Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Automatické rozpoznávání dialogových aktů v tištěných textech

Plzeň 2021 Matěj Zeman

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Matěj ZEMAN**
Osobní číslo:	**A18B0360P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Automatické rozpoznávání dialogových aktů v tištěných textech**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s dodanými datovými kolekcemi pro automatické rozpoznávání dialogových aktů (DA) v textové podobě.
2. Navrhněte a implementujte nástroj, který z dodaných textových kolekcí vytvoří analogicky anotované datové sady složené z množiny obrazových dokumentů.
3. Seznamte se se systémem optického rozpoznávání znaků (OCR) Tesseract.
4. Prostudujte relevantní metody automatického rozpoznávání dialogových aktů založené na neuronových sítích.
5. Na základě předchozí studie navrhněte a implementujte prototyp systému pro automatické rozpoznávání dialogových aktů z obrazových dokumentů. U vytvořeného prototypu použijte k převodu obrazu na text systém Tesseract.
6. Funkčnost prototypu otestujte na vytvořených datových kolekcích.
7. Výsledky diskutujte a navrhněte další možná rozšíření.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Doc. Ing. Pavel Král, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **5. října 2020**
Termín odevzdání bakalářské práce: **6. května 2021**

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 26. října 2020

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 10. května 2021

Matěj Zeman

Abstract

Modelling spontaneous dialogues is important, especially for their interpretation and understanding. Their accurate modelling is still an open problem, however, it is possible to identify some of their characteristics, such as dialogue acts.

Dialogue act recognition is usually realized from the audio signal or its text transcription. However, nobody other has analysed the dialogues in printed documents. The aim of this work is therefore to design and implement a prototype system for automatic recognition of dialog acts from documents in the form of raster images. Corpus in the form of raster images was automatically generated from the English part of the Verbmobil corpus. The experiments with automatic recognition of dialogue acts were then performed on the resulting data set.

The Tesseract system was used to convert images into text and two topologies of convolutional neural networks and one LSTM network were used to recognize dialog acts.

The experiments have shown that it is possible to recognize dialogue acts with very good accuracy from dialogues in the form of images.

Abstrakt

Modelování spontánních dialogů je důležité především pro jejich interpretaci a porozumění. Jejich přesné modelování je stále otevřeným problémem, je ale možné určit některé charakteristické znaky, kterými se dialogy vyznačují, jako jsou například dialogové akty.

Rozpoznávání dialogových aktů se zpravidla realizuje z audio signálu, případně z jeho textového přepisu. Zatím se ale nikdo nezabýval analýzou dialogu v tištěných dokumentech. Cílem práce je proto navrhnout a implementovat prototyp systému pro automatické rozpoznávání dialogových aktů z dokumentů v podobě rastrových obrázků. Z anglické části korpusu Verbmobil byl automaticky vygenerován analogický korpus v podobě rastrových obrázků. Na takto vzniklé datové sadě pak byly provedeny experimenty automatického rozpoznávání dialogových aktů.

Pro převod obrázků na text byl použit systém Tesseract a pro rozpoznávání dialogových aktů byly použity dvě topologie konvolučních neuronových sítí a síť typu LSTM. Experimenty ukázaly, že je možné z dialogů v podobě obrázků rozpoznávat dialogové akty s velmi dobrou přesností.

Poděkování

Rád bych poděkoval doc. Ing. Pavlu Královi, Ph.D za jeho trpělivost, ochotu a hlavně za obrovské množství času, který mi věnoval po celou dobu vypracování bakalářské práce.

Obsah

1 Úvod

Modelování spontánních dialogů je velmi důležité především pro jejich interpretaci a porozumění. Přesné modelování spontánních dialogů je stále otevřeným problémem, ale je možno určit charakteristické znaky, kterými se dialogy vyznačují. Dialogový akt je jednou z těchto charakteristik, která je již blíže určena a pomáhá porozumět významu určité části dialogu. Využívá se například v hlasových asistentech a dialogových systémech nebo při rozpoznávání mluvené řeči.

Rozpoznávání dialogových aktů je velmi důležitým krokem v dialogových systémech, neboť odhaluje účel jednotlivých vět. Například účelem tázací věty „Je dnes pondělí?“ je získat nějakou informaci.

Automatické rozpoznávání dialogových aktů se v současné době realizuje ze zvukového signálu, případně z jeho textového přepisu. Zatím se ale nikdo nezabýval analýzou dialogu v tištěných dokumentech.

Cílem bakalářské práce je navrhnout a implementovat prototyp systému pro automatické rozpoznávání dialogových aktů z dokumentů v podobě rastrových obrázků. Ze vstupních dat bude tedy třeba vytvářet dokumenty podobající se ručně psanému textu. Ty bude třeba následně naskenovat. Dále je vhodné přidat umělý šum, aby byly obrázky stránek věrohodné a co nejvíce se podobaly skenovaným stránkám. Z takto vytvořených obrázků budou poté rozpoznávány dialogové akty. Ke správnému rozpoznávání je třeba mít velké množství trénovacích dat. Z tohoto důvodu jsme se rozhodli převádět text na tištěné stránky, protože v případě tisku a zpětném skenování by byla velká spotřeba papíru. Navíc umělým vytvářením obrázků stránek bude možné získat větší množství trénovacích dat. Zároveň bude možné uměle upravovat množství šumu a kvalitu obrázků a tím získat různé datasety pro testování klasifikátorů.

Struktura práce je následující. Nejdříve je popsán dataset Verbmobil, který bude v práci použit pro tvorbu vstupních dat. Dále následuje popis umělých neuronových sítí, které se budou využívat k optickému rozpoznávání znaků (OCR) a rozpoznávání dialogových aktů. Následuje popis metody OCR, která se bude používat pro rozpoznávání textu z obrázků simulujících naskenované stránky. Další kapitola popisuje analýzu problému, zde se budu zabývat možnými způsoby jak vytvářet MS Word dokumenty, jejich převodem do podoby obrázků, do kterých bude následně přidáván šum pro věrohodnější podobu naskenované stránky. Následně popíši systém Tesseract a modely, které budu používat pro rozpoznávání dialogových aktů. V po-

slední kapitole se budu zabývat implementací jednotlivých kroků a jejich výsledky.

2 Dataset Verbmobil

2.1 Dialogový akt

Dialogový akt definuje J. L. Austin [13] jako význam řečeného výroku na úrovni komunikační síly. Jednoduše řečeno, dialogový akt je funkce věty (nebo její části) v dialogu. Probíhá-li dialog mezi dvěma či více osobami, funkce jednotlivých vět se označuje jako dialogový akt. Například funkce oznamovací věty je předat nějakou informaci, funkce tázací věty je snaha o získání nějaké informace.

Dialogové akty se využívají například pro rozpoznání úmyslu věty. Pokud je věta tázací, jako odpověd se nejspíš očekává věta oznamovací s informací, na kterou se věta tázací ptá.

2.2 Korpus

Korpus používaný v této práci vznikl v rámci projektu Verbmobil. Cílem projektu bylo vytvořit mobilní aplikaci pro překlad spontánních dialogů. Kromě německých a anglických dialogů obsahuje i dialogy japonské [1].

Korpus obsahuje anotované promluvy, ve kterých si účastníci dialogu snaží domluvit nějakou schůzku, případně i nějakou další aktivitu, jako třeba výlet. Účastníci se v dialogu střídají po tzv. „kolech“.

Peter:		Petr:	
	Hello		Ahoj
	Do you have time tomorrow?		Měl bys zítra čas?
Adam:		Adam:	
	I do have time tomorrow.		Zítra mám čas.

Tabulka 2.1: Ukázka krátkého dialogu

V tabulce 2.2 je ukázka krátkého dialogu, kde v prvním kole je Petrův pozdrav a otázka a v druhém kole Adam odpoví.

Při anotaci se v korpusu dodržují definovaná pravidla popisující, jak od sebe oddělit jednotlivé promluvy. Níže si popíšeme základní jednotky dialogu.

1. **Kolo**
   Každé kolo by mělo obsahovat alespoň jednu větu nebo promluvu.

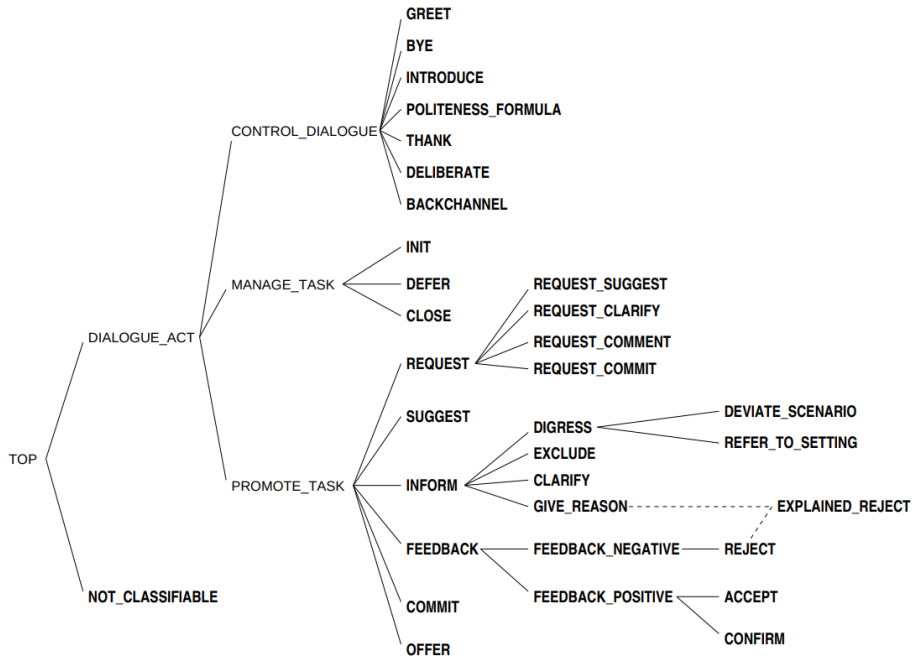
Platí, že v rámci jednoho kola mluví stále jeden řečník.

2. **Promluva**
   Promluva (*utterance*) je zde definována stejně jako věta. Věta musí obsahovat konečné sloveso a v případě komplexnějších vět, obsahujících více než jedno konečné sloveso se věta zpracovává následujícím způsobem. Buď se věta rozdělí na dvě promluvy, kde každá promluva obsahuje konečné sloveso, nebo musí mít druhé sloveso ve tvaru doplňku (complement) [1]. Například "I asked her to leave.", kde "to leave" je doplňkovým slovesem věty.

3. **Fráze**
   Určité dialogové akty je možné vyjádřit pomocí různých, někdy i jednoslovných frází. Například:
   `FEEDBACK_POSITIVE: 'ja', 'gut', 'alles klar', 'schön'` atd.
   `FEEDBACK_NEGATIVE: 'nein', 'tut mir leid', 'genau'` atd.
   `BACKCHANNEL: 'mhm', 'aha', 'ja'` atd.
   `THANK: 'danke', 'vielen dank'` atd.
   `GREET: 'guten Tag, 'hallo'` atd.
   Bez kontextu můžou některé fráze patřit do více klasifikačních tříd. Kontext ale tyto dráze jasně odděluje.

4. **Nominální skupina**
   Dialogové akty `SUGGEST, CLARIFY a DELIBERATE` mohou být vyjádřeny pomocí takzvané nominální skupiny (*nominal phrase*). Nominální skupina je fráze, která může ve větě plnit funkci podmětu, nebo předmětu. Například:
   `DELIBERATE: ja, also, im November.`
   `CLARIFY: <;T> a, eventuell, ja, am Dienstag.`
   `SUGGEST: vielleicht Montag und Dienstag, der achte, neunte November?`
   `ACCEPT: ja, das würde bei mir sehr gut passen.`
   `CLARIFY: Samstag, zwölfter und dreizehnter.`

2.2.1 Pravidla pro anotaci dialogových aktů



Obrázek 2.1: Rozhodovací strom pro anotaci dialogových aktů

K anotaci dialogových aktů korpusu se používá rozhodovací strom (viz obrázek 2.1). Postupným procházením stromu se dojde na některý z listů stromu, které představují klasifikační třídu (tj. dialogový akt). Na základě promluvy, kterou se snažíme anotovat, dochází v každém uzlu k rozhodování, kam se ve stromu dále vydat.

