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

Bakalářská práce

Rozpoznávání pojmenovaných entit pomocí neuronových sítí

Plzeň 2018 Martin Matas

Prohlášení

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

V Plzni dne 1. května 2018

Martin Matas

Abstract

Named Entity Recognition is a subtask of information extraction. Its goal is to recognize words with special meaning in a sentence, such as names of people, organizations, places, etc. Named Entity Recognition is often used to automatically answer questions.

The aim of this work is to explore methods used for the Named Entity Recognition, to choose one of them and together with the set of features implement the choosen solution. Subsequently test the success on standard data and evaluate the results.

Abstrakt

Rozpoznávání pojmenovaných entit je podúloha extrakce informací. Jejím cílem je rozpoznání slov ve větě se speciálním významem, např. osoby, organizace, místa atd. Rozpoznávání pojmenovaných entit se často využívá při automatickém odpovídání na otázky.

Cílem této práce je prozkoumat metody používané pro rozpoznávání pojmenovaných entit, jednu z nich vybrat a společně s množinou příznaků zvolené řešení implementovat. Následně otestovat úspěšnost na standardních datech a zhodnotit dosažené výsledky.

Obsah

1 Úvod

Zpracování pojmenovaných entit je jednou z úloh automatické extrakce informace (IE). Úkolem IE je získat strukturované informace z nestrukturovaných nebo částečně strukturovaných dat. Ve většině případů nestrukturovanými daty jsou dokumenty psané v přirozeném jazyce, v nichž chceme vyhledat důležité informace. Důležitou informací mohou být například odpovědi na otázky, kde se snažíme odhalit subjekty, které jsou součástí otázky. Ukázalo se, že tyto odpovědi na otázky mohou být snadno roztříděny do tříd na základě jejich sémantiky (tj. osoby, organizace, místa, data, času atd.) a právě tím se zabývá úloha zpracování pojmenovaných entit (NER), která se snaží nalézt a roztřídit výrazy patřící do těchto tříd (pojmenovaných entit, NEs).

Jedním ze způsobů jak efektivně řešit úlohu NER je použít neuronové sítě (NN), které se v oblasti zpracování přirozeného jazyka (NLP) využívají desítky let. NN jsou v současné době široce používány pro různé klasifikační úkoly a další úlohy spojené s umělou inteligencí. Hlavní motivy k používání NN zahrnují jejich robustnost, škálovatelnost a zejména schopnost aproximovat libovolnou funkci a zlepšovat se každým dnem, zatímco vědci stále rozvíjejí a zlepšují koncepty a učební metody těchto sítí.

V této práci se nejprve podrobněji seznámíme s úlohou NER a NN. Prozkoumáme různé druhy architektur NN a budeme diskutovat o jejich výhodách a nevýhodách pro řešení úlohy NER. Nakonec v praktické části jednu z vhodných architektur implementujeme, provedeme několik experimentů a porovnáme jejich výsledky s ostatními architekturami.

2 Specifikace úlohy

Úloha NER spočívá ve snaze detekovat a klasifikovat slovní výrazy do předem definovaných tříd. Těchto tříd je více druhů a mohou se lišit, ale velmi často se jedná o třídy jako je osoba (PER), organizace (ORG) nebo místo (LOC), atd. NER úloha může navíc obsahovat extrakci popisné informace z textu o zjištěných entitách. Například v případě osob to může zahrnovat extrakci titulu, postavení, národnosti, pohlaví a dalších atributů osoby. Níže můžete vidět příklad výsledné klasifikace viz obr. 2.1. [20, 24]

Obrázek 2.1: Příklad výstupu rozpoznávání pojmenovaných entit.

Každá NE má dvě klíčové vlastnosti: rozpětí a typ. Předpokládejme, že máme text „Bank of England“. Typ entity je v tomto případě organizace a skládá se ze tří slov. Označení „England“ s typem organizace je jistě nesprávné, neboť slovo „England“ není celé slovní spojení, stejně tak označení „Bank of England“ jako země. Označení „England“ jako země je sporné (v tomto případě) a správnost tohoto výstupu závisí na situaci. Jak vidíte, existují různé možnosti správné odpovědi. [12]

