet A a B. V 50% prípadoch bol trénovací príklad zložený z viet A a B a označený *IsNext*. V ostatných príkladoch bola za A pripojená náhodná veta z korpusu s označením *NotNext*. Táto trénovacia úloha sa ukázala byť užitočná napríklad pre úlohy odpovedania na vety.

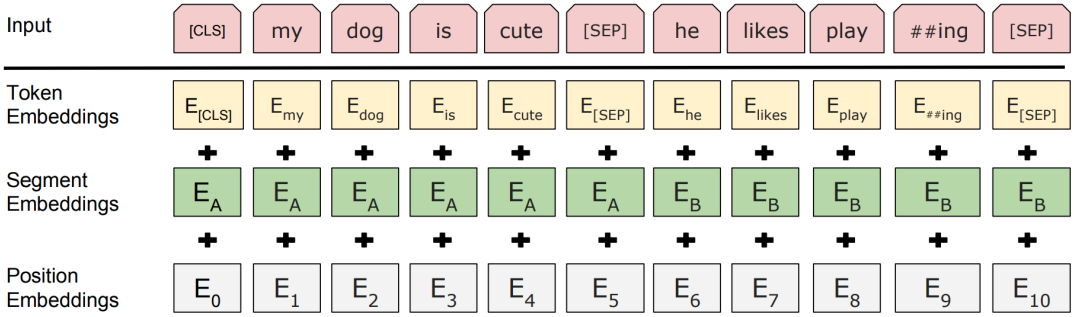
Takto predtrénovaný model je následne možné použiť na mnoho NLP úloh. Pre fine-tuning je BERT model inicializovaný predtrénovanými parametrami. Fine tuning prebieha trénovaním označených dát pre konkrétnu úlohu v spracovaní prirodzeného jazyka.



Obr. 3.6: Schéma modelu BERT, prevzaté z [6].

Architektúra a formát vstupu

Pre všetky úlohy používa BERT rovnakú architektúru s malými obmenami. BERT využíva dizajn obojsmerného transformerového enkodéru. Táto architektúra bere v úvahu kontext z oboch strán. Takže napríklad pre úlohy predikcie maskovaného tokenu využíva aj slová za týmto tokenom.



Obr. 3.7: Vstupná reprezentácia tokenov BERT modelu, prevzaté z [6].

Model $\text{BERT}_{\text{BASE}}$ je zložený z 12 enkodérových blokov L. Využíva vstupné embeddingy s 768 parametrami H. Každý použitý blok má 12 hláv pozornosti A s celkovo 110 miliónom parametrov. Pre porovnanie model $\text{BERT}_{\text{LARGE}}$ má nasledujúce zloženie: L=24, H=1024, A=16, s celkovým počtom parametrov 340 miliónov.

Vstupná reprezentácia môže obsahovať jednu alebo pár viet (napr. <Otázka, Odpoveď>) v jednej sekvencii. Prvý token v každej sekvencii je špeciálny token (`[CLS]`), ktorý slúži pre plnenie klasifikačných úloh pre celú sekvenciu. Pre oddelenie viet vrámci sekvencie je použitý špeciálny token (`[SEP]`) a naučené embeddingy, ktoré určujú do ktorej vety patrí každý z tokenov. Pre každý token je vytvorená vstupná reprezentácia sčítaním príslušného tokenu, segmentu a pozičného embeddingu. Reprezentácia vstupu je znázornená na obrázku 3.7.

Použitie modelu BERT pre úlohy NLP

BERT unifikuje trénovacie postupy pri fine-tuningu. Pre aplikácie využívajúce páry textov umožňuje obojsmerná architektúra mechanizmu pozornosti jednotné enkódovanie do jednej sekvencie. Toto je možné využiť pre páry hypotéza – premisa prípadne pri odpovedaní na otázky. Taktiež je možné na vstup poskytovať samostatný nepárovaný text pre klasifikačné a označovacie úlohy. Pri tomto je využitá reprezentácia tokenom `[CLS]`, ktorý je na výstupe možné interpretovať pre analýzu sentimentu.

Pre použitie modelu BERT môže byť táto architektúra predtrénovaná od počiatočných hodnôt, čo je výhodné pri väčších korpusoch pre špecifickú doménu. Zatiaľ čo predtrénovanie je výpočetne náročné, fine - tuning nevyžaduje veľké výpočetné prostriedky, ale taktiež profituje z väčšieho množstva dát.

V článku [12] bol predstavený model RoBERTa (angl. A Robustly Optimized BERT Pretraining Approach). Tento model používa architektúru modelu BERT. Keďže tento model benefituje z veľkého množstva trénovacích dát, autori tejto práce zvýšili množstvo dát na 160GB z pôvodných 16GB použitých na predtrénovanie modelu BERT. Overfitting[4] nie je problémom ani v tejto konfigurácií a daná architektúra by profitovala z ešte väčšieho množstva dát a dodatočného trénovania.

Model	počet parametrov	trénovacie dáta
BERT	110M	16GB
XLNet	110M	158GB
RoBERTa	125M	160GB
DistilBERT	66M	16GB
ALBERT	12M	16GB

Tabuľka 3.1: Prehľad jazykových modelov, prevzaté z blogu [21].

Problémom transformerových modelov je kvadratická náročnosť mechanizmu pozornosti. Veľké dĺžky vstupu v kombinácií s modelom s vysokým počtom parametrov preto spôsobujú zvyšovanie cien pri škálovaní. Ďalším smerom vývoja enkodérových modelov vychádzajúcich z architektúry BERT je tzv. destilácia. Tento spôsob bol použitý pri trénovaní modelu DistilBERT [19]. Pri tomto princípe je použitý jazykový model s vyšším počtom parametrov pre trénovanie menšieho modelu s nižšou výpočetnou náročnosťou. Pri takomto vývoji je snaha o zachovanie čo najvyššej podobnosti výstupov pôvodného modelu. Porovnanie vybraných modelov je v tabuľke 3.1.

[4]Overfitting – Parametre modelu sa príliš naviažu k testovacím dátam, kde dosahujú správne výsledky, no pri iných druhoch dát model nedosahuje dostatočne dobré výsledky.

Kapitola 4

Implementácia NER modelu

Kapitola popisuje postup návrhu a realizácie systému pre hľadanie volacích znakov vo VHF komunikácií. Obsahuje taktiež dosiahnuté výsledky trénovaných modelov na testovacích sadách.

4.1 Návrh

Postupy používané pri leteckej VHF komunikácii podliehajú stanoveným pravidlám. Táto komunikácia však v realite nie vždy prebieha podľa týchto presne vymedzených pravidiel, popísaných v kapitole 2. V rôznych častiach sveta sú špecifické iné prízvuky a rýchlosť reči v týchto zvukových záznamoch je pomerne vysoká v porovnaní s konverzačnou rýchlosťou [20]. Toto môže znižovať presnosť systémov ASR (systém automatického rozpoznania reči, angl. automatic speech recognition), čo môže do viet pri použití počítačových prepisov zamiešať chybné slová. Tieto a ďalšie faktory ako napríklad syntaktické odlišnosti môžu mať vplyv na porozumenie takejto komunikácie v počítačovom spracovaní.

Spomínaná komunikácia spadá medzi prirodzené jazyky. Preto sú k rozpoznaniu entít v týchto vetách použité postupy NLP. Pri návrhu bolo čerpané z kapitoly 3.

Rozpoznanie tejto komunikácie je realizované vhodným NER modelom schopným tagovania slov v sekvencii. V súčasnosti sú na tento účel používané hlavne modely založené na transformeroch (sekcia 3.3). Pre trénovanie tohto modelu je treba poskytnúť vhodne rozdelené dáta. Za týmto účelom je datová sada spracovaná a taktiež rozšírená o ďalšie vety. Trénovaných je viacero modelov a tieto sú následne porovnávané na základe testovacích sád. Z modelov je vybraný najvhodnejší pre nasadenie. Pre detekciu volacích znakov bude implementovaná služba, využívajúca vytvorený model. Detekované volacie znaky budú ďalej konvertované na ich ekvivalent zo zoznamu registrovaných znakov. Služba následne vráti výsledky s volacími znakmi v ICAO formáte.

4.2 Datová sada

Poskytnuté dáta pochádzajú z projektu ATCO2 [27]. Dáta obsahujú 4 hodiny letovej VHF komunikácie, pochádzajú z rôznych letísk v odlišných krajinách. Ku každej komunikácii existuje skupina súborov medzi ktorými je audio (`.wav`) súbor a XML súbor s tagovanými prepismi tejto komunikácie.

ICAO	Mesto	Počet súborov	Akceptované	Lokálny jazyk
LKPR	Praha	84	54	Český
LKTB	Brno	47	24	
LSGS	Sion	513	342	Nemecký, Francúzsky
LSZH	Zürich	419	272	
LSZB	Bern	310	209	
LZIB	Bratislava	88	53	Slovenský
YSSY	Sydney	656	547	Anglický
Všetky	–	2117	1501	–

Tabuľka 4.1: Popisy letísk, z ktorých pochádzajú dáta. V stĺpci „ICAO“ je uvedený kód letiska evidovaný u ICAO. Nasleduje mesto v ktorom sa dané letisko nachádza. Počet súborov je celkový počet súborov v datovej sade k príslušnému letisku. Stĺpec „akceptované“ označuje počet súborov, ktoré boli znovu skontrolované ľuďmi a akceptované ako korektne anotované (súbor `accepted.list`). Nasleduje jazyk používaný v danej krajine. Všetky z týchto súborov sú v angličtine.

V tabuľke 4.1 je zloženie datovej sady podľa letísk. Síce sú v tejto práci použité prepisy len v angličtine, jazykové faktory môžu mať vplyv na ASR systémy, z ktorých tieto texty pochádzajú preto je uvedený aj lokálny jazyk.

Každý súbor má koreňový element *data*, ktorý obsahuje segmenty komunikácie. Každý segment obsahuje prehovor 1 z komunikujúcich strán. Komunikujúce strany sú označené v každom segmente v elemente *speaker* písmenom abecedy. Komunikujúca strana je taktiež označená pri každom segmente jej názvom (napr. ATCO tower), prípadne volacím znakom. Segment taktiež obsahuje informácie o začiatočnom, koncovom čase a informácie či je prepis a tagovanie korektné.

Dôležitou časťou segmentu je textový element, ktorý obsahuje prepis s tagmi daného prehovoru. Tagy označujú príkazy, číselné hodnoty, príkazy, volacie znaky a iné. V niektorých súboroch sú slová označené jedným spoločným tagom, no časť prepisov má tagy rozdelené pre každé slovo osobitne.