Například hned první otázkou zjistíme, zda je věta vůbec pochopitelná a dá se s ní nějak pracovat. Pokud ne, přiřadíme jí třídu `NOT_CLASSIFIABLE`. Pokud ano, pokračujeme na další uzel, kde rozhodujeme o další cestě stromem.

Dodaný korpus obsahuje dialogové akty a k nim přiřazenou klasifikační třídu. Vyskytuje se zde 16 tříd: `inform`, `thank`, `init`, `request`, `politeness_formula`, `backchannel`, `commit`, `feedback`, `close`, `suggest`, `deliberate`, `introduce`, `defer`, `bye`, `offer`, `greet`.

Rozdělení jednotlivých klasifikačních tříd je v rámci korpusu nerovnoměrné. Třídy `feedback`, `suggest`, `inform` a `request` jsou v korpusu zastoupeny ve velkém počtu a ostatních 12 tříd nemá velké zastoupení.

2.3 Formát dodaných dat

Vstupní data, která byla dodána vedoucím práce, vznikla převodem z původního Verbmobil formátu a jsou formátu `conll`. V souborech se vždy na řádce nachází jedno slovo a za ním jsou uvedeny parametry slovo popisující. Parametry jsou oddělené tabulátorem. Ve vstupních souborech bakalářské práce se jako parametry slova používá třída dialogového aktu a použitý jazyk (např. EN, GE).

V souborech jsou jednotlivá slova rozdělena do řádků. Dialogové akty jsou odděleny mezerou a jednotlivé dialogy pak středníkem.

are	REQUEST	EN
you	REQUEST	EN
going	REQUEST	EN
to	REQUEST	EN
the	REQUEST	EN
opera	REQUEST	EN
with	REQUEST	EN
Jay	REQUEST	EN
	;	
no	FEEDBACK	EN
all	SUGGEST	EN
I	SUGGEST	EN
have	SUGGEST	EN
free	SUGGEST	EN
on	SUGGEST	EN
November	SUGGEST	EN
is	SUGGEST	EN
the	SUGGEST	EN
fifth	SUGGEST	EN
sixth	SUGGEST	EN
and	SUGGEST	EN
seventh	SUGGEST	EN

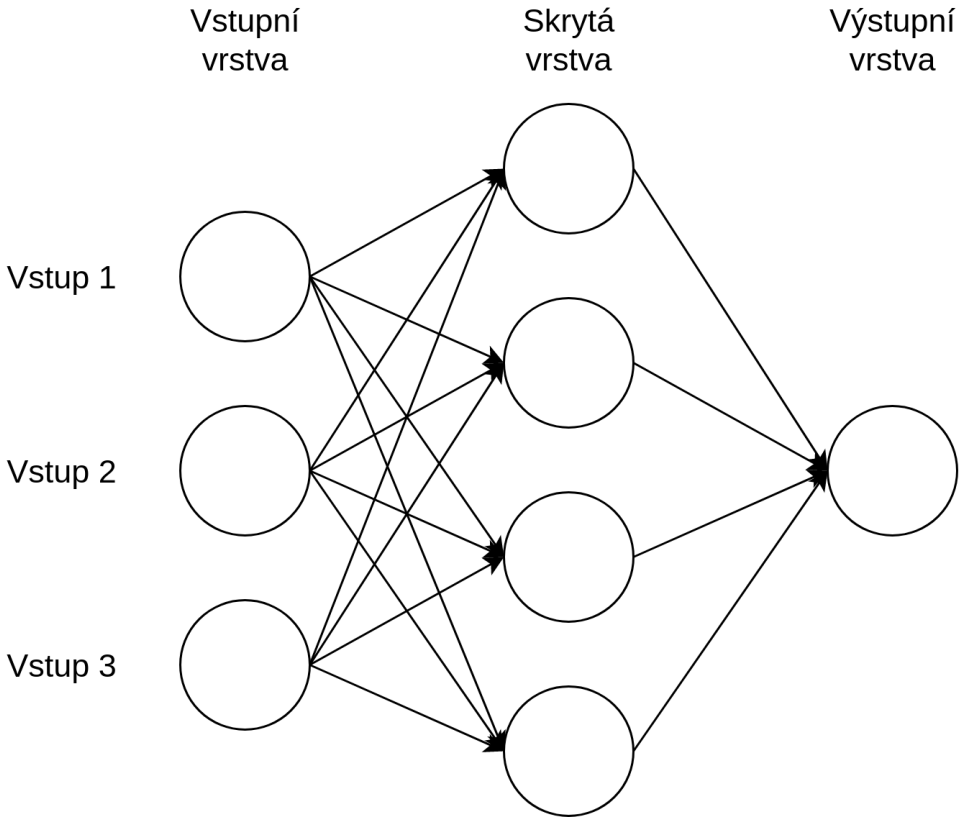
Tabulka 2.2: Ukázka dialogu z *conll* souboru

Tabulka 2.2 ukazuje dva dialogy z korpusu Verbmobil. První dialog se skládá pouze z jedné věty, která odpovídá dialogovému aktu. Druhý dialog se skládá ze dvou dialogových aktů zařazených jako FEEDBACK (*zpětná vazba*) a SUGGEST (*návrh*).

3 Umělé Neuronové sítě

V rámci této práce budou použity umělé neuronové sítě, proto následuje jejich stručný popis

Umělá neuronová síť je systém modelující biologické neurony. Každý neuron vyjadřuje matematickou funkci, která zpracovává vstupní informaci, a na základě vah všech vstupů rozhodne o výstupní informaci [18]. Každý neuron pracuje samostatně, zpracovává vstupní váhové uskupení a generuje výstup. Neurony jsou v síti uspořádány do vsrtev, kde každý neuron má ohodnocené spojení s každým neuronem následující vrstvy, ale neurony v jedné vrstvě spolu spojeny nejsou.



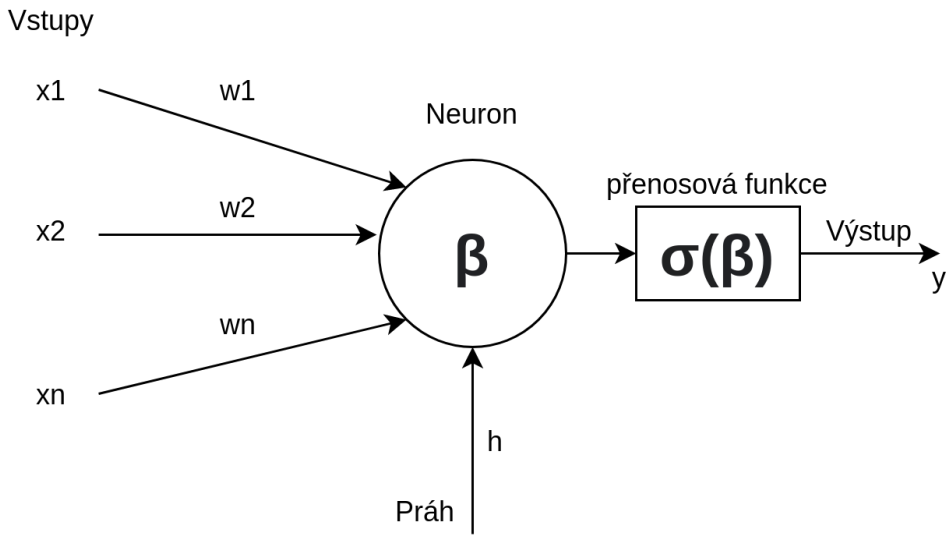
Obrázek 3.1: Příklad uspořádání neuronů do vrstev

V neuronové síti se vždy vyskytuje jedna vstupní a jedna výstupní vrstva. Skrytých vrstev může být několik a počet neuronů v každé z nich může být různý. Existuje tedy mnoho kombinací, jak sestavit neuronovou síť, a počet vrstev a neuronů se liší na základě problému, který má síť řešit. Každá

neuronová síť se musí nejprve trénovat. Je třeba nastavit váhu jednotlivých spojení podle trénovacího datasetu. Výhodou je, když se dataset vyvážený, tedy zastoupení klasifikačních tříd je rovnoměrné.

Základní jednotkou neuronové sítě je neuron. Vstupem do neuronu je vektor, který u dopředné plně propojené (feedforward) sítě obsahuje tolik prvků, kolik obsahuje předchozí vrstva neuronů x. Uvnitř neuronu se provede součet prvků vstupního vektoru, které jsou vynásobeny jejich příslušnými vahami w_i určenými při trénování sítě. Od této hodnoty se odečte hodnota prahu h.

$$\beta = \sum_{i=1}^{n} w_i x_i - h \tag{3.1}$$



Obrázek 3.2: Matematický model neuronu

Přenosová funkce poté provádí obecnou nelineární transformaci. Na základě vybrané přenosové funkce se vygeneruje výstup neuronu, který pokračuje do další neuronové vrstvy, nebo rovnou na výstup [11].

Váhy jednotlivých propojení neuronů jsou určeny během trénování sítě. Síť se učí pomocí algoritmu zpětného šíření chyby (*back-propagation*) [11]. Ten má k dispozici množinu vstupních a výstupních dat a nastavuje váhy jednotlivých spojení tak, aby docházelo k co nejlepším transformacím vstupních vektorů na příslušný výstupní vektor. Tento proces vyžaduje poměrně velké množství dat, a proto bývá dataset určený pro trénování neuronové sítě obsáhlý.

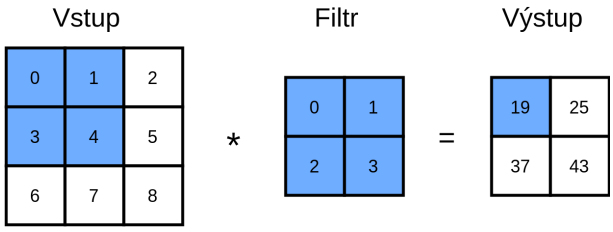
Neuronové sítě jsou schopny řešit problémy, které jsou analyticky řešitelné s velkými obtížemi nebo vůbec.

3.1 Konvoluční neuronové síťe

Konvoluční neuronové sítě (CNN) [15] byly původně vyvíjeny pro rozpoznávání obrázků nebo zvuků. Jedná se o jednu z důležitých a populárních neuronových sítí. Konvoluční část sítě se stará o předzpracování obrazu, který následně předá ve formě vektoru další části neuronové sítě. Tato vsrtva v podstatě extrahuje důležité informace nacházející se v obrázku a předá je klasifikátoru. Je tedy schopná redukovat počet vstupních parametrů.

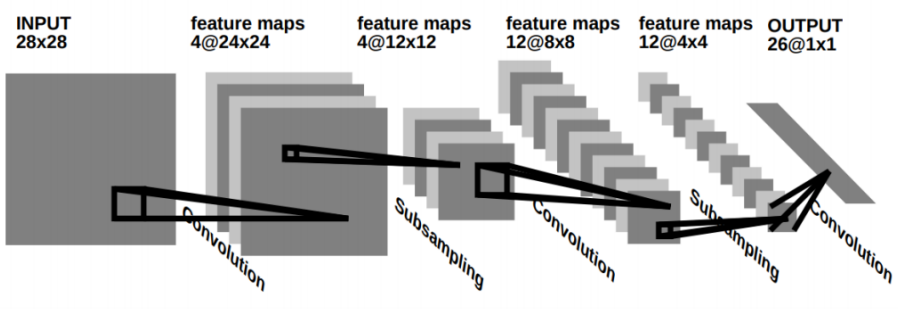
Typicky se CNN skládá z více konvolučních vrstev následovaných podvzorkováním (poolingovou vrstvou). Poslední vrstva bývá obvykle plně propojená. To znamená, že v poslední vrstvě má každý neuron spojení s každým neuronem v následující vrstvě.