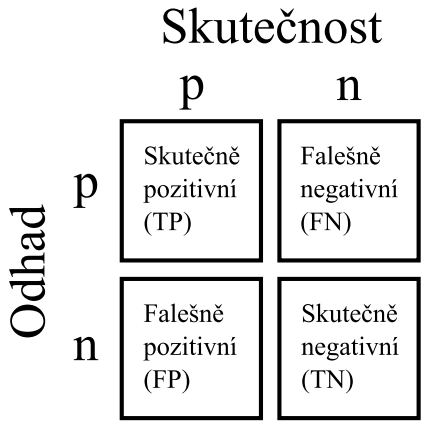
2.1 Metriky hodnocení

V každé oblasti výzkumu je důležitým krokem zhodnotit a porovnat výsledky nových metod, přičemž metriky, které zvolíte pro vyhodnocení metod, jsou velmi důležité. Výběr metriky ovlivňuje, jak se měří a porovnává výkonnost algoritmů, proto je třeba použít některou objektivní míru (nebo míry), která by dobře pokryla účel výzkumu.

Vymezíme-li míry pro obecnou klasifikaci objektů do dvou tříd: pozitivní a negativní. Pak existují čtyři třídy pro klasifikaci výsledků. Pokud systém rozpozná objekt jako pozitivní a pokud je skutečně pozitivní, pak se jedná o pozitivní rozpoznání (true positive, TP). Totéž platí pro negativní případ (true negative, TN). Pokud by však systém označil objekt jako pozitivní,

přičemž by nebyl, jedná se o falešně pozitivní rozpoznání (false positive, FP). A naopak, pokud systém objekt neoznačí, přičemž by měl, jedná se o falešně negativní chybu (false negative, FN). [12]

Jednou z nejintuitivnějších a nejsnadnějších metrik používaných k vyhodnocování úspěšnosti rozpoznávání je chybová matice. Při vyhodnocování může dojít k výskytu dvou druhů chyb (TN, FN). Tyto chyby je možné vizualizovat za pomoci chybové matice, kterou zobrazuje obr. 2.2.



Obrázek 2.2: Chybová matice zobrazující různé možnosti výstupu systému oproti skutečným hodnotám.

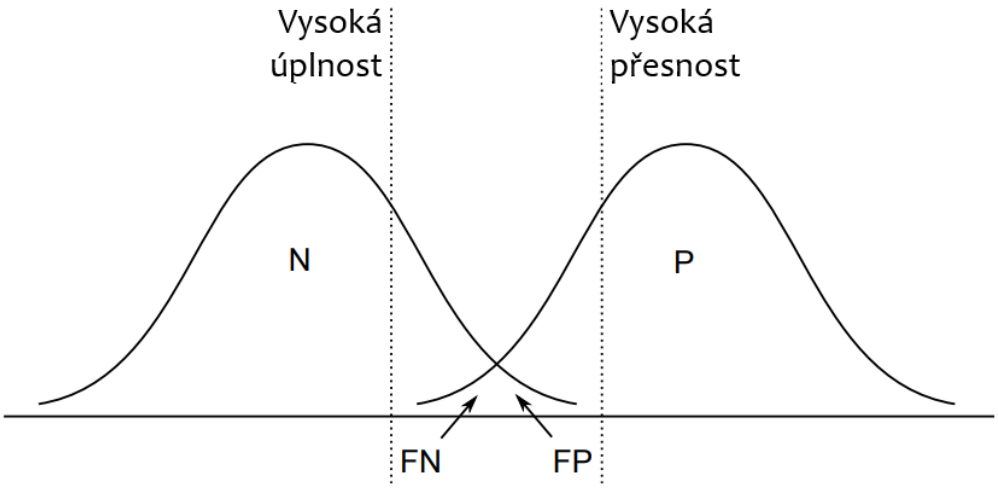
Další možnou metrikou je přesnost (precision). V klasifikačních problémech se počítá jako poměr počtu správných odpovědí a všech odpovědí, jinak řečeno, že objekty označené jako pozitivní jsou opravdu pozitivní. Vypočte se podle vzorce 2.1.

$$\text{Přesnost} = \frac{TP}{TP + FP} \tag{2.1}$$

Často se také využívá úplnost nebo-li recall, která vyjadřuje poměr jaká část pozitivních objektů je označena. Vypočte se dle vzorce 2.2.