> [#callsign] Delta Charlie Kilo Victor India [/#callsign] [#unnamed] tower hello QNH [/#unnamed] [#value] one zero two seven [/#value] [#command] startup approved [/#command] [#value] runway three two [/#value]

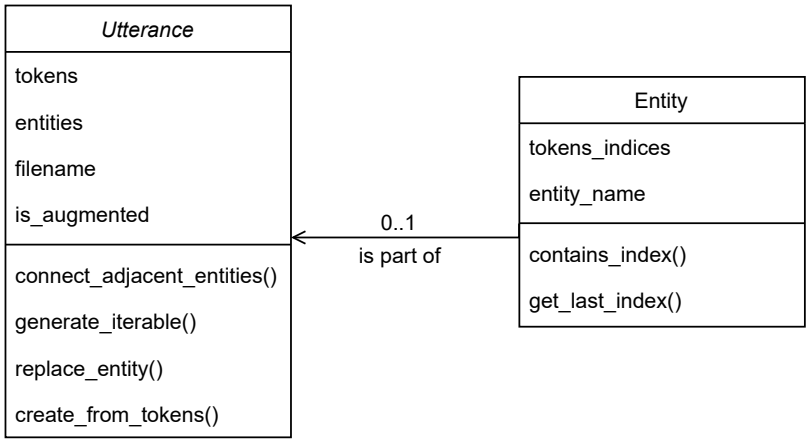
Spracovanie datovej sady

Datovú sadu treba najprv prekonvertovať do vhodného formátu. Zvolený formát musí byť kompatibilný s použitým modelom. Pre konverziu dát do požadovanej reprezentácie bol použitý jazyk Python a knižnica potrebná pre prácu s XML súbormi.

Pre nakladanie z jednotlivými vetami zo vstupných súborov boli vytvorené triedy *Utterance* pre jednotlivé vety a *Entity* pre každú pomenovanú entitu zo vstupných dát.

Trieda pre vetu obsahuje pole tokeny – slová vety a pole entít, kde každá ukladá pozíciu tejto pomenovanej entity počiatočným a koncovým indexom medzi tokenmi.

Zo vstupného súboru sú najprv vybrané neprázdne textové elementy každého zo segmentov. Keďže tieto elementy sú zložené zo slov prepisu a príslušných tagov, sú následne všetky slová a tagy rozdelené na tokeny. Program potom iteratívne prechádza tokeny a v prípade, že narazí na začiatočný tag volacieho znaku, pridelí nasledujúce tokeny za ním do tejto



Obr. 4.1: Diagram tried skriptu pre konverziu dát. Okrem potrebných parametrov obsahujú objekty aj metódy pre jednoduchšie nakladanie s nimi.

pomenovanej entity dokým nenarazí na koncový tag. Takýmto spôsobom program vytvorí objekty viet a ich entít pre každý zo vstupných súborov.

Program je navrhnutý, aby podporoval možné rozšírenie vstupných dát, tým že vyberie zoznam volacích znakov, následne vytvorí nové vety s náhodne vybranými volacími znakmi z iných viet. Takto je možné datovú sadu zväčšiť dvojnásobne. Podobný postup bol využitý v práci [26]. Pri týchto experimentoch boli použité dáta z ľudských prepisov, bez chýb zavedených ASR systémom.

Rozdelenie datovej sady

Pôvodne bolo predpokladané nasadenie knižnice SpaCy pre trénovanie modelu, takže trénovacie dáta sú následne prevedené do binárneho formátu, ktorý táto knižnica vyžaduje.

Rozdelenie vstupných dát bolo pôvodne 70, 15 a 15 % v trénovacej, validačnej a testovacej sade. V pôvodných fázach bolo trénovanie uskutočňované na dátach z rovnakých letísk ako aj testovanie a validovanie. Tento prístup bol následne zmenený a letiská Praha a Brno boli vylúčené z trénovacej sady, pre presnejšie testovanie modelu. Novým rozdelením je možné pozorovať rozdiely v detekcii volacích znakov z letísk, ktoré model pri fáze trénovania nezaznamenal:

- `train` – trénovacia sada, obsahuje 2397 záznamov z letísk Sion, Bern, Zürich, Bratislava a Sydney.
- `validation` – validačná sada použitá pri trénovaní, skladá sa z 525 sekvencií z rovnakých letísk ako trénovacia sada,
- `test1` – testovacia sada, zahŕňa 588 sekvencií z letísk na ktorých prebiehalo trénovanie, ale aj nevidených letísk,
- `test2` – druhá testovacia sada, je zložená z 274 sekvencií letísk Praha a Brno, kde sú volacie znaky nevidené.

Pri testovacej sade test2 je veľké množstvo volacích znakov od lokálne registrovaných lietadiel. Tieto majú začínajúci prefix „Oscar Kilo“, ktorý je používaný lietadlami registrovanými v Českej Republike.

4.3 Evaluácia modelu

Pre zhodnotenie výstupu rozpoznania entít z testovacej sady je potrebné vedieť výkon modelu pomocou štandardizovanej metriky.

Precision, recall a F-skóre sa používajú na meranie výkonu počítačového systému pre úlohy rozpoznávania pomenovaných entít. Pre výpočet je potreba vedieť počet tokenov správne priradených k entitám (true positive – t_p), počet tokenov, ktorým boli priradené entity aj napriek tomu, že k nim nepatria (false positive – f_p). K výpočtu je treba aj počet tokenov, ktoré boli správe nezaradené do žiadnej entity (true negative – t_n).

Precision (presnosť) meria percento pomenovaných entít, ktoré boli správne predikované systémom. Je to vypočítané ako pomer počtu správnych predikcií k celkovému počtu predikcií systému.

$$precision = \frac{t_p}{t_p + t_n} \tag{4.1}$$

Recall (pokrytie) meria percento skutočných pomenovaných entít, ktoré boli systémom identifikované.

$$recall = \frac{t_p}{t_p + f_p} \tag{4.2}$$

F - skóre (harmonický priemer) je zváženým priemerom precision a recall. Je vypočítané nasledovne:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4.3}$$

F - skóre sa pohybuje v rozmedzí od 0 do 1, pričom vyššie hodnoty znamenajú lepší výkon systému.

Je dôležité mať na pamäti, že výber metriky závisí od konkrétnych cieľov a požiadaviek systému na NER. Napríklad systém navrhnutý na identifikáciu veľkého počtu pomenovaných entít môže uprednostniť pokrytie pred presnosťou, zatiaľ čo systém navrhnutý k identifikácii malého počtu pomenovaných entít s vysokou mierou istoty môže uprednostniť presnosť pred pokrytím.

4.4 Model rozpoznania volacích znakov

Rozpoznanie pomenovaných entít je možné realizovať spôsobmi popísanými v časti 3. Za účelom detekcie volacích znakov boli v minulosti použité viaceré metódy ako napríklad Skryté Markovove modely alebo podmienené náhodné polia, ktoré sú stručne popísané v sekcii 3.2. Niektoré z nich je možné nájsť v článku [15]. Dnes často voleným prístupom je použitie transformerov schopných klasifikácie tokenov, ktoré využíva táto práca.

Detekcia modelom BERT

Jedným z prístupov bol fine-tuning predtrénovaného modelu BERT. Tento model podporuje mnohé úlohy spojené so spracovaním prirodzeného jazyka vrátane rozpoznania pomenovaných entít. Transformer BERT je bližšie popísaný v časti 3.4.

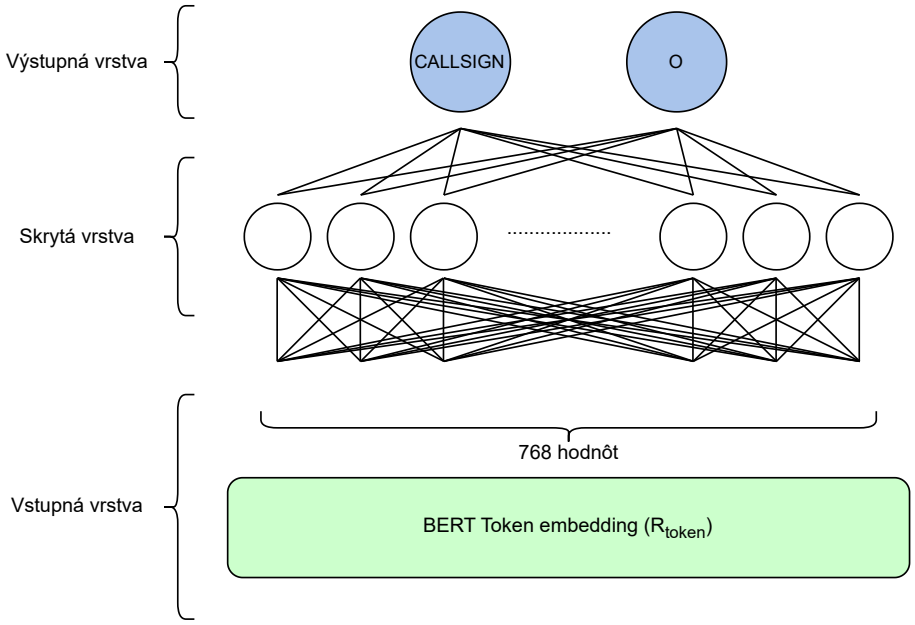
Pre použitie tohto modelu bol zvolený jazyk Python s využitím HuggingFace[1] knižníc ako Tokenizer, Datasets a Trainer. Pre toto použitie bola datová sada najprv skonvertovaná do JSON formátu, ktorý je kompatibilný z knižnicou Datasets. Dáta obsahujú vety, kde pri každej vete je pole, ktoré pozostáva z tagov priradených k jednotlivým slovám.

Vstupné vety je potrebné pred trénovaním tokenizovať. Toto zahŕňa rozdelenie vety na slová zo slovnej zásoby modelu. U slov ktoré sa v tejto slovnej zásobe nenachádzajú, dôjde k ich rozdeleniu na niekoľko „podslov“ (subwords). Podslová majú prefixy alebo sufixy symbolizujúce ich pôvodné spojenie s iným slovom. Tokenizovanie vety môže vyzerať napríklad nasledovne:

> Pôvodná veta: Hotel golf papa frequency change approve bis später.
>
> Tokenizovaná sekvencia: Hotel, golf, papa, frequency, change, approve, bis, sp##, ##ä, ##ter

Tento krok zapríčiní, že počet tokenov jednej sekvencie, je vyšší ako pôvodný počet slov vo vete. Keďže informácie o príslušných entitách priradených k slovám sú závislé od poradia týchto slov, je treba pole s týmito entitami zarovnať s novými tokenmi.

Následne sú tokenom priradené embeddingy. BERT pre vektorovú reprezentáciu slov používa kontextové WordPiece embeddingy.



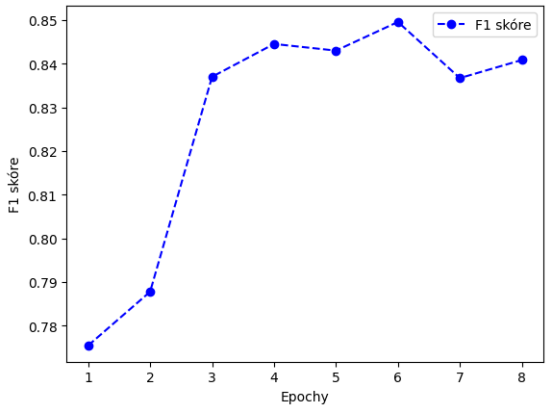
Obr. 4.2: Neurónová sieť nad modelom BERT pre klasifikáciu tokenov používaná napríklad knižnicou HuggingFace. V spodnej časti vstupuje do siete kontextová vektorová reprezentácia tokenu (embedding) z enkodéru BERT. Tá vstupuje do jednej plne prepojenej skrytej vrstvy. Výstupná vrstva je zložená z uzlu pre zaradenie aktuálneho tokenu medzi volacie znaky a uzlu „O“ ak tento token nie je volacím znakom.

Model BERT bol predtrénovaný na veľkom množstve dát, no jednalo sa o úlohy maskovaného jazykového modelovania a predikcie nasledujúcej vety. Model nebol predtrénovaný

[1]Knižnica HuggingFace – https://huggingface.co/

pre úlohy rozpoznania pomenovaných entít. Pre tento účel je možné vnímať úlohu rozpoznania pomenovaných entít ako tagovanie tokenov. Každému tokenu vety je priradený tag klasifikátorom.