V každé konvoluční vrstvě se nachází určitý počet filtrů (kernelů). Každý z těchto filtrů má nějakou velikost, např. 2x2.



Obrázek 3.3: Ukázka použití vah filtru na vstupní hodnoty pixelů obrázku

Hodnoty představující váhy, kterými se vynásobí hodnoty pixelů vstupního obrázku. Filtr se postupně posouvá po vstupním obrázku a počítá výstup. Podle nastavení těchto vah jsou jednotlivé filtry schopny detekovat určité vzory vyskytující se v obrázku. S přibývajícím počtem konvolučních vrstev jsou filtry schopny detekovat složitější vzory. Ukázku činnosti jednoduchého filtru na části obrázku znázorňuje obrázek 3.3.

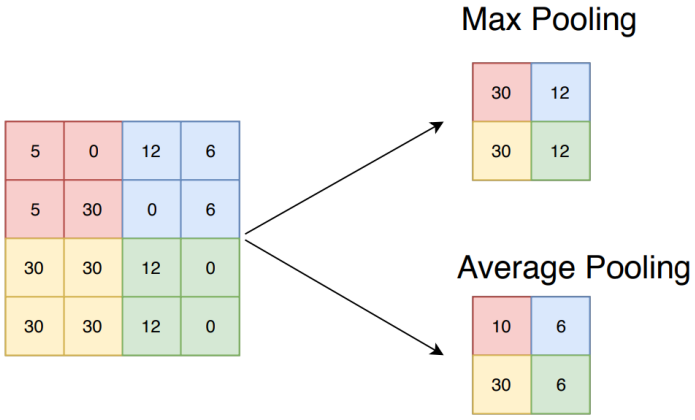


Obrázek 3.4: Ukázka konvolučních vrstev v CNN[15]

Na obrázku 3.4 je vidět několik vrstev složitější konvoluční neuronové sítě. Postupným procházením vrstvami se zmenšují a rozdělují vstupní data na jednotlivé části a vzory. Subsampling (podvzorkování) je způsob jak dále zmenšovat získaná data z vrstvy filtrů. Výsledkem je menší počet parametrů a zmenšování dimenze jednotlivých vstupů.

Podvzorkování

Podvzorkování, nebo také pooling, je metoda používaná pro redukci počtu parametrů a dimenze. Na obrázku 3.5 jsou vyobrazeny hlavní metody podvzorkování.



Obrázek 3.5: Ukázka max a average poolingu

Max pooling vybírá ze vstupu největší hodnotu a average pooling vezme průměrnou hodnotu vstupu. Max pooling vybírá vždy nejsvětlejší pixely. Používá se, když z obrázku chceme zachovat světlejší pixely. Average pooling se používá při vyhlazování obrázků.

3.2 LSTM sítě

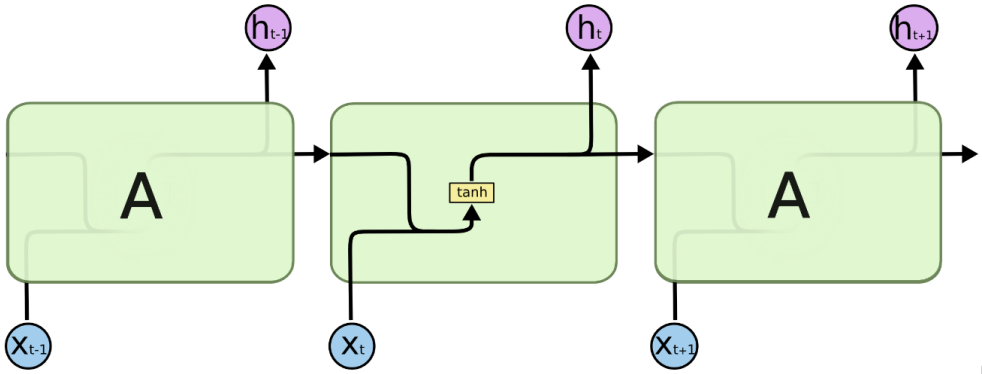
Long short-term memory (LSTM) [10] je rekurentní neuronová síť s pamětí. Používá se například v oblasti zpracování přirozeného jazyka, kde se snaží brát v potaz kontext předešlých slov, aby lépe rozpoznala význam vět, nebo odhadla následující slovo. Podobně jako člověk při čtení textu porozumí čteným slovům na základě slov již přečtených, tedy z předchozího kontextu.

Tradiční dopředné neuronové sítě tohoto nejsou schopny, a proto se využívá rekurentní neuronová síť, která obsahuje smyčky. Tímto způsobem je síť schopná uchovávat si předchozí kontext a v úlohách sekvenčního zpracování vstupu dává většinou výborné výsledky a díky tomu se v posledních letech využívá v mnoha disciplínách, například v rozpoznávání řeči, strojovém překladu, rozpoznávání textu a tak dále.

Sítě byly představeny pány Hochreiter a Schmidhuber (1977) a byly postupně popularizovány, neboť fungovaly velmi dobře pro řešení velkého množství problémů. LSTM sítě jsou speciálně navrženy, aby opravily nedostatky standartních rekurentních (RNN) sítí.

Problém u standartní RNN nastává, když je nějaký důležitý kontext příliš vzdálený a tudíž se jeho význam ztrácí. Pokud by měla standartní rekurentní síť (RNN) odhadnout následující slovo ve větě `Mraky jsou na ___`, tak správně určí následující slovo jako `obloze`. Pokud by ovšem měla RNN odhadnout následující slovo ve větě `Vyrostl jsem v Česku .... mluvím plynně ___`, tak nejbližší kontext naznačuje doplnění slova `jazykem`, ale v rámci dlouhodobého kontextu by dávalo větší smysl doplnit slovo `česky`.

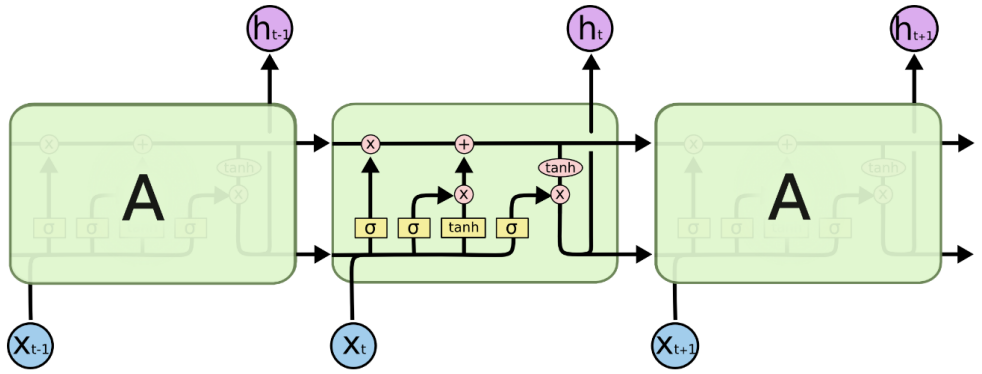
LSTM sítě jsou schopny tento problém vyřešit a jsou schopny brát v potaz i starší kontext. Od RNN se liší především strukturou vrstev. Ve standartních RNN sítích má modul celkem jednoduchou strukturu jedné neuronové vrstvy viz obr. 3.6. Dále si přiblížíme strukturu rekurentní sítě typu LSTM



Obrázek 3.6: Opakující se moduly standartní RNN sítě s jednou neuronovou vrstvou [8]

Obdélník představuje blok neuronové síťě, která zpracuje vstupní vektor x_t na jeden výstupní h_t. Tento vektor se získá spojením předchozího vektoru h_{t-1}, který byl vytvořen v předchozím kroku, a nového vstupního vektoru x_t.

`LSTM` sítě mají stejnou podobu řetězících se modulů jako standartní RNN síť, ale vnitřní struktura modulů je již oproti `RNN` sítím rozdílná. Struktura `LSTM` sítě je znázorněna na obrázku 3.7.



Obrázek 3.7: Opakující se moduly LSTM sítě se čtyřmi vrstvami neuronové sítě [8]

Každé spojení v obrázku představuje vektor, od vstupu na výstup, který prochází různými změnami, které zajistí správné fungování sítě. Růžové kroužky a elipsa představují vektorové operace, například násobení nebo součet. Žluté obdélníky představují natrénované vrstvy neuronové sítě. Spojující a rozdělující se čáry představují řetězení a kopírování vektorů.

Klíčem úspěšnosti `LSTM` sítě je pomocný vektor znázorněný v horní části modulu, který představuje dlouhodobou paměť modulu a probíhá celým řetězcem modulů. Modul má možnost tento vektor upravovat a rozhodovat,

zda-li je informace v něm obsažená stále relevantní a která informace z nového vstupu se do něj má přidat. Toho docílí pomocí takzvaných bran. Brány jsou označené písmenem σ a rozhodují, jaké části vstupní informace mají být přidány do dlouhodobé paměti. Pokud bude na vstupu informace, která zaslouží být uchována a přenesena do dalšího modulu, bude výstupní hodnota σ vrstvy `1`, v opačném případě `0`.

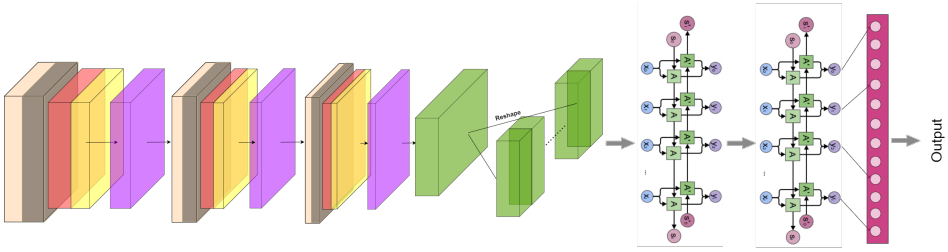
σ vrstvy modulu tedy rozhodují o tom, kterou z informací obsažených v dlouhodobé paměti má stále cenu zachovávat a kterou již není třeba pamatovat. Vrstva tak rozhodne ze vstupních vektorů x_t a h_{t-1}. Toto chování zajišťuje relevanci předchozí informace v dlouhodobé paměti. Obsah dlouhodobé paměti je dále porovnáván s novým vstupem. Například pokud na začátku věty bylo použito slovo určující rod, je možné, že s novým vstupem je třeba tento rod změnit na jiný a není již třeba zachovávat informaci o rodu předchozím.

Posledním krokem v modulu je vytvoření výstupního vektoru, který se spolu s dlouhodobou pamětí předá do dalšího modulu. Výstupní vektor se vytvoří kombinací aktuální dlouhodobé paměti a nově získané informace.

Toto je pouze jedna z mnoha variant `LSTM` sítí. Různé úpravy toku informací i rozložení jednotlivých vrstev sítí v modulu má různé následky na chování sítě. Pro určité specifické problémy se hodí jiné rozložení uvnitř modulů, například síť GRU (gated recurrent network).

3.3 CRNN

Konvoluční rekurentní neuronová síť je kombinací konvoluční neuronové sítě a rekurentní neuronové sítě [19]. Konvoluční část se opět stará o extrakci znaků, kterými se vstupní obrázek vyznačuje. Takto získaná data se poté předávají jako vstup pro rekurentní neuronovou síť, která využívá LSTM buňky popsané v sekci 3.2.



Obrázek 3.8: Ukázka architektury CRNN

Na výstup LSTM vrstvy se aplikuje aktivační funkce softmax a výsledek

pak představuje pravděpodobnostní rozdělení jednotlivých znaků v určitém časovém okénku. Znak, který má největší pravděpodobnost v daném čase, je pak rozpoznaným znakem z obrázku.

4 OCR

OCR [7] (*optické rozpoznávání znaků*) je metoda, která převádí text obsažený v obrázků nebo skenované stránce, do podoby textu.