$$\text{Úplnost} = \frac{TP}{TP + FN} \tag{2.2}$$

Je zřejmé, že přesnost a úplnost popisují různé aspekty výsledků. Navíc jsou tyto míry konkurenční. Jak je znázorněno na obrázku 2.3, pokud se prahová hodnota posune doleva, bude méně objektů FN a více objektů FP, což povede k vyšší úplnosti a nižší přesnosti. Toto je důležité při hodnocení klasifikátoru, protože klasifikátor s vysokou úplností (resp. přesností) může být lepší pro různé úkoly. [12]



Obrázek 2.3: Změna přesnosti a úplnosti

Poslední uvedenou mírou je F-míra (F-measure či F1-score). Lze ji vyjádřit jako harmonický průměr mezi přesností a úplností a představuje celkovou perspektivu. Vypočte se podle vzorce 2.3.

$$\text{F-míra} = \frac{2TP}{2TP + FP + FN} = 2\frac{\text{Přesnost} * \text{Úplnost}}{\text{Přesnost} + \text{Úplnost}} \tag{2.3}$$

Při vyhodnocování úspěšnosti rozpoznávání se přesnost (procento správně rozpoznaných slov) nepovažuje za dostatečně informující míru výkonnosti rozpoznávacího systému, jelikož většina symbolů v textu není pojmenována jako entity a jsou obvykle správně rozpoznány jako takové, přesnost snadno dosahuje 90 - 95% nebo více. [24] Namísto ní se používá F-míra.

2.2 Jazykové korpusy

Pro hodnocení NER úlohy je nutné vytvořit korpus s označenými entitami. Používáme-li přístup strojového učení s učitelem, potřebujeme také korpus pro odhad optimálních parametrů. Nejčastěji je korpus vytvořen lidmi, takový korpus se označuje termínem *gold data*. Někdy může být korpus automaticky odvozen z již existujících zdrojů. Tento přístup obvykle umožňuje vytvářet mnohem větší korpus, ale korpus obsahuje chyby. Pro tento korpus používáme termín *silver data*.

Korpus je většinou rozdělen na části s odlišným účelem využití. *Výcviková* část korpusu se používá pro učení parametrů systému a *testovací* část pouze

pro hodnocení systému. Hodnocení systému podle testovací sady během jeho vývoje je špatný postup, protože výsledky jsou pravděpodobně lepší, než by byly na ještě neviděných datech. *Ověřovací* část se používá pro ověření návrhu systému během vývoje a také pro nalezení optimálních hyperparametrů modelu (např. velikosti oken pro funkce, parametrů pro kombinaci modelů).

Dále představíme existující datové sady, které se v praxi často využívají. První dvě byly připraveny pro konferenci CoNLL v letech 2002 a 2003. Třetí, česká, byla vytvořena pro češtinu v roce 2007 [22].

2.2.1 CoNLL-2002

Datová sada pojmenovaných entit CoNLL-2002 se skládá ze šesti souborů, které pokrývají dva jazyky: španělštinu a nizozemštinu. Každý z těchto jazyků sestává ze tří souborů: výcvikového souboru, vývojového souboru a testovacího souboru. Metody učení budou trénovány na výcvikových datech a pro ladění parametrů těchto metod mohou být použita vývojová data. Po nalezení nejlepších parametrů se metody učení natrénují na základě výcvikových dat a otestují na testovacích datech. Rozdělení dat na vývojová a testovací bylo provedeno tak, aby se zabránilo ladění systému na testovacích datech. [2]

Španělská data jsou tvořena sbírkou novinových článků, které zpřístupnila španělská zpravodajská agentura EFE a anotaci provedla technická univerzita v Katalánsku společně s univerzitou v Barceloně. Nizozemské údaje byly získány ze čtyř vydání belgických novin „De Morgen“ a byly anotovány univerzitou v Antverpách. [2]

Formát dat je následující: první položkou na každém řádku je slovo, následuje slovní druh (POS), označení syntaktického bloku a druh entity. Označení bloku a názvy jmenných entit mají formát I-TYPE, což znamená, že slovo je uvnitř fráze typu TYPE. Pouze pokud se dvě fráze stejného typu vyskytují hned za sebou, první slovo druhé fráze bude mít značku B-TYPE, aby ukázalo, že začíná novou frázi. Slovo se značkou O není součástí fráze. Tento způsob značení se nazývá BIO (IOB) model. [12]

2.2.2 CoNLL-2003

Jedná se o datovou sadu skládající se ze šesti souborů pokrývajících dva jazyky: angličtinu a němčinu. Formát dat je stejný jako u CoNLL-2002 s výjimkou německých datových souborů, které mají navíc přidán jeden sloupec (druhý) obsahující lemma (slovníkový tvar) jednotlivých slov. [3]

Původ dat se u jednotlivých jazyků liší. Anglická data jsou tvořena sbírkou novinových článků od Reuters Corpus a německá data pocházejí z článků od Frankfurter Rundschau. Anotace byly provedeny osobami z univerzity v Antverpách.