Model BERT na základe vstupnej sekvencie slov vytvorí vnútornú reprezentáciu. Táto reprezantácia prejde niekoľkými vrstvami enkodérových blokov. Pre ďalšie úlohy spracovania prirodzeného jazyka je možné využiť posledný skrytý stav, tj. výstup poslednej vrstvy enkodéru. Nad týmto stavom je pre úlohy klasifikácie tokenov dopredná vrstva o veľkosti počtu tagov popísaná na obrázku 4.2.



Obr. 4.3: F1 skóre transformeru vzhľadom k počtu prebehnutých epoch. Dáta boli vyhodnotené počas trénovania na validačnej sade.

Trénovanie prebiehalo pomocou Jupyter notebooku v prostredí služby Kaggle[2], ktorá poskytuje dedikované grafické karty určené pre trénovanie rozsiahlych neurónových sietí. Pre trénovanie bolo použité GPU NVIDIA Tesla P100 s 16GB pamäte. Pri trénovaní boli použité postupy v článkoch [24, 17]. Krok učenia (learning rate) bol nastavený na 2×10^{-5}. Evaluácia je vykonávaná na konci každej epochy. Veľkosť trénovacej várky je nastavená na 16. Počet epoch je nastavený na 3, pretože u predtrénovaných modelov sa neodporúča zvyčajne väčší počet epoch, čo sa ukázalo aj na tejto sade (obr. 4.3).

Po natrénovaní modelu s konfiguráciou popísanou vyššie boli dosiahnuté výsledky na testovacích sadách popísané v tabuľke 4.2. Výsledky boli získavané na CPU v lokálnom prostredí. Je vidno, že pri prvej testovacej sade boli dosiahnuté podstatne lepšie výsledky ako pri druhej. Rozdiel medzi výsledkami je značnejší ako u modelu trénovaného knižnicou SpaCy. Z tohto je možné usúdiť, že tento model nevystihuje komplexné vzťahy medzi slovami tak účinne ako predošlý, čomu naznačuje aj nižšie F1 skóre.

Metrika	**test1**	**test2**
Precision	68,71 %	54,93 %
Recall	68,34 %	55,06 %
F1	68,24 %	54,76 %

Tabuľka 4.2: Dosiahnuté výsledky pri detekcii volacích znakov modelom BERT natrénovaným knižnicou HuggingFace. Testovacie sady *test1* a *test2* sú popísané v časti 4.2. Použité metriky sú popísané v časti 4.3

[2]Služba Kaggle – https://www.kaggle.com/

Použitie knižnice SpaCy

SpaCy je knižnica s otvoreným zdrojovým kódom určená pre spracovanie prirodzeného jazyka. Táto knižnica má moduly pre prácu s rôznymi úlohami NLP ako napríklad POS tagovaním, tvorbou derivačných stromov, kategorizáciou a NER[3]. Používa jazyk Python a Cython, ktorý je navrhnutý pre poskytnutie vyššieho výkonu v niektorých situáciách. SpaCy využíva vlastnú knižnicu strojového učenia Thinc. Poskytuje množstvo predtrénovaných modelov pre rôzne jazyky. Umožňuje prácu s predtrénovanými transformermi ako napríklad RoBERTa. Umožňuje vizualizáciu výstupných dát z klasifikácie tokenov. Ďalšou praktickou výhodou je snaha autorov knižnice o optimalizáciu. Pipeline pre tagovanie a spracovanie pomenovaných entít je niekoľko násobne rýchlejšia ako konkurenčné systémy a je optimalizovaná pre beh na procesore bez použitia grafickej karty.

Knižnica SpaCy poskytuje hotové fine-tunované modely s rozhraním pre prístup k nim. Fine-tunované modely poskytujú rozpoznanie pomenovaných entít pre zvyčajné triedy ako napríklad názvy organizácií alebo osôb. Umožňuje taktiež fine-tunovať vlastné modely.

Podmienkou trénovania z poskytnutých dát je formát v binárnych DocBin súboroch. Z tohto dôvodu bola pripravená datová sada tokenizovaná a konvertovaná do DocBin formátu. Trénovacej pipeline sú poskytnuté príklady v trénovacej a evaluačnej sade v príkladových objectoch `Doc`. Tento objekt obsahuje tokenizovaný text a tagy entít vo vete. Jeden objekt obsahuje jeden segment z komunikácie.

NER pipeline tohto modelu má stratovú funkciu optimalizovanú pre detekciu celých pomenovaných entít[4]. Táto pipeline je preto vhodná aj pre dlhšie entity, ktoré sú obsiahnuté práve aj v datasete tejto práce.

Metrika	**test1**	**test2**
Precision	72,82 %	61,09 %
Recall	86,35 %	84,38 %
F1	79,01 %	70,87 %

Tabuľka 4.3: Dosiahnuté výsledky pri detekcii volacích znakov modelom RoBERTa natrénovaným knižnicou SpaCy. Testovacie sady *test1* a *test2* sú popísané v časti 4.2. Použité metriky sú popísané v sekcii 4.3.

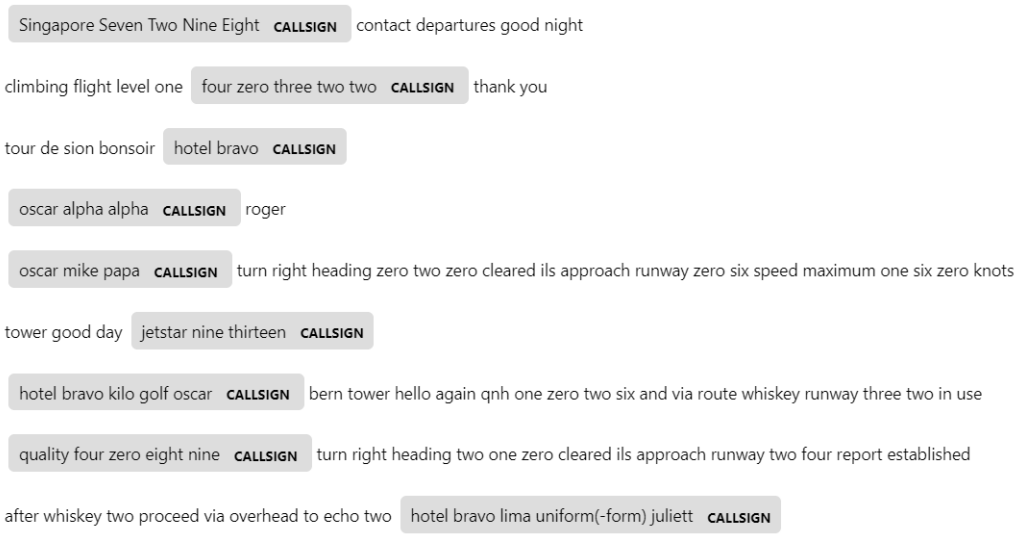
Trénovanie prebiehalo v prostredí služby Kaggle s rovnakou hardvérovou konfiguráciou ako u predošlého modelu. Metriky tejto knižnice boli následne počítané na CPU v lokálnom prostredí. Pri trénovaní bola nastavená miera dropout[5] na 0,1 a rýchlosť učenia bola nastavená na 5×10^{-5}.

Pri použití knižnice SpaCy boli metriky počítané funkciou Scorer tejto knižnice. Z úspešnosti detekcie volacích znakov tohto modelu je patrný rozdiel približne 8 percentuálnych bodov medzi použitými testovacími sadami. Prvá testovacia sada obsahuje volacie znaky, ktoré z väčšej časti pochádzajú z letísk spoločných so sadou trénovacou. Je pravdepodobné, že váhy modelu sú lepšie prispôsobené pre detekciu týchto volacích znakov. Druhá testovacia sada pochádza z letísk v ČR, ktoré neboli v trénovacej sade zahrnuté. Rozdielom dvoch testovacích sád je možné vidieť, ako dobre model zovšeobecňuje vzťahy medzi slovami v komunikácii.

[3]Modely poskytované knižnicou SpaCy – https://spacy.io/models
[4]Popis NER modelu SpaCy – https://spacy.io/api/entityrecognizer
[5]dropout – miera deaktivácie neurónov počas trénovania

Z výsledkov je v tabuľke 4.3 patrné, že pokrytie (recall) je vyššie ako presnosť (precision), a preto je pravdepodobnejšie, že budú niektoré slová falošne pozitívne vyhodnotené ako volacie znaky. U druhého modelu je rozdiel medzi hodnotami minimálny, ale naopak celková účinnosť je nižšia, ako si možno všimnúť v tabuľke 4.2.

Na obrázku 4.4 je možné vidieť výstupy pre 9 náhodne zvolených viet z testovacej sady.



Obr. 4.4: Ukážka výstupu modelu fine-tunovaného pomocou SpaCy. V ukážke figurujú náhodne zvolené vety z datovej sady. Zobrazenie tagovania je realizované knižnicou DispaCy.

4.5 Rozšírenie detekcie

V leteckej komunikácii sa vyskytujú okrem volacích znakov aj ďalšie dôležité prvky. Piloti si často s riadením letovej letovej prevádzky vymeňujú príkazy, ktoré sú doplňované hodnotami. Poskytnutá datová sada obsahuje prepisy s anotovanými príkazmi aj hodnotami. Ďalším spôsobom vylepšenia systému je rozšírenie detekcie NER modelu pre tieto triedy. Použitá datová sada má podobné pomery počtov súborov ako v predošlých sekciách. Zmenou je, že v tejto časti boli použité len súbory nachádzajúce sa na zozname `accepted.list`, ktoré boli skontrolované a nemali by obsahovať chyby v anotáciách. Trénovanie prebieha s rovnakou konfiguráciou ako v časti 4.4. Tabuľka 4.4 obsahuje výsledky detekcie viacerých tried.

Trieda	**Volací znak**		**Príkaz**		**Hodnota**	
Sada	T1	T2	T1	T2	T1	T2
Precision	90,68	91,59	60,00	52,56	57,50	54,49
Recall	86,48	85,15	67,20	65,60	59,54	60,24
F1	88,53	88,25	63,39	58,36	58,50	57,22

Tabuľka 4.4: Dosiahnuté výsledky NER modelu pri detekcii volacích znakov, príkazov a hodnôt Testovacie sady *test1* (T1) a *test2* (T2) sú popísané v časti 4.2. Použité metriky sú popísané v časti 4.3.

Kapitola 5

Systém detekcie a rozpoznania volacích znakov

Pre ďalší postup vyhodnocovania a vývoja systému je potrebné definovať 2 úlohy. Systém by mal byť schopný detekovať a rozpoznať volacie znaky z VHF komunikácie. Toto je možné dosiahnuť analýzou samotných prepisov, prípadne ďalších dostupných dát.

Pre tieto úlohy sú implementované konkurenčné systémy schopné vyhľadávať, prípadne rozpoznať volacie znaky v poskytnutom texte. Táto kapitola okrem úloh popisuje taktiež implementáciu a evaluáciu týchto systémov.

5.1 Úloha - detekcia volacieho znaku

Konvenčné NLP systémy umožňujú rozpoznanie pomenovaných entít, taktiež nazývané klasifikácia tokenov. Rozpoznanie pomenovaných entít určí ktoré tokeny (slová) patria danej entite. V tomto prípade sú entitami volacie znaky.