Jiří Hlavenka [12] definuje pojem OCR jako technologii převodu dokumentu z digitální obrazové do textové podoby pomocí metod optického rozpoznávání znaků. Metoda je založena na porovnání hustoty bodů předlohy na pomyslné síti s typickými znaky jednotlivých písmen uloženými v paměti programu. Míra schopnosti rozpoznávání závisí na úrovni jazykové analýzy a národních slovníků a na možnostech doplňovat porovnávací databázi o nové znaky. Převádět lze tištěné i rukopisné znaky.

OCR má velké využití v automatickém zpracovávání textu v oblastech průmyslu a obchodu. Například čtení údajů z kreditních karet, kde pro větší přesnost a zjednodušení byl vyvinut speciální druh písma, který se v některých případech stále používá. Dalším využitím je například čtení čárových kódů, čtení adres z poštovních obálek, nebo i digitálních snímků. Začátkem 21. století se objevuje snaha poskytnout technologii širší veřejnosti a objevují se první nástroje, které tuto funkcionalitu nabízí zadarmo a online. OCR se hojně využívá v praxi při digitalizaci knih nebo dokumentů.

OCR je tedy technologie, která rozpozná text ze skenu nějaké stránky knihy nebo fotografie. Obecně je úspěšnost OCR velmi závislá na kvalitě vstupního souboru, většinou je třeba provádět předzpracování vstupních obrázků, aby byla úspěšnost co nejvyšší.

4.1 Předzpracování vstupních obrázků

Velkou roli při rozpoznávání znaků ze snímku hraje jeho důkladné předzpracování. Čím lepší bude kvalita vytvářených vstupních obrázků, tím úspěšnější bude i následné rozpoznávání obsahu těchto obrázků.

4.1.1 Rotace a oříznutí

Před zpracováním snímku je velmi užitečné mít text snímku vodorovný, protože se z něj poté mnohem lépe rozpoznává. Užitečné je i oříznutí hran, které mohou obsahovat velké množství šumu. Tímto způsobem lze výrazně zvýšit výslednou úspěšnost rozpoznávání [3].

4.1.2 Binarizace

Binarizace je proces, kdy se u každého pixelu snímku rozhodne, zda bude mít černou, nebo bílou barvu. Většinou se toho docílí převedením obrázku do šedotónového formátu a následným porovnáním hodnoty pixelu s definovaným prahem T [9].

$$g(x,y)\begin{cases}0 & \text{pro } f(x,y) < T \\ 1 & \text{pro } f(x,y) > T\end{cases} \tag{4.1}$$

Nová hodnota pixelu se určí porovnáním aktuální hodnoty jasu *f(x,y)* s nastaveným prahem T. Pokud bude hodnota nižší než práh, přiřadí se jí nová hodnota $g(x,y) = 0$. V opačném případě se pixelu přiřadí hodnota 1.

Takto se převedou všechny pixely na černou, nebo bílou barvu a je poté snazší rozpoznávat jednotlivé znaky viz obrázek 4.1.

Obrázek 4.1: Binarizace šedé fotografie [6]

4.1.3 Redukce šumu

Při redukci šumu se snažíme odstranit šum. Pro správné rozpoznávání znaků je třeba mít šum co možná nejmenší. Většinou lze redukce šumu docílit pomocí nepatrného rozostření snímku gausovským rozostřením [5].

4.2 Rozpoznávání vzorů

Dříve se pro rozpoznávání jednotlivých znaků používaly metody z oblasti rozpoznávání vzorů [16]. Každé písmeno je možné rozložit na sadu vzorů, ze kterých se skládá, a při automatické detekci těchto vzorů poté rozpoznat

nejpravděpodobnější písmeno, které bylo určeno jako výsledek klasifikace. Při řešení tohoto problému je možné vytvořit rozpoznávací systém na míru pro určitá konkrétní data. Takto vytvořený systém by měl sice dosahovat velmi dobrých výsledků, ale je velmi neflexibilní, a to především v případě rozpoznávání znaků z ručně psaného textu. Při psaní ručně psaného textu může člověk napsat stejný znak pokaždé trochu jinak. Proto se pro rozpoznávání vzorů a znaků z datasetu s měnícím se stylem písma často používají neuronové sítě, které dosahují dobrých výsledků i v případě nepravidelných znaků a jsou schopny se postupně učit.

4.3 Tesseract

Pro účely této práce byl použit systém Tesseract. Je vyvíjen již od roku 1985 společnostmi Hewlett-Packard Laboratories Bristol a Hewlett-Packard Co, Greeley Colorado a od roku 2006 jeho vývoj sponzoruje společnost Google. Tesseract byl vybrán, protože se jedná o jeden z nejlepších OCR enginů a má skvělé výsledky.

Balíček obsahuje OCR jádro `-libtesseract` a příkazovou řádkou ovládaný program `-tesseract`. Nové verze systému Tesseractu přidávají rozpoznávání textu po řádcích za pomoci rekurentních neuronových sítí. Tesseract podporuje `UTF-8` kódování znaků a lze použít pro více než 100 jazyků.

Pro účely bakalářské práce jsem se rozhodl použít knihovnu `py-tesseract`, která obaluje zmíněný Tesseract balíček a zároveň ji lze použít v python skriptech. Jedná se o open source nástroj. Jeho další velkou výhodou je podpora obrovského množství jazyků, možnost konfigurace a možnost trénování neuronové sítě, aby lépe rozpoznávala specifický dataset.

Jak již bylo řečeno, systém Tesseract je založen na umělých neuronových sítích, v současné verzi jde o rekurentní síť typu LSTM (`Long Short Term Memory Network`). Tesseract používá `LSTM` síť napsanou v jazyce C++ zvanou `CLSTM`. K numerickým operacím používá knihovnu `Eigen`.

4.4 Evaluační metody

Pro určení úspěšnosti OCR se v práci používají metriky WER (*word error rate*) a CER (*character error rate*) [14]. Vzorce pro výpočet WER a CER jsou následující:

$$WER\ (Truth, OCR) = \frac{WordDistance\ (Truth, OCR)}{WordCount\ (Truth)} \tag{4.2}$$

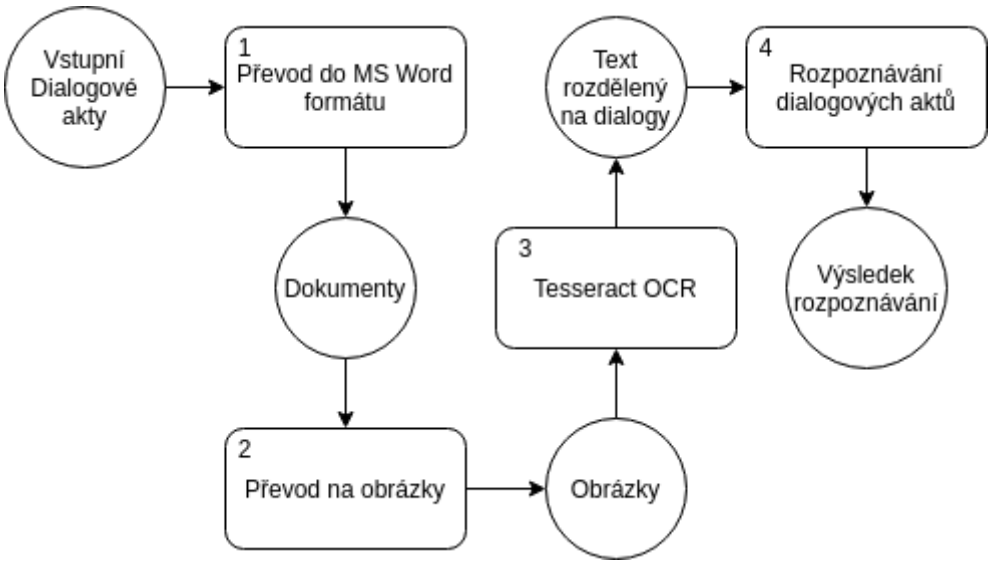
$$CER\ (Truth, OCR) = \frac{CharacterDistance\ (Truth, OCR)}{CharacterCount\ (Truth)} \tag{4.3}$$

WordDistance(Truth, OCR) a *CharacterDistance(Truth, OCR)* představují vzdálenost řetězců rozpoznaného textu a originálního textu. K určení této vzdálenosti se v projektu používá Levenshteinova vzdálenost dvou řetězců. Ta je definována jako minimální počet operací `vkládání`, `mazání` a `substitucí`, aby po jejich provedení byly dva řetězce totožné [2]. Vydělením této vzdálenosti s počtem slov, nebo znaků originálního řetězce dostaneme přesnost rozpoznaného textu.

5 Analýza problému

Jak již bylo uvedeno, cílem bakalářské práce je navrhnout a implementovat prototyp systému automatické rozpoznávání dialogových aktů z dokumentů v podobě rastrových obrázků. Převáděním elektronické podoby dokumentů do obrázků se vyhneme zbytečnému plýtvání velkého množství papíru. Navíc jsme díky tomu schopni lépe regulovat kvalitu vzniklých obrázků a přidávat do nich umělý šum.

Nejdříve bylo potřeba provést převod vstupního textu z elektronické podoby do podoby obrázků s co největší podobností s ručně psanému textu a věrohodností naskenované stránky. Z těchto stránek následně získat text s pomocí OCR a poté rozpoznávat jednotlivé dialogové akty. Blokové schéma navrženého prototypu systému je znázorněného na obrázku 5.1.



Obrázek 5.1: Diagram prototypu systému

5.1 Převod do MS Word formátu

Text bude nejprve potřeba převést do formy ze které by se dal snadno tisknout a zároveň simulovat podobu ručně psaného textu. Napodobit ručně psaný text je velmi obtížné, neboť i jednotlivá písmena mohou vypadat trochu jinak a přitom představovat stejný znak.

Ručně psaný text se dá napodobit pomocí různých fontů. Především některé fonty z programu Microsoft Word se ručně psanému textu velmi

podobají. Aby bylo možné vytvářet MS Word dokumenty, bude potřeba použít nějakou z dostupných knihoven.

Pro převod dialogových aktů získaných z korpusu Verbmobil v podobě `conll` formátu do formy Microsoft Word dokumentů je vhodné použít knihovnu `python-docx`. Knihovna obsahuje metody pro základní zacházení s Word dokumenty, například jejich vytváření, psaní odstavců, různé velikosti písma a především různé fonty, kterými lze text psát. Krom zapsání dialogů a jejich dialogových aktů bude třeba také zachovat informaci o tom, do jaké klasifikační třídy daný dialogový akt patří. Jednou z možností jak tohoto docílit by bylo napsat danou klasifikační třídu do dokumentu. To by ovšem mohlo snížit následnou úspěšnost `OCR`. Lepší možností je si informaci o klasifkační třídě uchovávat mimo dokumenty v odděleném souboru.

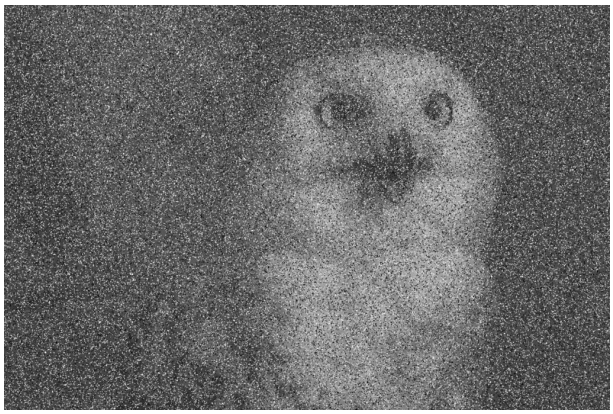
5.2 Převod z MS Word formátu na obrázky

Knihovna `python-docx` bohužel neumožňuje převod dokumentu do jiného datového typu než .docx a v našem případě bychom potřebovali dokument převést rovnou na jednotlivé obrázky. Jednou z možností jak tohoto docílit je převést soubor z formátu .docx do formátu PDF a z něj poté vytvořit dané obrázky. Uložit MS Word dokument do formátu PDF je možné přímo v aplikaci MS Word. K následnému převodu na obrázky bude použita knihovna `pdf2image`.

Kvalita takto vzniklých obrázků je ovšem příliš vysoká a jen těžko by se daly obrázky považovat za naskenované stránky. Proto bude potřeba kvalitu obrázků uměle snížit a co nejvíce napodobit podobu skenované stránky. K tomuto účelu mohou posloužit různé metody pro přidání umělého šumu do obrázků.

5.2.1 Salt and Pepper

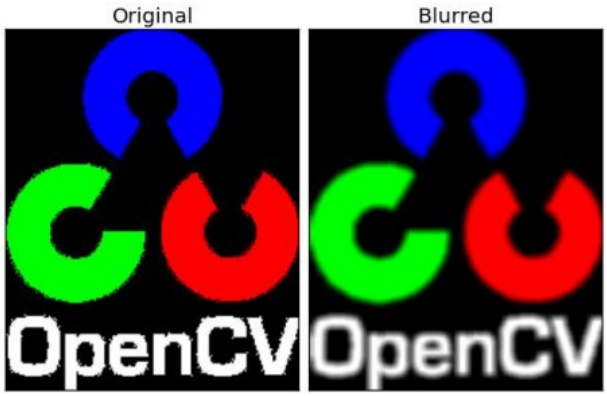
Metoda `Salt and Pepper` [20] je jednou z metod, která do obrázku přidává černé a bílé pixely. O tom jestli se pixel změní na černou nebo bílou rozhoduje parametr, který určuje počet změněných pixelů. Toto chování se snaží napodobit šum, který vzniká například výskytem většího množství prachu při pořizování obrázku viz obrázek 5.2.



Obrázek 5.2: Ukázka Salt and Pepper šumu

5.2.2 Gaussovo rozostření

Metoda funguje na základě analýzy pixelů v okolí daného bodu. Tedy hodnota každého pixelu je ovlivněna hodnotami jeho okolí. Jak moc bude výsledný obrázek rozostřený určuje parametr poloměru okolí. Hodnota pixelu se průměruje z definovaného okolí, čím větší toto okolí je, tím větší bude i rozostření. Výsledek této metody zobrazuje obrázek 5.3



Obrázek 5.3: Ukázka Gaussova rozostření

5.2.3 Poissonův šum

Poissonův šum [20] je svým účinkem podobný Salt and Pepper šumu. Narozdíl od metody Salt and Pepper šumu se ale řídí Poissonovým rozdělením. Podle tohoto rozdělení se pixelům přiřadí nová hodnota. Většinou se jedná o velký zásah do kvality obrázků. Přirozeně vzniká, když na digitální senzor dopadá rozdílný počet fotonů. Pro skenované stránky tedy není příliš běžný.

5.2.4 Přidání pozadí

Tato metoda přidává pozadí pod vygenerovanou textovou vrstvu. Pozadím může být nažloutlá, nebo pomuchlané stránka. To vytvoří velmi věrohodnou podobu naskenované stránky, především v kombinaci s některou z předešlích metod.

5.3 Tesseract OCR

V kapitole 4.1 jsem pojednával o nutném kroku předzpracování obrázků. Pokud chceme co možná nejúspěšnější rozpoznávání znaků, je třeba zbavit se šumu, který se ve vstupním obrázku nachází.

Obrázky zbavené šumu bude třeba převést zpět do textové podoby pomocí metod `OCR`. Jak bylo již výše uvedeno, k tomuto účelu bude použita knihovna Tesseract, která nabízí mnoho možností jak s rozpoznáváním zacházet.

Jednou z výhod systému Tesseract je jeho možnost konfigurace. Při konfiguraci je možné nastavit parametry tak aby `OCR` fungovalo co nejlépe na specifickém datasetu. Tyto parametry také mohou ovlivnit, jak se bude na stránku nahlížet, jestli jako na blok textu, nebo jednu řádku textu. Dále je možné přidat i whitelist znaků, které se v textu mohou objevit a tím i snížit výskyt nesmyslných znaků ve výsledném textu.

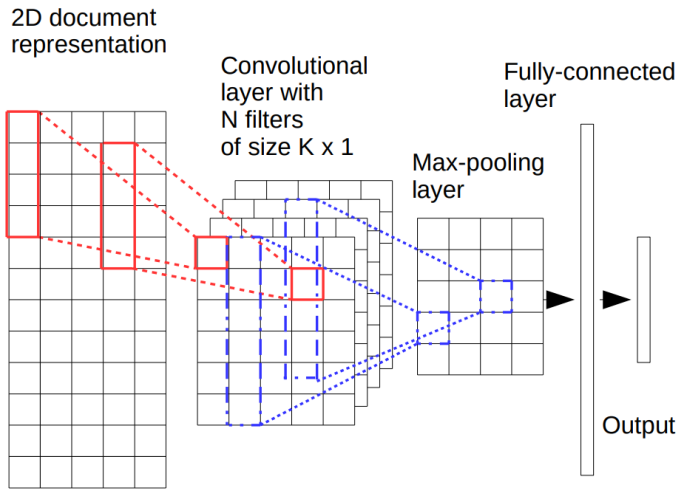
Základní metodou pro jednoduché zpracování stránky je metoda `image_to_string`. Ta převede s nastavenou konfigurací text v obrázku do elektronické podoby. Metoda je rychlá, ale neposkytuje žádnou kontrolu, nebo bližší údaje o výsledném textu.

Další metodou je metoda `image_to_data`. Tato metoda poskytuje především jednotlivé pozice rozpoznaných řádků slov a tím poskytuje vyšší kontrolu nad výsledným textem. Například pokud se z důvodu vyššího šumu vyskytnou náhodné znaky v pravém spodním rohu obrázku, metoda `image_to_string` tento nesmyslný text přidá do výsledku, ale s pomocnými informacemi, které dodá metoda `image_to_data`, můžeme sami rozhodnou, zda-li se jedná o novou řádku, nebo chybně rozpoznaný text mimo relevantní oblast obrázku. Tímto způsobem získaný text je třeba upravit do formy ve které jsou s ním modely pro rozpoznávání dialogových aktů schopny pracovat. Při zapisování vstupních `.conll` souborů do formátu Microsoft Word dokumentů bude ke každé datové sadě přiřazen vedlejší textový soubor, který určuje do které z klasifikačních tříd každý dialogový akt patří. Pro správnou funkci modelů je potřeba mít ke každémi dialogovému aktu odpověď učitele.

5.4 Rozpoznávání dialogových aktů

V rámci bakalářské práce budou použity dvě konvoluční neuronové sítě a jedna síť typu obousměrné LSTM (BiLSTM), konkrétně:

1. **CNN1**
   První konvoluční síť (CNN1) byla původně použita pro klasifikaci dokumentů. Jedná se o konvoluční síť s menším konvolučním jádrem, aby se přizpůsobila úloze rozpoznávání dialogových aktů oproti původní klasifikaci dokumentů viz obr. 5.4.

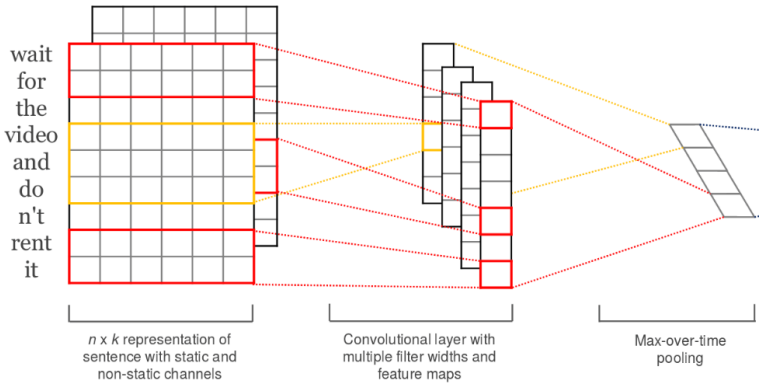


Obrázek 5.4: Architektura CNN1 [17]

   V síti se využívá 40 konvolučních jader s *relu* aktivační funkcí. Experiments have shown that it is possible to recognize dialogue acts with very good accuracy from dialogues in the form of images. Vrstva, následující za konvoluční vrstvou, obsahuje 256 neuronů a výstupní vrstva odpovídá počtu klasifikačních tříd. Jako chybová funkce se využívá Categorical Cross-Entropy a jako aktivační funkce se využívá softmax funkce.

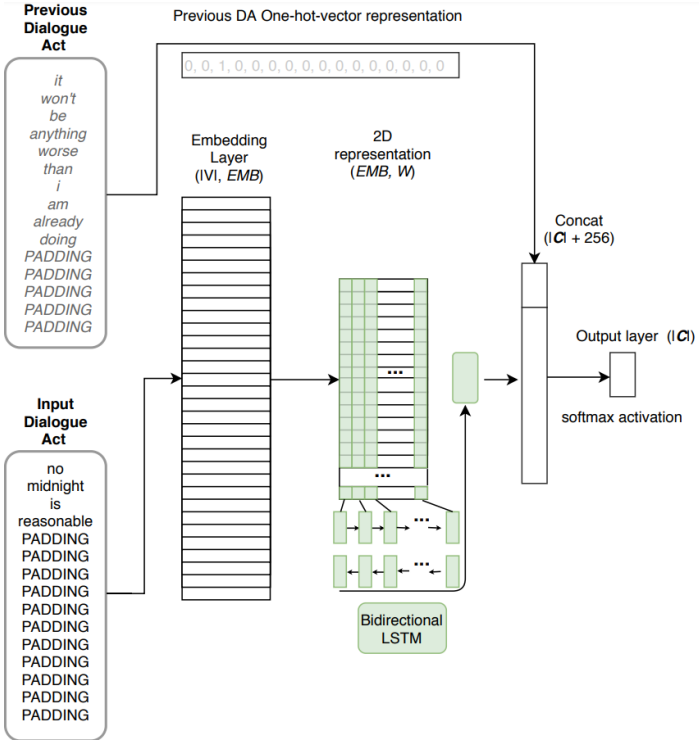
2. **CNN2**
   Druhý CNN model vychází z modelu, který byl navržen v práci [17]. Využívá tři různé velikosti konvolučních jader - (*3,EMB*), (*4,EMB*), (*5,EMB*), kde *EMB* představuje rozměr slovního vektoru (tzv. embedding). 100 jader těchto velikostí se počítá současně a jejich výstup

je spojen a předán plně propojené poslední vrstvě jako bylo použito v předchozím případě viz obr. 5.5.



Obrázek 5.5: Architektura CNN2 [17]

3. **BiLSTM**
   Tento model využívá obousměrných LSTM sítí. Vstup a embedding vrstva jsou stejné jako u CNN. Jádrem modelu je obousměrná LSTM vrstva, jejíž výstupní vektor se spojí s reprezentací předchozího dialogového aktu (one hot vector). Výstupní vrstva má počet neuronů, který odpovídá počtu klasifikačních tříd a využívá softmax aktivační funkci. Obrázek 5.6 ukazuje architekturu tohoto modelu.



Obrázek 5.6: Architektura BiLSTM [17]

6 Implementace

V následující kapitole popíši postup při řešení jednotlivých kroků. Nejprve popíši postup při vytváření MS Word dokumentů a způsob zápisu dialogových aktů. Dále se budu zabývat převodem vytvořených dokumentů do obrázkové podoby a možnými metodami pro přidání šumu a vytvoření obrázků co možná nejvíce podobných naskenované stránce. Poté popíši postup při rozpoznávání textu ze vzniklých obrázků, především předzpracování obrázků a nastavení systému Tesseract pro co nejlepší výsledky OCR. V poslední sekci popíši postup zpracování textové podoby dialogových aktů do formy vektorů, se kterými jsou klasifikátory schopny pracovat.