Úloha detekcie preto zahŕňa vyhodnotenie toho, ktoré entity patria volaciemu znaku vo vete. Často je pri vyhodnocovaní podobných úloh použité pozícií v texte (index tokenu). Pri tejto úlohe sú použité aj automatizované prepisy, u týchto prepisov často vzniká posuv indexov voči ľudským prepisom. Jednou možnosťou by bolo manuálne označovanie rozsahu volacích znakov v týchto prepisoch. Rozhodovanie ktoré tokeny patria entite by bolo náročné a nejasné, pretože automatizovane prepísané tokeny obsahujú informácie o časovaní, no ľudské nie.

Preto bola zvolená možnosť evaluácie na základe obsahu detekovaných rozsahov kde sa volací znak nachádza. V tomto prípade je potreba pripustiť určité prípadné nepresnosti v detekovaných tokenoch. Tieto tokeny nemusia byť zhodné v automatizovaných a ľudských prepisoch. Preto bolo zvolené použitie Levenštajnovej vzdialenosti [11] pre posudzovanie či detekovaný obsah odpovedá reálnemu obsahu z ľudského prepisu. Levenštajnova vzdialenosť väčšinou pracuje na úrovni znakov, no v tomto prípade bol algoritmus upravený pre meranie vzdialenosti na úrovni tokenov[1].

Detekovaný obsah je považovaný za správny pokiaľ Levenštajnova vzdialenosť tohoto obsahu a správneho obsahu je menšia alebo rovná 2. V prípade, že dĺžka správneho obsahu je menšia alebo rovná 3, je limit Levenštajnovej vzdialenosti 1.

[1] Pôvodná implementácia Levenštajnovej vzdialenosti z https://github.com/Yuvashree135/levenshtein-js bola upravená pre potreby tejto práce.

Majme napríklad volací znak „Hotel zulu papa echo“ v ľudskom prepise a k tomu príslušný automatizovaný prepis je „Mozart zulu papa echo.“ Ak systém túto sekvenciu označí ako obsah s volacím znakom, bude to vyhodnotené ako správna detekcia aj napriek mierne odlišnému obsahu.

5.2 Úloha - rozpoznanie ICAO volacieho znaku

V predošlej úlohe je vykonávaná detekcia na úrovni časti viet. Systémy pre prácu s leteckými dátami ako napríklad OpenSky Network[2] používajú štandardizovanú ICAO formu volacích znakov. Pre tento účel je potrebné aby systém rozpoznal najpravdepodobnejší volací znak obsiahnutý vo vete.

Pre tento účel je možné využiť taktiež dodatočných dát. Možnosť určiť volací znak, môže pomôcť napríklad zoznam blízkych volacích znakov. Radarové dáta boli pre účel rozpoznania volacích znakov použité v práci [2]. Takýto zoznam je možné získať napríklad z OpenSky Network. Tieto dáta sú taktiež obsiahnuté v poskytnutej datovej sade.

V druhej úlohe je teda potrebné, aby systém na základe obsahu vety prípadne ďalších pomocných dát vyhodnotil, ktorý volací znak lietadla v ICAO forme je možné priradiť k vete. Možné formáty volacích znakov podľa ICAO sú popísané v kapitole 2. Pre porovnanie, splnenie oboch úloh na ukážke vyzerá napríklad takto:

> Veta: Singapore two one two contact departures good day.
> Prvá úloha: Singapore two one two
> Druhá úloha: `SIA212`

5.3 Konverzia datovej sady

Pre tieto úlohy je potrebné aby každá vzorka testovacej sady obsahovala:

- ľudský (ground-truth) prepis so správnym tagovaním,
- prepis vytvorený ASR systémom,
- zoznam blízkych volacích znakov.

Formát `.XML` súboru s ľudskými prepismi je opísaný v časti 4.2. Z tohto súboru sú získané ľudské prepisy, informácie o časovaní segmentov a označenie komunikujúcich strán (`speaker_label`).

Pre verifikovanie boli použité prepisy z ASR systému. Tieto prepisy sú vo forme `.CNET` súboru. Tento súbor je zoznam rozpoznaných slov s ich časmi a n-best hypotézami. Na každom riadku sa nachádza najprv: názov súboru, označenie rečníka, čas začiatku slova, trvanie slova. Za týmito informáciami je medzerami oddelený zoznam rôzneho počtu dvojíc (hypotéz) vo formáte `<slovo> <pravdepodobnosť>`, počínajúc slovom s najvyššou pravdepodobnosťou. Časť súboru je možné vidieť na príklade 5.1.

[2] OpenSky Network – organizácia poskytujúca letové dáta (https://opensky-network.org/)

```
LKPR_711 A 0.12 0.27 <eps> 0.976 and 0.02 to 0.01 is 8.1e-05
LKPR_711 A 0.39 0.36 oscar 0.93 k_l_m 0.03 ascot 0.02 <eps> 0.01
LKPR_711 A 0.75 0.21 kilo 0.86 k_l_m 0.12 echo 0.07 <eps> 0.001
LKPR_711 A 0.96 0.05 <eps> 0.916 four 0.05 five 0.03 golf 0.02
LKPR_711 A 1.01 0.19 alfa 0.43 foxtrot 0.26 four 0.15 five 0.05
LKPR_711 A 1.2 0.11 <eps> 0.74 four 0.13 charlie 0.09 papa 0.06
LKPR_711 A 1.32 0.28 romeo 0.75 charlie 0.11 your 0.04 on 0.02
LKPR_711 A 1.6 0.01 <eps> 0.93 on 0.03 kilo 0.01 alfa 0.01
LKPR_711 A 1.62 0.28 five 0.851 on 0.003 <eps> 0.002 i 0.001
```

Obr. 5.1: Ukážka časti súboru s automatizovaným prepisom. Názov v prvom stĺpci súboru bol skrátený. Taktiež sú zaokrúhlené hodnoty pravdepodobností. V ukážke vidno zlomok z reálneho počtu hypotéz.

Zhruba u polovice slov počet hypotéz prevyšuje 100. Medzi hypotézami sa občas nachádza znak `<eps>`, ktorý je treba z hľadiska vyhľadávania volacích znakov ignorovať.

Pre verifikáciu bolo potrebné hľadať volacie znaky v týchto súboroch. Súbor `.INFO` obsahuje informácie o nahrávke ako napríklad čas, miesto, frekvencia a taktiež zoznam blízkych volacích znakov. Tento zoznam je na konci súboru. Každý volací znak sa nachádza na samostatnom riadku a je zložený z krátkej a dlhej (hovorenej) verzie. Jeden volací znak vyzerá nasledovne: `DLH7LP : Lufthansa Seven Lima Papa`.

Pre ďalšiu prácu a testovanie boli dáta z troch súborov konvertované do jedného JSON súboru. Tento JSON súbor obsahuje informácie o segmentoch konkrétneho súboru. Obsahuje taktiež zoznam reálnych volacích znakov v súbore. Pri každom segmente je informácia o tagovaní volacích znakov podľa jednotlivých segmentov.

V pôvodnej datovej sade boli označené volacie znaky pre slová ľudského prepisu. Prepisy z ASR nie sú segmentované, a preto bolo potrebné ich segmentovať na základe času a trvania jednotlivých slov a časovania segmentov v ľudských prepisoch.

Pre úlohy verifikácie je potrebné pri každom segmente evidovať správny volací znak v jeho ICAO formáte. Pre tento účel boli extrahované volacie znaky v tagu `speaker_label` v zdrojovom XML súbore. Z pôvodnej datovej sady boli pre ďalšie spracovanie vylúčené súbory pre ktoré platí aspoň jedna z nasledujúcich možností:

- názov vzorky sa nenachádzal na zozname akceptovaných vzoriek so správnym ľudským prepisom. Zoznam bol dodaný spolu s datovou sadou (`accepted.list`),
- zoznam blízkych volacích znakov je prázdny,
- obsiahnutý volací znak sa nenachádza na zozname blízkych volacích znakov,
- volací znak je tagovaný ako entita, no súbor neobsahuje v `speaker_label` žiaden volací znak,
- obsahuje volacie znaky výhradne s prefixom `UNK`, a teda nie je možné jednoznačne určiť volací znak bez manuálneho označovania.

5.4 Použitie enkodérových modelov

Jedným zo spôsobov pre splnenie úlohy určenia časti textu s volacím znakom je použitie modelu využívajúceho transformery. Tieto modely sú schopné detekovať rozsah volacieho

znaku. SpaCy modely boli natrénované na trénovacej a evaluačnej sade tvorenej ľudskými prepismi. Konkrétne sa jedná o NER model z časti 4.4, v ďalších častiach referovaný ako *SpaCy NER* a model SpanCategorizer[3] referovaný ako *SpaCy SC*. SpanCat je odlišný v tom, že nepoužíva tradičné označovanie sekvencie tokenov. Model SpanCat má tzv. suggester, ktorý rozdelí vetu na n-gramy podľa nastavených dĺžok a týmto je pomocou hlavy enkodéru priradená príslušnosť k určitej triede. V tomto prípade je teda trieda volací znak. Použitý SpanCat model bol natrénovaný na rovnakej datovej sade ako NER model.

Pre účel označovania časti textu s volacím znakom bola implementovaná služba vo frameworku Flask. Vstupom do tejto služby je POST request s JSON telom 5.1. V tele requestu sa nachádza text prepisu.

```
{
    "text": "Lufthansha Eight Hotel Romeo descend flight level one hundred"
}
```

Výpis 5.1: Príklad HTTP POST requestu so segmentom letovej komunikácie.

```
[
    {
        "confidence": "0.9924429",
        "startChar": 0,
        "endChar": 28,
        "span": "Lufthansha Eight Hotel Romeo",
        "tags": [ 1, 1, 1, 1, 0, 0, 0, 0, 0 ],
    }
]
```