6.1 Tvorba dokumentů v MS Word formátu

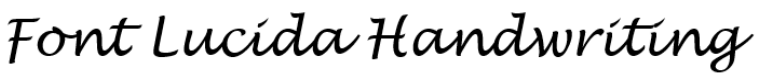
Prvním částí implementace bylo vytvoření Word dokumentů ze vstupních `.conll` souborů. K tomuto účelu jsem se rozhodl využít knihovnu `python-docx`. Knihovna obsahuje metody pro snadné vytváření a upravování Microsoft Word (.docx) dokumentů. Při vytváření dokumentů bylo naším cílem co nejvíce se přiblížit vzhledu ručně psaného textu. Uměle imitovat ručně psaný text není zcela možné. Když někdo píše text na papír, může napsat každý znak trochu jinak, toho lze v počítačích dosáhnout stěží. Je ale možné se k podobě ručně psaného textu přiblížit pomocí různých fontů. K tomuto účelu se jako nejvhodnější jeví fonty: `French Script MT` viz obr. 6.1, `Lucida Handwritting` viz obr. 6.3, `Mistral` viz obr. 6.2, `Segoe Script` viz obr. 6.4 a `Pristina` viz obr. 6.5.

Font French Script MT

Obrázek 6.1: Ukázka fontu French Script MT

Font Mistral

Obrázek 6.2: Ukázka fontu Mistral



Obrázek 6.3: Ukázka fontu Lucida

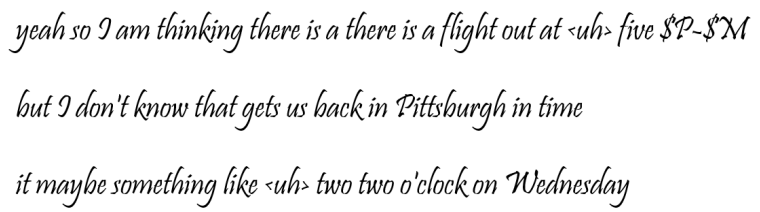
Obrázek 6.4: Ukázka fontu Segoe Script

Obrázek 6.5: Ukázka fontu Pristina

Vzhledem k moc velkému objemu práce při použití všech pěti fontů, bylo potřeba zvolit jeden typ písma, který nám připadal nejpodobnější ručně psanému textu. Na základě konzultace s vedoucím práce byl vybrán font Pristina.

První slovo každé řádky v `.conll` souboru představuje jednotlivá slova dialogových aktů. Při generování dokumentů z dialogových aktů jsem každý dialogový akt zapisoval jako nový odstavec a jednotlivé dialogy jsem odděloval jednořádkovou mezerou. To se ale ukázalo jako problémové, neboť při následném rozpoznávání textu z obrázkové formy dokumentů bylo těžké rozpoznat zda-li se jedná o další dialog, nebo další dialogový akt.

Proto se v aktuální verzi zapisuje vždy jeden dialog na jednu stránku dokumentu a mezi dialogovými akty v dialogu je jednořádková mezera viz obr. 6.6.



Obrázek 6.6: Ukázka dialogu zapsaného v docx dokumentu

Vzhledem k tomu že potřebujeme znát i klasifikační třídu, do které je každý dialogový akt zařazen, je vytvářen pomocný textový soubor ve kterém je tato informace uložena viz tab. 6.1.

;
FEEDBACK
;
SUGGEST
INFORM
REQUEST
;

Tabulka 6.1: Ukázka pomocného textového souboru

Stejně jako ve vstupních `.conll` souborech jsou zde dialogy odděleny středníkem. Pokud je dialog tvořen více dialogovými akty, jsou jejich klasifikační třídy zapsány za sebe na nové řádky v souboru. Třídy dialogových aktů ve druhém dialogu v tabulce 6.1 odpovídají dialogovým aktům z ukázkového dialogu viz obr. 6.6.

Skript určený pro vytváření dokumentů se nazývá `DocumentWriter.py`. Ve skriptu se načítají vstupní soubory. Aby skript správně fungoval je třeba mít již předem rozdělený dataset na trénovací, testovací a validační část. Z testovacího datasetu se vytvoří Microsoft Word dokument a k němu pomocný textový soubor s klasifikačními třídami. Trénovací a validační dataset se přepíše do podoby, ve které jsou s ním klasifikátory schopny pracovat. Dialogové akty musí být vždy zapsány na jeden řádek spolu s přiřazenou klasifikační třídou a jazykem.

6.2 Převod dokumentů v MS Word formátu na obrázky

Po vytvoření dokumentů ve formátu MS Word bylo nutné daný dokument převést do obrázkové podoby, aby bylo možné provést `OCR`. V tomto kroku bylo také nutné přidat do vznikajícího obrázku nějaký šum. Snažil jsem dosáhnout co možná nejreálnější podoby naskenované stránky. V kapitole 5.2 jsou popsány různé metody tvorby šumu, které se mohou vyskytovat v obrázcích. Metody `Salt and Pepper` a `Poissonův šum` se vyskytují zřídka v elektronické podobě naskenovaných stránek. Proto jsem se rozhodl pro kombinaci přidaného pozadí viz 5.2.4 a Gaussovo rozostření viz 5.2.2. Vzhledem k tomu, že knihovna `python-docx` poskytuje pouze základní metody pro práci s Microsoft Word dokumenty, není možné s její pomocí přidávat do dokumentů pozadí. Proto bylo pozadí do dokumentů přidáno ručně v aplikaci MS Word. Po přidání pozadí je třeba dokument uložit ve formátu PDF a

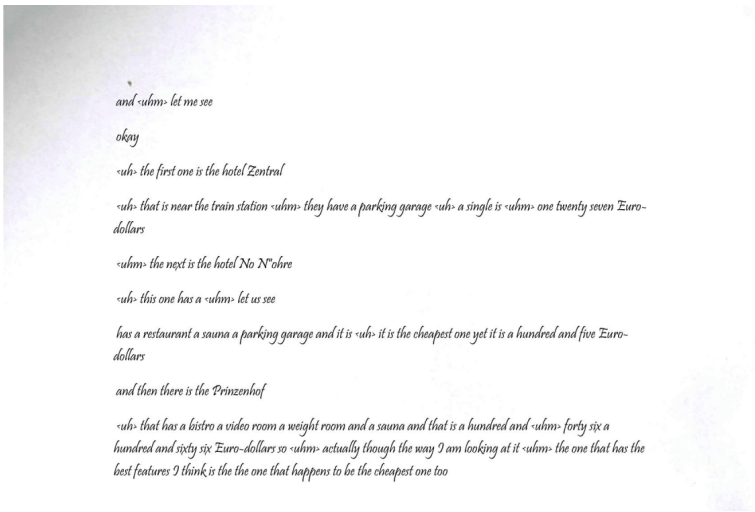
nakopírovat do složky `documents` v pracovním adresáři projektu, ze kterého se dokumenty při převodu načítají.

Pro převod PDF dokumentů do obrázků se v projektu používá knihovna `pdf2image`. Ta převede všechny stránky dokumentu do jednoho pole obrázků. Z tohoto pole poté vznikají dva sety obrázků. Jeden pouze s přidaným pozadím a druhý, do kterého se kromě pozadí přidá již zmíněné Gaussovo rozostření. Intenzitu Gaussova rozostření je možné definovat v konfiguračním souboru `config.py`.

Obrázky s přidaným šumem se v posledním kroku náhodně natočí na jednu stranu, neboť při skenování stránek se často stane, že je papír lehce natočený a není perfektně srovnaný s okraji skeneru.

Skript určený pro převod PDF dokumentů do obrázků se nazývá `main_docimg.py`. Obsahuje metody využívající již zmíněnou knihovnu `pdf-2image` a pro jeho správnou funkci je třeba mít ve složce `documents` připravené PDF dokumenty. Vznikající obrázky se ukládají do adresáře `EN_image_dacts/ EN_test_image_dacts/`, kde se vytváří příslušné podadresáře. V adresářích s příponou `_aug` se nachází obrázky s přidaným Gaussovo rozostřením a v adresářích bez přípony jsou uloženy obrázky, ve kterých je jediným šumem přidané pozadí.

Ukázka MS Word dokumentu bez přidaného Gaussova rozostření viz obr. 6.7 a s přidaným Gaussovo rozostřením viz obr. 6.8


and <uhm> let me see

okay

<uh> the first one is the hotel Zentral

<uh> that is near the train station <uhm> they have a parking garage <uh> a single is <uhm> one twenty seven Euro-dollars

<uhm> the next is the hotel No N"ohre

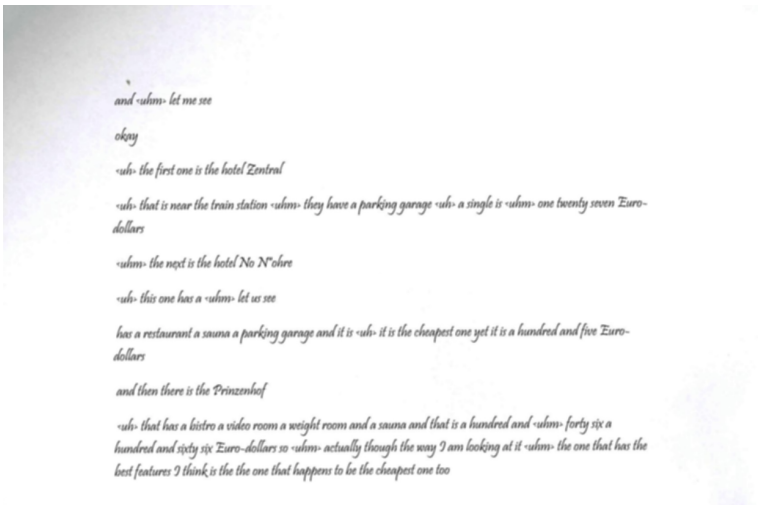
<uh> this one has a <uhm> let us see

has a restaurant a sauna a parking garage and it is <uh> it is the cheapest one yet it is a hundred and five Euro-dollars

and then there is the Prinzenhof

<uh> that has a bistro a video room a weight room and a sauna and that is a hundred and <uhm> forty six a hundred and sixty six Euro-dollars so <uhm> actually though the way I am looking at it <uhm> the one that has the best features I think is the the one that happens to be the cheapest one too

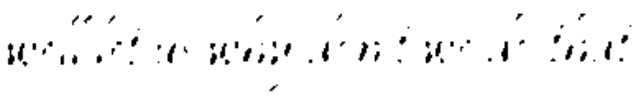
Obrázek 6.7: Ukázka části dokumentu bez Gaussova rozostření



Obrázek 6.8: Ukázka části dokumentu s Gaussovo rozostřením

6.3 OCR

Pro převod z obrázku do textu byla použita knihovna `py-tesseract`. Předtím než bylo možné z obrázků rozpoznávat text, bylo potřeba provést předzpracování vstupních dat. Každý obrázek bylo třeba nejprve zbavit šumu. Rohy stránek bývají často ohnuté, což se na skenu projeví tmavě šedými pixely, které může následně systém Tesseract chybně rozpoznat jako text. Zbavit se šumu, který vznikl díky zahnuté, nebo nažloutlé stránce nebylo zcela možné. Oříznutí hran obrázku, kde se tyto vady papíru často nachází bylo realizováno. Následně se ve skriptu provede zvětšení obrázku. Obrázek se zvětší, díky čemuž je pak následná binarizace přesnější a účinnější. Posledním krokem v předzpracování je převedení obrázku do odstínů šedi a následná binarizace. Vzhledem k tomu, že se na stránkách nacházejí i šedé pixely, kterým by při binarizaci s prahem 125 byla přiřazena černá barva, rozhodl jsem se použít nižší práh. To se ovšem ukázalo jako problematické v případě datasetů s přidaným Gaussovo rozostřením.