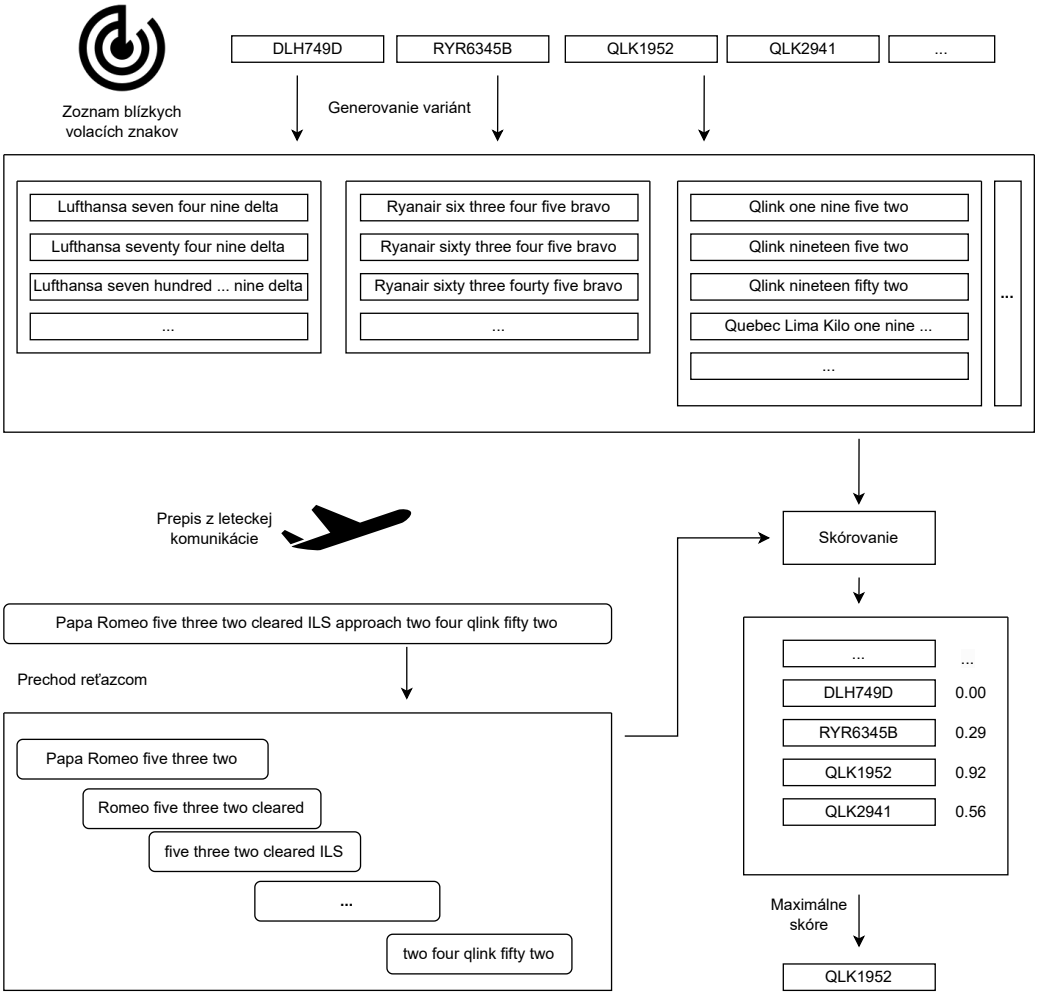
Výpis 5.2: Príklad odpovede na požiadavku. Odpoveď obsahuje istotu nálezu (pri modeli SpanCategorizer), počiatočný znak a koncový znak detekcie, časť textu obsahujúcu volací znak a pole tagov, kde 1 označuje, že token s príslušným poradím je volací znak.

V prípade, že model detekuje výskyt volacieho znaku, vráti najpravdepodobnejší detekovaný rozsah a pravdepodobnosť tohto výskytu ako JSON popísaný v 5.2.

5.5 Naivné rozpoznanie

Ďalším použitým systémom je tzv. naivné rozpoznanie. Tento spôsob využíva zoznam blízkych volacích znakov.

[3]SpaCy SpanCategorizer model je dostupný na https://spacy.io/api/spancategorizer



Obr. 5.2: Schéma modelu naivnej detekcie. Vo vrchnej časti je zoznam blízkych volacích znakov so súboru `.INFO`. Expanzia volacích znakov na varianty je zobrazená pod zoznamom. V ľavej časti figuruje prepis z letovej komunikácie, buď ľudský alebo ASR). Prepis spolu s variantami sú následne oskórované oknovým prechodom (vľavo dole a vpravo). V pravej spodnej časti je výstup s najvyšším skóre.

Systém naivného rozpoznania je implementovaný ako HTTP služba s API v JavaScriptovom frameworku express.js. Táto služba akceptuje POST requesty. Každý request musí obsahovať segment, ktorý sa skladá z textu a zoznamu blízkych volacích znakov.

Expanzia variánt

Prvým krokom je expanzia možných foriem vyslovenia volacieho znaku. V reálnej letovej komunikácii sú často vyslovované volacie znaky v rôznej forme.

Ak máme napríklad volací znak existuje viacero spôsobov akými môže byť tento volací znak vyslovený. Pri zanedbaní skrátených foriem, stále je treba uvažovať o viacerých možnostiach vyslovenia.

Napríklad volací znak `RYR8332`, je možné v plnej forme vysloviť nasledovnými spôsobmi:

- Ryanair eight three three two

- Ryanair eighty three three two
- Ryanair eight three thirty two
- Ryanair eighty three thirty two
- Ryanair eight three three two
- Romeo yankee romeo eight three three two
- ...

Zoznam blízkych volacích znakov síce obsahuje zoznam krátkych a väčšinou aj hovorených foriem volacích znakov. Problémom je, že obsahuje len jednu z týchto rôznych podôb ktoré môže tento vyslovený volací znak nadobúdať.

Z tohto dôvodu bol implementovaný modul `callsignsExpansion.js`, ktorý expanduje zvolený vybraný volací znak z jeho ICAO formy do rôznych variácií v hovorenej podobe.

Tento modul najprv prijatý volací znak rozdelí na písmennú a numerickú časť, za ktorou môže taktiež nasledovať písmenná časť. Pre volací znak `NJE473E` je rozdelenie nasledovné:

[Písmenná časť]	[Numerická časť]	[Písmenná časť]
NJE	473	E

Prvá písmenná časť vo väčšine prípadov obsahuje identifikátor letovej spoločnosti. Modul preto používa zoznam identifikátorov letových spoločností a im priradených volacích znakov [23]. V prípade, že sa jedná o takýto identifikátor, bude pri ďalšom postupe vytvárať aj variácie s príslušným volacím znakom. V každom prípade bude táto sekvencia písmen konvertovaná na slová podľa fonetickej abecedy.

Ďalším krokom je konverzia numerickej časti. Táto časť môže mať dĺžku od jedného do štyroch znakov. Modul preto vytvorí rôzne spôsoby vyslovenia podľa dĺžky a obsahu numerickej časti.

Následne je podľa fonetickej abecedy konvertovaná aj prípadná koncová písmenná časť. Všetky tri časti sú ďalej spojené a modul vracia možné variácie.

Vyhodnotenie variánt

Pre každú variáciu ohodnotenie prebieha tzv. oknovým algoritmom. Tento algoritmus je možné prirovnať k n-gramu. Dĺžka kontextu (okna) je rovná počtu tokenov danej variácie. Takto algoritmus prechádza n-tice slov z vety a pre každú pozíciu určí skóre. Pre určenie skóre bolo použité viacero spôsobov ako napríklad Levenštajnova vzdialenosť alebo Jaccardov index buď na úrovni znakov alebo tokenov. Najlepšie výsledky boli dosiahnuté pomocou algoritmu Jaro – Winkler [25] na úrovni tokenov z balíku [4].

V prípade použitia automatizovaných prepisov je pri každom slove jeho pravdepodobnosť p. Z tohto dôvodu je skóre vynásobené modifikátorom, ktorý predstavuje priemerné skóre tokenov v okne. Ak $\bar{p}$ je aritmetický priemer pravdepodobností, finálne skóre je vypočítané nasledovne:

$$Score(s_1, s_2) = JaroWinkler(s_1, s_2) * \bar{p} \tag{5.1}$$

[4] NPM Balík pre výpočet Jaro – Winkler podobnosti je dostupný na https://www.npmjs.com/package/Jaro-Winkler

Po nájdení n-tice slov s najvyšším slovom, je následne táto n-tica zmenšovaná na najmenšiu možnú dĺžku pri zachovaní skóre. Týmto je docielené presnejšie ohraničenie polohy volacieho znaku.

5.6 Pokrytie klasifikátorov

Na pripravenej datovej sade boli následne merané metriky. Pri každej úlohe bola meraná účinnosť na ľudských aj ASR prepisoch. Systémy pre realizáciu týchto úloh priraďujú vstupným segmentom najpravdepodobnejší výsledok. V podstate sa jedná o klasifikáciu, kde sa počet tried mení podľa počtu blízkych volacích znakov, ktorých počet sa medzi datovými vzorkami mení. Tieto klasifikátory pri výsledkoch uvádzajú taktiež pravdepodobnosť nálezu.

Pri testovaní klasifikátora môže byť problémom počet falošne pozitívnych nálezov. Preto je v praxi používaná minimálna hranica pravdepodobnosti nálezu. Ak bude pravdepodobnosť nálezu vyššia ako táto hranica, bude považovaný za dôveryhodný.

Výpočet detekčnej krivky

Pre určenie vhodnej hranice budú vypočítavané nasledujúce metriky:

- TPR (angl. true positive rate, pomer pozitívnych nálezov) – určuje citlivosť klasifikátora
- FPR (angl. false positive rate, pomer falošne pozitívnych nálezov) – určuje presnosť klasifikátora, nižší pomer je lepší

Počet prípadov kedy bol k segmentu správne priradený blízky volací znak je TP (true positive). Počet prípadov kedy bol vete priradený nesprávny volací znak je FP (false positive). Potom je možné TPR vypočítať nasledovne:

$$TPR = \frac{TP}{TP + FP} \tag{5.2}$$

S počtom falošne pozitívnych prípadov FP a počtom prípadov, kedy je vete správne nepriradený volací znak TN (true negative), je FPR vypočítané takto:

$$FPR = \frac{FP}{TN + FP} \tag{5.3}$$

Následne je možné pomocou ROC (angl. receiver operating characteristic) krivky určiť vhodnú hranicu detekcie. Pri nastavení hranice je možné uprednostniť jednu z metrík na úkor druhej. Určenie správnej hranice vyžaduje poznať aj pomer celkový pomer pozitívnych a negatívnych vzoriek v cieľových dátach.

Všeobecný ukazovateľ efektivity klasifikátoru je možné vyjadriť ako obsah pod ROC krivkou – AUC (area under curve). AUC je rozsah od 0 do 1, kde hodnota rovná 1 predstavuje perfektný klasifikátor. Výpočet tejto metriky je vhodné realizovať pomocou trapézoidov. Keďže hustota bodov na osiach grafu je odlišná vzhľadom k ich častiam. Napríklad v intervale $FPR \in (0; 0,5)$ môže byť iný počet bodov ako v $FPR \in (0,5; 1)$.

Výpočet detekčnej krivky v tejto práci sa odlišuje od tradičnej ROC krivky. V tomto prípade je pri nájdení správneho volacieho znaku vo vete pripočítaný true positive prípad v opačnom prípade je to false negative, prípadne false positive, ak bol detekovaný zlý volací

znak. Pri tradičnej ROC krivke by boli všetky nedetekované volacie znaky kategorizované osobitne. Ak máme teda vetu obsahujúcu volací znak `QLK153` a v zozname blízkych volacích znakov je ďalších 9 volacích znakov. V prípade že systém určí volací znak správne (teda `QLK153`), je pričítaný ako 1 true positive prípad. Pri bežnej ROC krivke by boli ostatné volacie znaky zarátané ako true negative.

WER

Pre určenie vplyvu kvality automatizovaných prepisov na efektivitu výstupov modelov, je treba zmerať ich kvalitu. Toto je väčšinou realizované výpočtom miery chyby slov (angl. word error rate, WER). Ak S je počet substitúcií (zle rozpoznaných slov), D je počet vymazaní (nedetekovaných slov), I je počet slov navyše a N je celkový počet slov v referenčnej vzorke, výpočet WER je nasledovný:

$$WER = (S + D + I)/N \tag{5.4}$$

5.7 Výsledky pokrytia

Pre vyhodnotenie pokrytia systémov bola použitá datová sada so zložením popísaným v tabuľke 5.1. Pre získanie WER prebiehajú mierne normalizácie textu. Porovnávaný je text ľudského prepisu s najlepšou hypotézou z ASR prepisu. Výpočet WER je realizovaný pomocou python balíku `jiwer` [5].

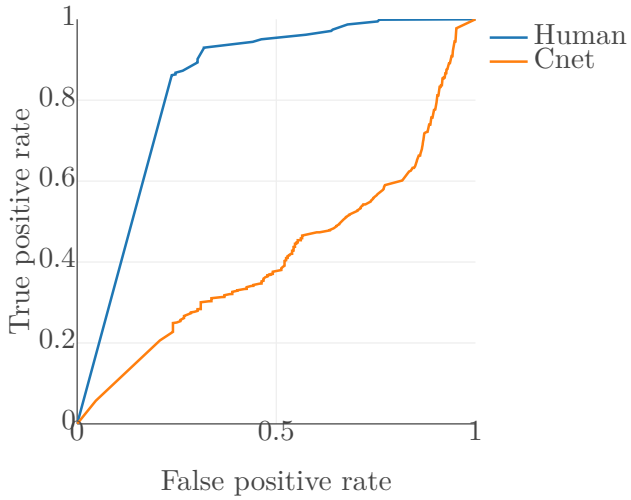
Vyhodnocovanie prebieha vzhľadom k jednotlivým segmentom. Segment má jedného rečníka.

Letisko	počet súborov	počet segmentov	WER
Všetky	605	1042	36,04
LK{PR,TB}	28	68	20,11
LSGS	53	98	30,68
LSZ{B,H}	103	190	25,79
LZIB	28	58	18,34
YSSY	393	628	43,71

Tabuľka 5.1: Prehľad datovej sady pre vyhodnotenie AUC. Jedná sa o súbory zvolené postupom popísaným v sekcii 5.3. Letiská Praha (LKPR) a Brno (LKTB) sú spojené do spoločnej kategórie. Vzorky z Letísk Bern (LSZB) a Zürich (LSZH) boli tiež združené v jednej kategórii.

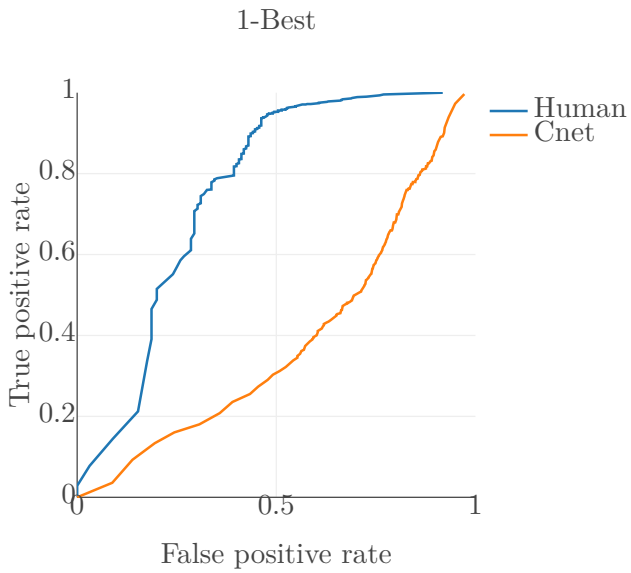
Evaluované systémy boli implementované ako REST API. Evaluačný skript preto najprv načíta `.JSON` súbory a posiela požiadavky so segmentami na testovaný systém. Po prijatí všetkých odpovedí prebieha evaluácia výsledkov. Zaznamenané odpovede sú roztriedené podľa rôznych prahov pre vyhodnotenie ROC. Následne sú vypísané štatistiky a zobrazená detekčná krivka.

[5] Balík `jiwer` pre výpočet WER – https://pypi.org/project/jiwer/



Obr. 5.3: Detekčná krivka – Úloha SPAN (5.1) – Naivný klasifikátor, všetky letiská. „Human“ označuje ľudský prepis s WER 0, „CNET“ označuje prepis z ASR systému s chybovosťou popísanou v tabuľke 5.1.

Pri vyhodnocovaní naivnej detekcie volacieho znaku bola získaná ROC krivka na obrázku 5.3. Vidno, že detekcia z ľudského prepisu je úspešnejšia ako pri ASR. Vyhodnotenie modelu trénovaného pomocou knižnice SpaCy je vidno na detekčnej krivke v obrázku 5.4.

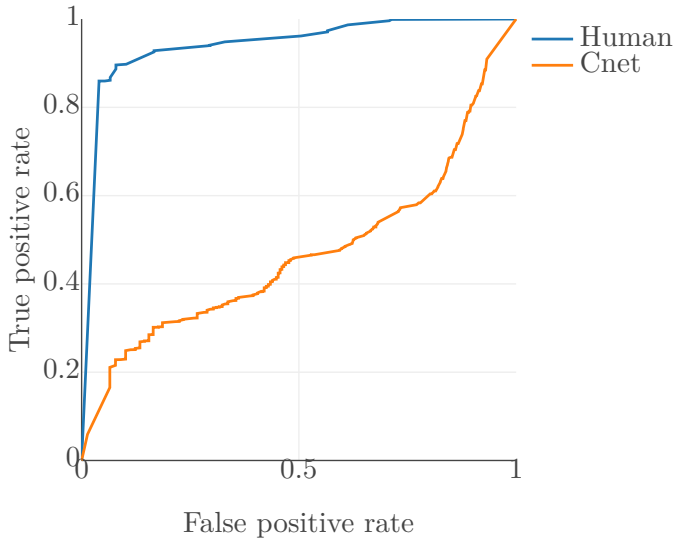


Obr. 5.4: Detekčná krivka – Úloha SPAN (5.1) – Klasifikátor SpaCy SC, všetky letiská. „Human“ označuje ľudský prepis s WER 0, „CNET“ označuje prepis z ASR systému s WER popísaným v tabuľke 5.1.

Detailné výsledky pokrytia pre úlohu detekcie volacieho znaku v texte vidno v tabuľke 5.2. Pri úlohe č. 1 vidno, že naivný systém ná celkovo vyššie pokrytie na ľudských aj automatizovaných prepisoch, úbytok pokrytia je ale pri ASR značný. Úbytok účinnosti pri ASR zhruba odpovedá chybovosti prepisov. Evaluačný skript je založený na editačnej vzdialenosti, alternatívou by bolo použitie polohy v rámci textu.

Letisko	**Naivný**		**SpaCy**		**WER**
	Human	**CNET**	**Human**	**CNET**	
Všetky	90.97	44.70	89.16	39.97	36.04
LK{PR,TB}	92.68	68.79	55.92	57.55	20.11
LSGS	89.14	68.34	81.50	68.04	32.68
LSZ{H,B}	84.65	66.93	92.86	74.19	29.79
LZIB	63.67	66.69	55.57	65.39	18.34
YSSY	68.29	24.34	55.25	11.43	46.41

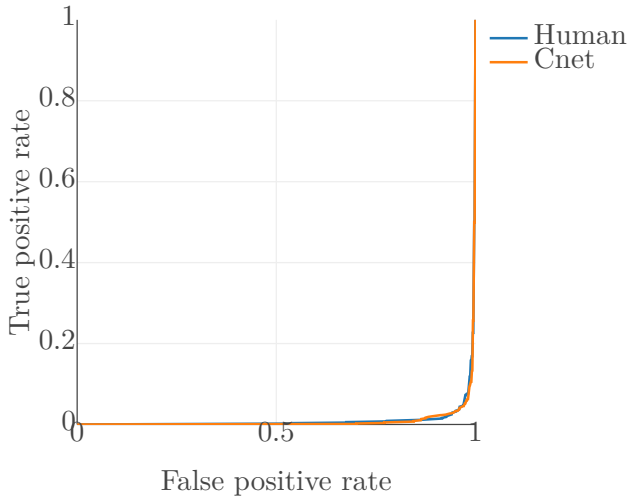
Tabuľka 5.2: AUC – Úloha SPAN. Stĺpec „Human“ označuje ľudský prepis s WER 0, stĺpec „CNET“ označuje prepis z ASR systému s WER popísaným v tabuľke 5.1.



Obr. 5.5: Detekčná krivka – Úloha ICAO (5.2) – Naivný klasifikátor, všetky letiská. „Human“ označuje ľudský prepis s WER 0, „CNET“ označuje prepis z ASR systému s WER popísaným v tabuľke 5.1.

Ďalej bolo vyhodnotené rozpoznanie volacieho znaku v podobe, aký udáva ICAO. Takže pri tejto úlohe je odpoveď systému validovaná s označením rečníka vo vzorkách. Na tejto úlohe vidno mierne vyššie pokrytie naivného systému pri ľudských prepisoch. Výsledky možno vidieť na obrázku 5.5. Pre porovnanie je vykreslená krivka aj pre náhodný klasifiká-

tor na obrázku 5.6. Náhodný klasifikátor priraďuje náhodné skóre všetkým blízkym volacím znakom a následne odošle v odpovedi volací znak s najvyššou pravdepodobnosťou.



Obr. 5.6: Detekčná krivka náhodného klasifikátora pre určenie ICAO volacieho znaku obsiahnutého vo vete.

Letisko	**Jaro – Winkler**		**Levenštajn**		**WER**
	Human	**CNET**	**Human**	**CNET**	
Všetky	95,17	47,71	83,24	35,25	36.04
LK{PR,TB}	97,61	72,06	97,61	69,95	20.11
LSGS	92,87	73,19	87,98	67,78	32.68
LSZ{H,B}	87,62	71,16	79,10	48,68	29.79
LZIB	91,52	78,54	70,95	62,47	18.34
YSSY	98,61	25,71	71,52	18,31	46.41

Tabuľka 5.3: AUC – Úloha ICAO – Naivný klasifikátor. Tabuľka znázorňuje rozdiel v pokrytí pri použití dvoch algoritmov pre skórovanie variánt hovorených volacích znakov. Stĺpec „Human“ označuje ľudský prepis s WER 0, stĺpec „CNET“ označuje prepis z ASR systému s WER popísaným v tabuľke 5.1.

Pri úlohe detekcie ICAO je evaluovaný systém naivnej detekcie. Výsledky možno vidieť tabuľke 5.3. Finálny systém teda používa Jaro – Winkler skóre. Treba poznamenať, že v datovej sade sa občas nachádzajú volacie znaky, ktoré nemajú ekvivalent v ICAO formáte. Jedná sa najmä o prípady, kedy je použitá nejaká lokálna varianta. Opäť je možné vidieť rozdiely v účinnosti medzi letiskami pri ASR prepisoch.

5.8 Porovnanie modelov

V tejto časti boli vyššie zvolené klasifikátory porovnávané štandardnými metrikami pre rozpoznanie pomenovaných entít. Pre každú vzorku boli vypočítavané precision, recall a F1 skóre, ktorých výpočet je popísaný v sekcii 4.3. Metriky precision a recall boli vypočítané npm balíkom *fscore*[6]. Tieto sú následne spriemerované a je z nich vypočítané F-skóre. Štatistiky boli vypočítavané pre každý segment vzhľadom na jednotlivé tokeny entít. Toto zabezpečí, že aj pri ASR prepisoch budú dáta porovnateľné.

Porovnanie s generatívnym modelom

Pre účely rozpoznania pomenovaných entít sú používané enkodérové architektúry transformerov ako napríklad BERT alebo RoBERTa. Tieto druhy transformerov sú brané ako najmodernejšie, SOTA (angl. state-of-the-art) pre mnoho úloh NLP. V priebehu tvorby tejto práce boli spoločnosťou OpenAI predstavené nové autoregresívne modely[7]. Tieto generatívne jazykové modely neboli doteraz príliš využívané na klasifikačné úlohy, ktoré boli väčšinou vykonávané enkodérmi. Modely založené na enkodéroch boli síce od ich predstavenia vylepšované, no generatívne modely boli doposiaľ natrénované na podstatne väčšom množstve dát.