Obrázek 6.9: Ukázka vlivu binarizace s nižším prahem na dataset s Gaussovo rozostřením

Díky nižšímu prahu a rozostřenému obrázku se po binarizaci stal text nečitelný viz obr. 6.9. Rozostření zapříčinilo, že znaky ztratily jasné obrysy a přímo černé pixely se nacházely pouze na některých místech. Z tohoto důvodu bylo třeba pro dataset s rozostřením použít vyšší práh viz obr. 6.10

Obrázek 6.10: Ukázka vlivu binarizace s vyšším prahem na dataset s Gaussovo rozostřením

Při použití vyššího prahu se zvýší tloušťka jednotlivých znaků, ale znaky získají jasné obrysy. Při zpracování takto upravených obrázků bylo dosaženo při následném rozpoznávání textu lepších výsledků, než při zpracování obrázků bez binarizace.

Pro datasety, ve kterých se Gaussovo rozostření nevyskytuje, bylo (v případě binarizace) možné použít nízký práh. Znaky v těchto datasetech mají ostré hrany, tudíž se binarizací pouze odstraní šum vzniklý přidáním pozadí viz obr. 6.11.

Obrázek 6.11: Ukázka vlivu binarizace s nižším prahem na dataset bez Gaussova rozostření

Oříznutí hran obrázků, zvětšení obrázků, převedení do šedotónového formátu i následnou binarizaci zajišťují metody knihovny `open-cv`.

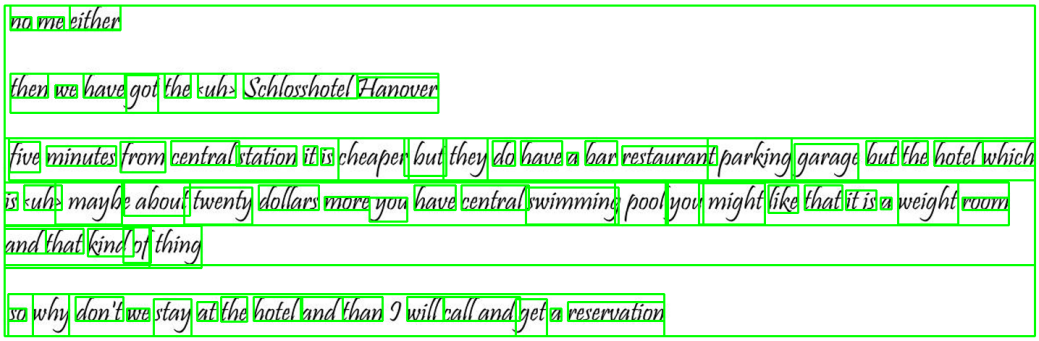
Obrázky upravené tímto způsobem jsou připraveny k rozpoznávání textu. V tabulce 6.2 jsou výsledky rozpoznávání bez nastavení vlastní konfigurace systému Tesseract. Použili jsme metriky WER a CER, které se pro danou úlohu běžně používají.

Pro lepší výsledky bylo třeba napsat vlastní konfiguraci. Jeden z hlavních parametrů konfigurace je jaká segmentace stránky se má používat. Například

při segmentaci s parametrem `psm- 7` bude Tesseract se stránkou zacházet jako s jedním řádkem textu. Nejlepších výsledků jsem dosáhl s použitím parametru segmentace `psm- 6`, při které se stránkou zachází jako s jednotným blokem textu. Dalším užitečným parametrem je pak jazyk, ve kterém by měl být text na obrázcích napsán a seznam povolených znaků (whitelist). Do tohoto seznamu jsem přidal všechny znaky abecedy, čísla a speciální znaky, které se ve vstupních datech vyskytovaly, například `< , >, ’, $` nebo `%`. Pouze znaky, které se na seznamu nachází pak budou mezi rozpoznanými znaky z obrázku.

Nejprve jsem pro rozpoznávání textu použil základní metodu knihovny, tedy `image_to_string`. Tato metoda se ukázala jako velmi chybová. Metoda převedla nalezený text na předaném obrázku do jednoho řetězce a často chybně rozpoznávala mezery mezi jednotlivými dialogovými akty.

Jako lepší možnost se ukázala metoda `image_to_data`.



Obrázek 6.12: Ukázka nalezených pozic metodou `image_to_data`

Tato metoda poskytuje informace o pozicích rozpoznaných slov viz obr. 6.12. To je velmi užitečné pro rozhodování, zda-li daný dialogový akt pokračuje na další řádce, nebo již začíná další dialogový akt. Informace o pozicích slov také pomohou při detekci chybně rozpoznaného textu. Pokud se nějaká rozpoznaná slova nacházejí pouze pár pixelů od horní hrany obrázku, lze předpokládat, že se jedná o špatně rozpoznaný text následkem šumu. Pokud budou mít rozpoznaná slova podezřele velkou mezero oproti slovům předchozím, dá se předpokládat, že je opět jedná o chybně rozpoznaný text následkem nějakého šumu. Díky těmto informacím je možné lépe kontrolovat mezery mezi dialogovými akty.

	0	0_aug	1	1_aug	2	2_aug	3	3_aug
WER	23.83	45.11	24.84	44.14	34.27	41.23	35.39	55.74
CER	13.46	21.72	17.68	24.14	14.89	19.33	20.41	28.95

Tabulka 6.2: Výsledky OCR s použitím standardní Tesseract konfigurace [v %]

Tabulka 6.2 ukazuje, že výsledky prvního OCR při kterém nebyla použita vlastní konfigurace systému Tesseract nebyly příliš přesné. Při optimálním nastavení systému Tesseract se výsledky výrazně zlepšily viz tab. 6.3. Zlepšení bylo zvláště patrné v případě datasetů se šumem (_aug).

	0	0_aug	1	1_aug	2	2_aug	3	3_aug
WER	19.12	24.22	19.16	25.07	19.59	25.24	19.43	26.21
CER	5.65	7.43	5.68	7.73	5.77	8.03	5.75	8.13

Tabulka 6.3: Výsledky OCR s použitím vlastní Tesseract konfigurace [v %]

K převodu obrázků do textové podoby jsem implementoval skript `main_imgtotxt`, který si postupně načte obrázky z podadresářů adresáře `EN_image_dacts/EN_test_image_dacts/`. Rozpoznaný text uloží do dočasného textového souboru.

6.4 Rozpoznávání dialogových aktů

Pro rozpoznávání dialogových aktů se v projektu používají dvě konvoluční neuronové sítě a jedna obousměrná LSTM síť viz 5.4. Nejprve bylo třeba připravit pro klasifikátory vstupní data, což zajišťuje skript `data_loader.py`. Ten si načte soubory `test.txt, train.txt a val.txt`. Tyto soubory představují připravené datasety pro testování, trénování a validaci.

well let us why don't we do that;FEEDBACK;en
all right;FEEDBACK;en
hi Jim;GREET;en
are you going to the opera with Jay;REQUEST;en
no;FEEDBACK;en
all have free on November is the fifth sixth and seventh;SUGGEST;en

Tabulka 6.4: Ukázka formátu souboru test.txt

V tabulce 6.4 je vypsána malá část souboru `test.txt`. Tento formát mají i soubory `train.txt a val.txt`. Obsah souboru se načítá po řádcích a je

rozdělen na věty (utterance) a klasifikační třídy, do kterých je věta přiřazena (label).

Načtené anotace je následně třeba převést do formy vektorů. Vzhledem k tomu že pracuji s 16 klasifikačními třídami, bude vektor vždy velikosti 16. Vektor bude obsahovat samé nuly, až na pozici, ve které se daný anotace nachází v souboru `config.py` v parametru `categories`. Například klasifikační třída FEEDBACK se nachází na pozici 0, její vektor bude tedy vypadat následovně: `[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]`. Tímto způsobem se převedou všechny načtené anotace.

Načtené věty je také potřeba převést do formy vektorů. Pro vektorovou reprezentaci jednotlivých slov jsme použili model word2vec [4]. Ten obsahuje vektorovou reprezentaci všech slov vyskytujících se ve vstupních datech. Důležitá vlastnost těchto vektorů je v jejich vzdálenosti mezi sebou. Platí, že slova, která si jsou významově blízká mají podobné vektory a jsou si tedy blízko ve vektorovém prostoru Například slovo "mom"má významově blíž ke slovu "dad", než například ke slovu "mustard"a proto bude vzdálenost slovního vektoru odpovídajícího slovu "mom"ke slovnímu vektoru pro slovo "dad"mnohem bližší než k vektoru slova "mustard". Pro převedení věty do vektoru je tedy potřeba najít vektorovou reprezentaci každého slova věty v souboru `en_w2v.vec`.

Vektory anotací mají vždy stejnou délku, ale každá věta může obsahovat různý počet slov a tudíž i vektorová reprezentace těchto vět může mít rozdílnou velikost. Z tohoto důvodu se zavádí padding symbol `%%PADDING%%` a společná velikost vět nastavená v parametru `text_input_size` v konfiguračním souboru (`config.py`). Pokud by věta obsahovala menší počet slov, než nastavená velikost vět, doplní se věta padding symboly. Pokud by věta obsahovala větší počet slov než je nastavená délka vět, vezme se z věty pouze tolik slov, aby jejich počet odpovídal nastavené délce vět.

Takto vytvořené vektorové reprezentace vět a anotací se předávají na vstup klasifikátorům. Nejprve se klasifikátor trénuje z dat vytvořených ze souboru `train.txt` a poté provádí klasifikaci u dat vytvořených ze souboru `test.txt`. V bakalářské práci jsem použil nastavení hyperparametrů dle [17].

6.5 Dosažené výsledky

Úspěšnost klasifikace jednotlivých modelů je uvedena v tabulce 6.5. K vyhodnocení byla použita metrika accuracy (*přesnost*). Accuracy je poměr správných předpovědí vůči celkovému počtu vstupních vzorků.

	0	0_aug	1	1_aug	2	2_aug	3	3_aug
BiLSTM	69.9	67.7	69.8	60.3	69.6	64.9	70.1	65.1
CNN1	69.8	67.8	69.9	60.5	69.1	65.2	69.3	65.2
CNN2	68.2	66.4	68.6	58.2	68.4	64.1	67.8	63.2

Tabulka 6.5: Výsledky klasifikace jednotlivých modelů [v %]

Úspěšnost modelů BiLSTM a CNN1 je velmi podobná. Úspěšnost modelu CNN2 je o trochu nižší. Datové sady s přidaným Gaussovo rozostřením mají nižší úspešnost. To bylo očekávané chování vzhledem k tomu, že WER a CER je u těchto datových sad vyšší než u sad bez Gaussova rozostření. Úspěšnost klasifikace datové sady `1_aug` je ale nižší než u ostatních sad s přidaným Gaussovo rozostřením.

Překvapivým výsledkem byly malé rozdíly ve výsledcích mezi datovými sadami bez Gaussova rozostření. Kromě datové sady `0` bylo do každé další přidáno pozadí, které simuluje podobu naskenované stránky. V procesu předzpracování se ale prováděla binarizace obrázků a ta provedla odstranění šumu vzniklého přidáním pozadí. Následkem toho nebyl při následném přepisu do textu v obrázkových sadách bez Gaussova rozostření skoro žádný rozdíl a výsledné texty jsou téměř totožné. Z tohoto důvodu jsou i výsledky těchto sad velmi podobné.

7 Závěr

V rámci této bakalářské práce jsem se věnoval problematice rozpoznávání dialogových aktů, vytváření obrázkových datových sad a jejich převodem do textové podoby.

Proto jsem se musel seznámit s korpusem Verbmobil, díky čemuž jsem získal představu o dialogových aktech a jak se anotují. Dále jsem se seznámil s možnými metodami pro práci s Microsoft Word dokumenty a jejich následným převodem do obrázkové podoby.

Při vytváření obrázkových sad byl kladem důraz na co možná největší podobnost k ručně psanému textu. K tomu byl vybrán font Pristina. Pro vytváření dokumentů byla použita knihovna `python-docx`. Aby vzniklé datové sady obrázků vypadaly co možná nejvíce jako reálné skenované stránky, bylo třeba se seznámit s metodami a způsoby jak přidat do obrázků šum.

Většina metod, které přidávají umělý šum do fotografií a obrázků, působila příliš uměle, a proto bylo nakonec rozhodnuto pro kombinaci Gaussovo rozostření a přidání pozadí. Jako pozadí byly přidány zmuchlané papíry, které dobře simulují poškozenou, skenovanou stránku.

Pro převod obrázkových sad do textové podoby bylo nutné se seznámit s metodami OCR a to konkrétně se systémem Tesseract. Dále bylo třeba prozkoumat metody používané pro předzpracování (preprocessing) obrázků, neboť jsou tyto metody schopné výrazně zvýšit úspěšnost následného OCR. Proto byla použita knihovna `openCV`, která nabízí mnoho metod pro snadnou práci a úpravu obrázků.

Pro klasifikaci dialogových aktů bylo nutné prozkoumat možnosti neuronových sítí a metody pro přípravu dat. Pro reprezentaci textu byl použit systém `word2vec`, jehož pomocí byly vytvořeny vstupní vektory dimenze 300.

Úspěšnost OCR se odvíjela především od nastavení systému Tesseract. Při použití vlastního nastavení se u obrázkových sad s přidaným Gaussovo rozostřením se WER průměrně snížil o 21.25 % a CER o 12.4 % oproti standardní konfiguraci. U obrázkových sad bez rozostření k tak velkému zlepšení nedošlo, WER se průměrně snížilo o 10 % a CER o 11 %. Úspěšnost klasifikace modelů se příliš nelišila. Síť BiLSTM byla ve výsledku nepatrně lepší než CNN1 a CNN2.

Vzhledem k tomu, že se jedná o první pokus rozpoznávání dialogových aktů z tištěných dokumentů, nabízí se celá řada rozšíření této práce. Je možné provést podobnou analýzu a experimenty na jiných jazycích. V pří-

padě korpusu Verbmobil je možné použít existující prototyp systému na zpracování němčiny. Toto by bylo možné provést v podstatě bez zásahu do zdrojových kódů stávajícího programu. Dalším zajímavým směrem práce by byla analýza dialogu v komixech. Zde by bylo potřeba doplnit detekci textu z ilustrací a rozlišit, ve kterých sekcích se jedná o dialog a co jsou doprovodné texty. V rámci dalších prací by bylo možné se dále zabývat vícejazyčností, což je dnes velmi aktuální téma.

Literatura

[1] Alexandersson, J. et al. Dialogue Acts in VERBMOBIL-2 Second Edition. 09 1998.

[2] *Levenshteinova vzdálenost* [online]. Oracle, 2020. [cit. 2021/01/16]. Algoritmy.net Levenshteionava vzdálenost. Dostupné z: `https://www.algoritmy.net/article/1699/Levenshteinova-vzdalenost`.

[3] Bieniecki, W. – Grabowski, S. – Rozenberg, W. Image Preprocessing for Improving OCR Accuracy. In *2007 International Conference on Perspective Technologies and Methods in MEMS Design*, s. 75–80, 2007. doi: 10.1109/MEMSTECH.2007.4283429.

[4] Bojanowski, P. et al. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*. 2017, 5. doi: 10.1162/tacl_a_00051. Dostupné z: `https://doi.org/10.1162/tacl_a_00051`.

[5] Deng, G. – Cahill, L. An adaptive Gaussian filter for noise reduction and edge detection. 1993, s. 1615–1619 vol.3. doi: 10.1109/NSSMIC.1993.373563.

[6] Eddins, S. *Image binarization – new R2016a functions* [online]. MathWorks.com, 2016. [cit. 30.4.2021]. Dostupné z: `https://blogs.mathworks.com/steve/2016/05/16/image-binarization-new-r2016a-functions/`.

[7] Eikvil, L. OCR - Optical Character Recognition. 1993.

[8] Filip Zelic, A. S. *A comprehensive guide to OCR with Tesseract, OpenCV and Python* [online]. Nanonets, 2020. [cit. 2021/04/26]. Lstm sítě. Dostupné z: `https://nanonets.com/blog/ocr-with-tesseract/#introduction`.

[9] Gupta, M. R. – Jacobson, N. P. – Garcia, E. K. OCR binarization and image pre-processing for searching historical documents. *Pattern Recognition*. 2007, s. 389–397. doi: https://doi.org/10.1016/j.patcog.2006.04.043. Dostupné z: `https://www.sciencedirect.com/science/article/pii/S0031320306002202`.

[10] Hochreiter, S. – Schmidhuber, J. Long Short-Term Memory. *Neural Computation*. 11 1997, 9, 8, s. 1735–1780. ISSN 0899-7667. doi:

10.1162/neco.1997.9.8.1735. Dostupné z: https://doi.org/10.1162/neco.1997.9.8.1735.

[11] Hopfield, J. Artificial neural networks. *IEEE Circuits and Devices Magazine*. 1988, 4, 5, s. 3–10. doi: 10.1109/101.8118.

[12] Jiří, H. *Nový výkladový slovník výpočetní techniky*. Computer Press, 1995. ISBN 80-85896-13-3.

[13] J.L.Austin. *How to do Things with Words*. Clarendon Press, 1962.

[14] Kolak, O. – Byrne, W. – Resnik, P. A Generative Probabilistic OCR Model for NLP Applications. 2003, s. 134–141. Dostupné z: https://www.aclweb.org/anthology/N03-1018.

[15] Lecun, Y. – Bengio, Y. *Convolutional networks for images, speech, and time-series*. MIT Press, 1995.

[16] Mantas, J. An overview of character recognition methodologies. *Pattern Recognition*. 1986, s. 425–430. ISSN 0031-3203. doi: https://doi.org/10.1016/0031-3203(86)90040-3. Dostupné z: https://www.sciencedirect.com/science/article/pii/0031320386900403.

[17] Martínek, J. et al. Multi-Lingual Dialogue Act Recognition with Deep Learning Methods. In *Interspeech 2019*, s. 1463–1467, Graz, Austria, 15-19 September 2019. doi: 10.21437/Interspeech.2019-1691.

[18] Schalkoff, R. Artificial Inteligence: An Engineering Approach. 1990.

[19] Shi, B. – Bai, X. – Yao, C. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2017, 39, 11, s. 2298–2304. doi: 10.1109/TPAMI.2016.2646371.

[20] Verma, M. R. – Ali, D. J. A Comparative Study of Various Types of Image Noise and Efficient Noise Removal Techniques. 2013. Dostupné z: https://www.researchgate.net/profile/Rohit-Verma-31/publication/307545428_A_comparative_study_of_various_types_of_image_noise_and_efficient_noise_removal_techniques/links/60094667a6fdccdcb86bd105/A-comparative-study-of-various-types-of-image-noise\-efficient-noise-removal-techniques.pdf.

A Obsah přiloženého DVD

K bakalářské práci je přiloženo DVD s elektronickou verzí práce, zdrojovými soubory a potřebnými soubory a složkami pro správnou funkci programu. Obsah DVD je následující:

1. `data_conll`
   Složka obsahuje jeden podadresář `EN`, ve kterém se nachází zdrojové vzorky dat text.conll, train.conll a val.conll.

2. `data`
   Složka obsahuje podadresáře vytvářené při převodu obrázků do textu. V podadresářích se nachází vytvářené textové soubory pro následné rozpoznávání dialogových aktů.

3. `EN_image_dacts`
   Složka obsahuje podadresáře, do kterých se ukládají upravené obrázky, připravené na extrahování textu.

4. `dact_recognition`
   Složka obsahuje skripty určené k rozpoznávání dialogových aktů.

5. `documents`
   Do této složky se ukládají PDF dokumenty, které se mají převést na obrázky.

6. `docx`
   Do složky se ukládá vytvořený MS Word dokument.

7. `text`
   Složka obsahuje zdrojové soubory LaTeX pro tento dokument.

8. `Matej_Zeman_BP_2021.pdf`
   Elektronická verze tohoto dokumentu.

DVD dále obsahuje soubory se zdrojovými kódy.

B uživatelská příručka

Pro správnou funkci projektu je třeba mít nainstalovaný `python3` a tyto knihovny:

1. python-docx
2. pdf2image
3. pytesseract
4. numpy
5. Pillow
6. opencv-python
7. imutils

Program je rozdělen do 4 částí a každá z nich plní jinou funkci.

Převod vstupních souborů do MS Word formátu

Pro převod vstupních .conll souborů do MS Word dokumentů slouží skript `DocumentWriter.py`. Pro jeho správnou funkci je třeba mít vytvořenou složku `data_conll` v adresáři se zdrojovými kódy, nebo nastavit jinou složku v parametru `conll_folder` v konfiguračním souboru. V této složce je třeba vytvořit složku s identifikací jazyka, ve kterém máme napsány naše vstupní data, například pro angličtinu je třeba založit složku `EN`. Do této složky je třeba uložit naše vstupní data rozdělená na testovací, trénovací a validační část a v konfiguračním souboru config.py nastavit jakou ze složek se vstupními daty má skript brát.

Skript se spouští příkazem `python3 DocumentWriter.py` a vytvoří ve složce docx jeden Microsoft Word dokument, dále pomocné soubory pro další funkcionalitu projektu.

Převod z MS Word formátu na obrázky

Pro převod dokumentu do obrázkových sad slouží skript `main_docimg.py`. Nejprve je nutné MS Word dokument vzniklý v předchozím kroku upravit. Pokud bychom chtěli přidávat do dokumentu pozadí, je třeba ho přidávat v aplikaci Microsoft Word a ukládat ve formě PDF dokumentů do složky documents, nebo nastavit jinou složku v parametru `document_folder` v konfiguračním souboru. Z této složky se poté vezmou všechny dokumenty a vytváří se obrázkové sady ve složkách `EN_image_dacts/xxx`, kde `xxx` představuje složku s obrázky.

Pro každý dokument jsou vytvářeny 2 datové sady. Jedna bez dalšího přidaného šumu a druhá, do které se přidává Gaussovo rozostření. Složka s obrázkovou sadou, do které bylo přidáno Gaussovo rozostření, bude mít příponu `_aug`. Rozostření lze upravovit v konfiguračním souboru `config.py` v parametru `noise_intensity`.

Skript se spouští příkazem `python3 main_docimg.py` a vytváří do příslušných složek obrázkové sady.

Tesseract OCR

Pro převod obrázkových sad do textové podoby slouží skript `main_imgtxt.py`. Skript využívá obrázkové sady vytvořené v předchozím kroku a vytváří z nich textové soubory pro klasifikátory. V souboru `config.py` je parametr `WER_CER`, který má být nastaven na `True` nebo `False` podle toho, zda chceme pro obrázkové sady počítat Word Error rate a Character Error rate. Po výpočtu, nebo přeskočení WER a CER se začnou vytvářet textové soubory test.txt, train.txt a val.txt do složky `data/xxx`, kde `xxx` představuje příslušnou složku s texty. V konfiguračním souboru je možné upravit konfiguraci systému Tesseract, nastavená konfigurace dosahovala nejlepších výsledků.

Skript se spouští příkazem `python3 main_imgtxt.py`.

Rozpoznávání dialogových aktů

Poslední část projektu zajišťující rozpoznávání dialogových aktů je prováděna skriptem `main_DArecognition.py`. V konfiguračním souboru jsou uvedeny všechny klasifikační třídy, které se v datech vyskytují. Parametr `model` v konfiguračním souboru určuje jaký model se má pro klasifikaci použít. Pro parametr je možné použít hodnotu "bilstm", "cnn1", nebo "cnn2". Jako vstupní data se berou data ze složky `data_folder`, která jde změnit v kon-

figuračním souboru. Z této složky se načtou všechny podadresáře a jejich textové soubory. V závisloti na zvoleném modelu se v podadresářích vytváří soubor s odpověďmi klasifikátoru a v adresáři se zdrojovými soubory se vytvoří soubor s výsledky klasifikace.

Skript se spouští příkazem `python3 main_DArecognition.py`.