Tradičné enkodérové modely je potrebné dodatočne natrénovať pre konkrétnu úlohu s pridaním dodatočnej vrstvy. Generatívne modely založené na architektúre dekodéru 3.3 vykazujú veľmi dobré porozumenie jazyka. Tieto modely umožňujú rýchle naučenie sa úlohy.

Modelom ako GPT - 3.5 alebo GPT - 4 stačí často uviesť požiadavku v prirodzenom jazyku a tieto modely vygenerujú na jeho základe požadovaný obsah. Modely vychádzajú z architektúry predstavenej v článku [16]. Výsledky týchto modelov je možné vylepšovať napríklad one-shot alebo few-shot scenárom [4]. Pri týchto scenároch je modelu poskytnutý príklad požiadavku a očakávaného výstupu. Model tieto príklady spracuje ako kontext. Plnenie úloh týmto spôsobom je teda realizované iným spôsobom ako napr. pri modeli BERT. Netreba preto trénovať model, no stačí zobrať vhodne zvolený formát výstupu modelu ako výsledok.

Tento spôsob použitia generatívnych modelov sa ukázal byť v niektorých prípadoch lepší ako použitie stávajúcich SOTA modelov s fine-tuningom, prípadne aj predtrénovaním na korpuse z cieľovej domény [13].

Pre tento experiment bolo použité API spoločnosti OpenAI v jazyku Python. API poskytuje viacero systémov, medzi nimi sú *Completion* – pre doplnenie textu a *ChatCompletion* – pre doplnenie konverzácie. Zvolený bol systém *ChatCompletion* s modelom `GPT-3.5-turbo`, ktorého cena je vzhľadom na token desatinná oproti modelom z druhého systému. Pre úlohu rozpoznania volacieho znaku (úloha č. 2) bola modelu poskytnutá nasledujúca veta v časti `system`:

> You detect airplane callsigns. I provide you prompt only containing the sentence, you only ouput one callsign no more words. In case of no callsign you ouput 'None'

Za systémovým popisom nasledujú dva príklady. Príklady sa skladajú z užívateľského vstupu, teda vety z letovej komunikácie a správneho výstupu, takže slov tvoriacich volací

[6] Npm balík *fscore* – https://www.npmjs.com/package/fscore

[7] Prehľad modelov spoločnosti OpenAI je dostupný na https://platform.openai.com/docs/models/overview

znak. Ako posledná je pridaná veta, v ktorej treba detekovať volacie znaky. Príklad vety je takýto:

> Turkish six foxtrot lima hello again taxiway juliett cross runway eight contact apron one two one decimal seven five five good day.

Po odoslaní požiadavku API vráti odpoveď znázornenú v príklade 5.3 a tá je zaznamenaná evaluačným skriptom.

```
{
    "choices": [
        {
          "finish_reason": "stop",
          "index": 0,
          "message": {
            "content": "turkish six foxtrot lima",
            "role": "assistant"
          }
        }
    ],
    "created": 1683183686,
    "id": "chatcmpl-7CNFikRmbJgMddqi0Jmc3f0TMjuxh",
    "model": "gpt-3.5-turbo-0301",
    "object": "chat.completion",
    "usage": {
        "completion_tokens": 8,
        "prompt_tokens": 146,
        "total_tokens": 154
    }
}
```

Výpis 5.3: Príklad odpovede API spoločnosti OpenAI pre *ChatCompletion*. Dôležitý je parameter „content“ v objekte „message“. API tiež vráti počet použitých tokenov, od čoho sa odvíja cena.

5.9 Dosiahnuté výsledky

V tejto časti bola použitá menšia testovacia sada popísaná v tabuľke 5.4. Pochádza z neviedených vzoriek pri trénovaní a každá vzorka bola vybratá, aby splňovala požiadavky z časti 5.3.

Porovnanie systémov na úlohe detekcie volacieho znaku je v tabuľke 5.5. Trénovaný model vedie na letiskách Sion, Zürich a Sydney, zatiaľ čo naivný systém vedie na prepisoch z Prahy a Brna. Keďže enkodérový model sa pri trénovaní nedostal k prepisom z týchto letísk, je tu účinnosť nižšia. Veľká časť trénovacích dát pochádza z letísk Sydney a účinnosť je tu relatívne vysoká. Taktiež je možné vidieť rozdiel v účinnosti na automatizovaných prepisoch medzi letiskami.

Letisko	**počet súborov**	**počet segmentov**	**WER**
Všetky	61	120	27,31
LK{PR,TB}	27	65	20,76
LS{GS,ZH}	11	18	27,31
LZIB	2	3	0.02
YSSY	21	34	42,05

Tabuľka 5.4: Prehľad datovej sady pre porovnanie modelov. Letiská Praha (LKPR) a Brno (LKTB) sú kvôli počtom spojené do jednej kategórie. Taktiež sú spojené vzorky z letísk Zürich (LSZH) a Sion (LSGS). Letisko Bratislava (LZIB) nie je obsiahnuté v osobitnej kategórii pri ďalších testoch, ale nachádza sa medzi všetkými vzorkami.

Letisko	Prepis	Naivný	SpaCy SC	SpaCy NER	GPT - 3.5
Všetky	Ľudský	93,3	**93,8**	**93,8**	89,8
	CNET	65,6	**65,8**	64,7	53,4
LK{PR,TB}	Ľudský	**93,5**	90,6	90,5	84,9
	CNET	**73,6**	70,6	70,7	59,9
LS{GS,ZH}	Ľudský	80,5	93,3	**99,9**	93,3
	CNET	67,2	**70,0**	68,5	57,8
YSSY	Ľudský	98,6	**99,6**	96,7	93,7
	CNET	46,9	**51,1**	48,2	35,3

Tabuľka 5.5: Porovnanie F-skóre modelov použitých pre detekciu volacích znakov vo vetách podľa letísk (Úloha 1). „CNET“ predstavuje 1. hypotézu z automatizovaných prepisov. Datová sada je popísaná v tabuľke 5.4.

Generatívny model mal k dispozícií len 2 príklady z dát, aj napriek tomu dokázal rozpoznať veľkú časť dát. Trénované modely však majú vyššiu celkovú účinnosť na oboch druhoch prepisov. Prompt engineering je dôležitá súčasť klasifikačných úloh pri použití takýchto modelov a môže mať vplyv na dosiahnuté výsledky. Tieto výsledky mohli by byť do istej miery vylepšené väčším množstvom príkladov. Treba ale poznamenať, že konkrétny autoregresívny model je v tomto prípade podstatne drahší na použitie. Spoločnosť OpenAI v čase písania tejto práce poskytuje len API a váhy modelu GPT - 3.5 nie sú zverejnené. Cena výstupov tohto modelu je viazaná na počet tokenov požiadavku a odpovede. Každý takýto požiadavok je zložený z predošlého kontextu tj. popisu úlohy, príkladov a prepisu. Natrénovaný enkodérový model je naproti tomu možné použiť v lokálnom prostredí buď na GPU alebo CPU.

Pri úlohe rozpoznania ICAO volacieho znaku bola použitá naivná detekcia samotná a potom spolu s NER prípadne SpanCat systémom. V tomto prípade naivný systém odošle požiadavku na jeden z týchto systémov pre upresnenie časti textu s volacím znakom. Z tejto časti následne vyhodnotí najpravdepodobnejší volací znak zo zoznamu. Pri takomto použití nebol zaznamenaný žiaden zisk skóre. Pri porovnaní na úlohe rozpoznania ICAO volacieho znaku v tabuľke 5.6 je možné vidieť, že samotný naivný model si vedie rovnako dobre ako pri spojení s NER, prípadne SpanCat modelom. Tento model si pri dátach s 0 WER vedie lepšie ako baseline. No pri zašumených ASR dátach vedie baseline model. Naivný systém je ale možné použiť iba za predpokladu, že sú dostupné radarové dáta o blízkych lietadlách.

Letisko	Prepis	Naivný	NER, SC	Baseline
Všetky	Ľudský	**94,8**	**94,8**	89,4
	CNET	65,0	65,0	**78,4**

Tabuľka 5.6: Porovnanie presnosti modelov použitých pre rozpoznanie ICAO volacích znakov vo vetách podľa letísk (Úloha 2). „CNET“ predstavuje 1. hypotézu z automatizovaných prepisov. „NER, SC“ predstavuje spojenie naivnej detekcie s každým z týchto modelov. Baseline údaje sú prevzaté z práce [2] (Tabuľka 2). Modely použité v tejto práci však boli trénované a evaluované na odlišnej datovej sade. Pri hodnote automatizovaných prepisov z Baseline bola prevzatá hodnota zo stĺpca s WER 28,4 čo je najpodobnejšie testovacej sade tejto práce (tabuľka 5.4).

5.10 Perspektívy a možnosti rozvoja práce

Ďalší postup pre zdokonalenie existujúcich systémov by mohol spočívať v lepšom natrénovaní modelov. Konkrétne vylepšenie na ASR prepisoch by mohlo byť dosiahnuté trénovaním NER modelu na dátach s chybovými prepismi. Pre tento účel by bolo potrebné označiť časové rozsahy volacích znakov v prepisoch. Toto by umožnilo aj zmenu evaluačných metrík. Prispieť k presnosti by mohlo aj poskytnutie n-najlepších hypotéz pre každé slovo z prepisu transformerovému modelu.

Jednou z možností je napríklad zmena systému pre rozpoznanie pomenovaných entít, tak aby fungoval ako model s výstupom volacieho znaku v ICAO formáte bez nutnosti ďalšieho systému. Toto by vyžadovalo zakomponovanie zoznamu blízkych volacích znakov ako kontext modelu podobne ako v práci [2].

Použitie autoregresívnych jazykových modelov pre tieto účely by mohlo zlepšiť výsledky vďaka schopnostiam porozumenia jazyka týmito modelmi. Pri nových autoregresívnych modeloch je úloha zakomponovania zoznamu volacích znakov z blízkych lietadiel jednoduchá, no zatiaľ pomerne drahá. V prípade, že tieto veľké jazykové modely budú zverejnené, bolo by zaujímavé ich fine-tunovanie pre tieto úlohy.

Kapitola 6

Záver

Cieľom bolo vytvoriť a porovnať systém pre detekciu a rozpoznávanie volacích znakov v rádiovej VHF komunikácii pomocou poskytnutých dát. Tento systém sa podarilo realizovať na základe znalostí a postupov popísaných vyššie.

Práca sa v úvodných častiach zaoberá princípmi komunikácie medzi pilotmi a riadením letovej prevádzky. Vychádza zo štandardizovaných postupov civilného letectva. Poznatky fungovania komunikácie v reálnom prostredí vychádzajú tiež z poskytnutých záznamov leteckej komunikácie. Pri popise oblasti letectva je vysvetlené rozličné použitie volacích znakov lietadiel.

Taktiež je priblížená oblasť spracovania prirodzeného jazyka a rozpoznania pomenovaných entít. V tejto kapitole sú priblížené postupy pre počítačové porozumenie jazyka. Práca sa venuje fungovaniu veľkých jazykových modelov založených na architektúre transformerov. Popísané sú spôsoby akými je možné použiť jazykové modely pre detekciu pomenovaných entít.

V nasledujúcich kapitolách je navrhnutý model detekcie volacích znakov. Čitateľ je oboznámený s jednotlivými časťami systému. Poskytnutá datová sada je predstavená spolu so spôsobom jej spracovania pre vykonávanie ďalších úloh. Následne sú popísané spôsoby implementácie modelov pre detekciu volacích znakov.

Pre vytvorenie systému boli celkovo použité štyri hodiny textových prepisov zo záznamov letovej komunikácie. Vytvorené modely boli ďalej evaluované na základe testovacích sád. Práca popisuje výsledky implementovaných modelov.

V ďalšej časti je popísaná potreba detekcie a rozpoznania volacích znakov v ľudských ako aj automatizovaných prepisoch. Hlavným výsledkom práce je implementácia systémov umožňujúcich detekciu a rozpoznanie volacích znakov. Tieto systémy sú ďalej vzájomne porovnané spolu s alternatívnou technológiou.

V práci bolo zistené, že dostupnosť trénovacích dát pre konkrétne letisko umožňuje zvýšiť detekciu volacích znakov veľkými jazykovými modelmi oproti naivnému systému. Taktiež bolo zistené, že pre tieto úlohy je možné použiť autoregresívne modely, no zatiaľ nedosahujú lepšie výsledky ako modely trénované pre tieto účely. Systém implementovaný v tejto práci dosahuje 94,8 % úspešnosť detekcie a rozpoznania volacích znakov pri prepisoch bez zavedenej chyby. V automatizovaných prepisoch s chybovosťou 27% dosahuje systém úspešnosť 65 %.

Literatúra

[1] *Annex 10 to the Convention on International Civil Aviation*. 6. vyd. ICAO, 2001. Dostupné z: https://www.icao.int/Meetings/anconf12/Document%20Archive/AN10_V2_cons%5B1%5D.pdf.

[2] Blatt, A., Kocour, M., Veselý, K., Szőke, I. a Klakow, D. Call-Sign Recognition and Understanding for Noisy Air-Traffic Transcripts Using Surveillance Information. In: *Proceedings of ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE Signal Processing Society, 2022, s. 8357–8361. DOI: 10.1109/ICASSP43922.2022.9746301. ISBN 978-1-6654-0540-9. Dostupné z: https://www.fit.vut.cz/research/publication/12789.

[3] Brochier, R., Guille, A. a Velcin, J. Global Vectors for Node Representations. *CoRR*. 2019, abs/1902.11004. Dostupné z: http://arxiv.org/abs/1902.11004.

[4] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D. et al. Language Models are Few-Shot Learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. a Lin, H., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020, sv. 33, s. 1877–1901. Dostupné z: https://arxiv.org/abs/2005.14165.

[5] CAA. *Radiotelephony Manual*. 23. vyd. Pristupované 8.1.2023. Dostupné z: https://publicapps.caa.co.uk/docs/33/CAP413%20E23%20A1%2026Nov2020.pdf.

[6] Devlin, J., Chang, M.-W., Lee, K. a Toutanova, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv, 2018. DOI: 10.48550/ARXIV.1810.04805. Dostupné z: https://arxiv.org/abs/1810.04805.

[7] ICAO. *Manual of Radiotelephony*. 4. vyd. International Civil Aviation Organization. ISBN 92-9194-996-5.

[8] Jurafsky, D. *Speech and language processing*. 3. vyd. návrh. Upper Saddle River: [b.n.]. Pristupované 2. 1. 2023. Dostupné z: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf.

[9] Jurafsky, D. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. 2. vyd. Upper Saddle River: Prentice Hall, 2009. Prentice Hall series in artificial intelligence. ISBN 0-13-504196-1.

[10] Lafferty, J. D., McCallum, A. a Pereira, F. C. N. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *Proceedings of*

the Eighteenth International Conference on Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, s. 282–289. ICML '01. ISBN 1-55860-778-1. Dostupné z: http://dl.acm.org/citation.cfm?id=645530.655813.

[11] Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*. 1965, zv. 10, s. 707–710.

[12] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M. et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv, 2019. DOI: 10.48550/ARXIV.1907.11692. Dostupné z: https://arxiv.org/abs/1907.11692.

[13] Liu, Z., Yu, X., Zhang, L., Wu, Z., Cao, C. et al. *DeID-GPT: Zero-shot Medical Text De-Identification by GPT-4*. 2023. Dostupné z: https://arxiv.org/abs/2303.11032.

[14] Mikolov, T., Chen, K., Corrado, G. a Dean, J. *Efficient Estimation of Word Representations in Vector Space*. arXiv, 2013. DOI: 10.48550/ARXIV.1301.3781. Dostupné z: https://arxiv.org/abs/1301.3781.

[15] Pellegrini, T., Farinas, J., Delpech, E. a Lancelot, F. The Airbus Air Traffic Control speech recognition 2018 challenge: towards ATC automatic transcription and call sign detection. *CoRR*. 2018, abs/1810.12614. Dostupné z: http://arxiv.org/abs/1810.12614.

[16] Radford, A., Narasimhan, K., Salimans, T. a Sutskever, I. Improving Language Understanding by Generative Pre-Training. *OpenAI Technical Report*. 2018. Dostupné z: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[17] redakcia. *Named Entity Recognition with BERT in PyTorch — towardsdatascience.com* [https://huggingface.co/learn/nlp-course/chapter7/2]. HuggingFace. Pristupované 1. 5. 2023.

[18] redakcia. *Spoločnosť SpaceX získa od investorov v novom kole financovania 750 miliónov dolárov*. Denník N, Jan 2023. Dostupné z: https://e.dennikn.sk/minuta/3175927.

[19] Sanh, V., Debut, L., Chaumond, J. a Wolf, T. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. Dostupné z: https://arxiv.org/abs/1910.01108.

[20] Tiewtrakul, T. a Fletcher, S. The challenge of regional accents for aviation English language proficiency standards: A study of difficulties in understanding in air traffic control-pilot communications. *Ergonomics*. Február 2010, zv. 53, s. 229–39. DOI: 10.1080/00140130903470033.

[21] Tung.M.Phung. *A review of pre-trained language models: From Bert, Roberta, to electra, deberta, BigBird, and more* [https://tungmphung.com/a-review-of-pre-trained-language-models-from-bert-roberta-to-electra-deberta-bigbird-and-more/]. Feb 2023. Pristupované 21. 4. 2023.

[22] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. *Attention Is All You Need*. arXiv, 2017. DOI: 10.48550/ARXIV.1706.03762. Dostupné z: https://arxiv.org/abs/1706.03762.

[23] Wikipedia. *List of airline codes — Wikipedia, The Free Encyclopedia* [http://en.wikipedia.org/w/index.php?title=List%20of%20airline%20codes&oldid=1125725897]. 2023. Pristupované 10. 1. 2023.

[24] Winastwan, R. *Named Entity Recognition with BERT in PyTorch — towardsdatascience.com* [https://towardsdatascience.com/named-entity-recognition-with-bert-in-pytorch-a454405e0b6a]. 2022. Pristupované 30. 4. 2023.

[25] Winkler, W. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research Methods*. Január 1990.

[26] Zuluaga Gomez, J., Sarfjoo, S. S., Prasad, A., Nigmatulina, I., Motlicek, P. et al. *BERTraffic: BERT-based Joint Speaker Role and Speaker Change Detection for Air Traffic Control Communications*. arXiv, 2021. DOI: 10.48550/ARXIV.2110.05781. Dostupné z: https://arxiv.org/abs/2110.05781.

[27] Zuluaga Gomez, J., Veselý, K., Szöke, I., Motlicek, P., Kocour, M. et al. *ATCO2 corpus: A Large-Scale Dataset for Research on Automatic Speech Recognition and Natural Language Understanding of Air Traffic Control Communications*. arXiv, 2022. DOI: 10.48550/ARXIV.2211.04054. Dostupné z: https://arxiv.org/abs/2211.04054.

Príloha A

Obsah priloženého pamäťového média

```
/
├── Encoders/ ..... API pre detekciu volacích znakov
│   ├── results/ ..... Natrénované modely použité v API
│   │   ├── n1/ ..... Model trénovaný SpaCy NER pre volacie znaky
│   │   ├── nc1/ ..... Model detekujúci aj príkazy a hodnoty
│   │   ├── s2/ ..... SpanCategorizer model
│   │   ├── m2/ ..... Pytorch NER model
│   │   └── ..
│   ├── ner.py ..... API NER modelu
│   └── spancat.py ..... API SpanCat modelu
├── Naive/ ..... Systém naivnej detekcie
│   ├── index.js ..... Indexový súbor API
│   ├── callsignsService.js ..... Implementácia naivnej detekcie
│   ├── callsignsExpansion.js ..... Generovanie variánt volacích znakov
│   └── ..
├── NER/ ..... Súbory potrebné k trénovaniu modelov
│   ├── data-conversion/ ..... Skripty pre konverziu XML súborov
│   ├── spacy_training.ipynb ..... Trénovanie SpaCy NER a SpanCat
│   ├── bert_training.ipynb ..... Trénovanie modelu BERT
│   └── ..
├── OpenAI/ ..... API pre detekciu pomocou GPT-3.5
├── Testers/ ..... Evaluačné skripty
├── plagat.pdf ..... Plagát práce
├── text.pdf ..... Textová časť
└── video.mp4 ..... Demonštračné video
```

Príloha B

Plagát

Detekce a rozpoznání volacích znaků ve VHF komunikaci

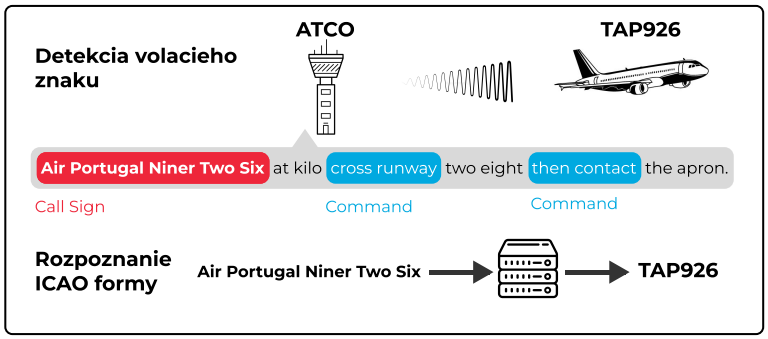
Juraj Dedič **Vedúci práce: Ing. Igor Szőke, Ph.D.**

Motivácia

VHF komunikácia poskytuje spôsob akým komunikujú **piloti** a riadenie letovej prevádzky – **ATCO**. Informácie a inštrukcie prenášané v tejto komunikácií sú veľmi dôležité pre bezpečnú a efektívnu prevádzku lietadiel. **Volacie znaky** poskytujú jednoznačnú identifikáciu lietadiel.

Letecká komunikácia

V práci boli použité textové **prepisy letovej komunikácie**. Použité boli ľudské a automatizované prepisy z ASR systému. Práca implementuje metódy **detekcie** a **rozpoznania volacích znakov lietadiel** vo vetách.



Systém detekcie a rozpoznania

Detekcia volacích znakov je realizovaná trénovanými **veľkými jazykovými modelmi**. Rozpoznanie ICAO formy znaku je realizované pomocou poskytnutých radarových dát.

Výsledky

Úspešnosť detekcie volacích znakov záleží aj od kvality textových prepisov. Pri bezchybných prepisoch dokáže systém správne detekovať a rozpoznať 95 % volacích volacích znakov. Pri prepisoch s chybovosťou 27 % dokáže systém rozpoznať 65 % volacích znakov.