2.2.3 CNEC 2.0

Pro češtinu máme dva korpusy. První Czech Named Entity Corpus (CNEC 1.0) z roku 2007 sestával z 5868 vět [22]. Druhá verze byla rozšířena o 125 vět, které obsahovaly e-mailové adresy a 3000 vět s pouze několika entitami, které modelují skutečnou distribuci entit v textech. Korpus byl později transformován do formátu podobného CoNLL (spolu s některými drobnými změnami) [13]. Celkem korpus tedy obsahuje 8993 českých vět s manuálně anotovanými českými entitami ve formátu BIO. Všechny verze korpusu jsou veřejně dostupné pro nekomerční účely. [1]

3 Algoritmy

Existují různé pohledy, které lze použít k rozdělení nebo popisu metod zabývajících se úlohou NER nebo obecně NLP úlohami. Nejčastěji jsou systémy rozděleny do dvou skupin - systémy založené na pravidlech a na strojovém učení.

Systémy založené na pravidlech fungují na základě ručně definovaných pravidel. Pravidlem může být například regulární výraz rozpoznávající konkrétní délku, malá a velká písmena, speciální znaky atd. Pro správnou funkčnost těchto systémů je často zapotřebí definovat velké množství pravidel, což je i jejich hlavní nevýhoda, neboť při každé změně charakteru dat je potřeba pravidla ručně redefinovat pro dosažení kvalitního výstupu.

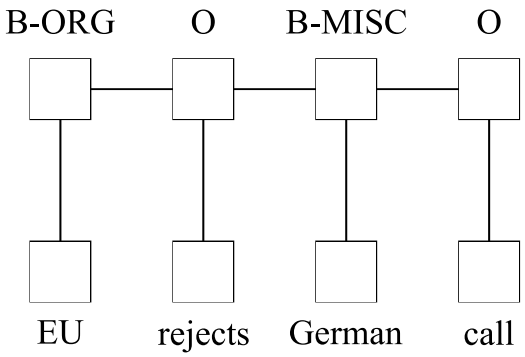
V tomto ohledu vychází lépe metody strojového učení, které se v dnešní době využívají ve většině systémů, neboť jsou velmi přizpůsobivé a na rozdíl od systémů s pravidlovým přístupem levné na údržbu.

Metody strojového učení mohou být dále rozděleny podle typu dat, které používají. Pokud systém potřebuje korpus s již označenými entitami, pak systém využívá metody učení s učitelem (supervised learning). Tyto algoritmy upravují své parametry tak, aby byly schopné správně tvořit požadované výstupy definované v korpusu. Nevýhodou tohoto postupu je nutnost doplňování požadovaných dat, což je často netriviální a časově náročné. Naproti tomu systém využívající učení bez učitele (unsupervised learning) používá samostatná data bez jakékoliv další informace o očekávaném výstupu. Na základě těchto dat si systém musí vytvořit formální reprezentaci použitelnou pro tvorbu rozhodnutí a předvídání neviděných dat. Metoda kombinující učení s učitelem a bez učitele (semi-supervised learning) je speciálním případem metody učení s učitelem. Systém může využívat korpus s označenými entitami, ale také je schopný využívat neoznačené údaje.

V následujích podkapitolách se zaměříme na metody strojového učení s učitelem pro řešení úloh NER a v sekci 3.7 na příznaky používané v NER.

3.1 Conditional random fields

Conditional random fields (CRF) jsou neorientované grafické modely viz obr. 3.1. [14] Základní myšlenkou CRF je použít řadu potenciálních funkcí



Obrázek 3.1: CRF síť.

pro aproximaci podmíněné pravděpodobnosti výstupní sekvence označení y vzhledem ke vstupní sekvenci slov x. [13, 16] Matematicky výpočet modelovaného rozdělení pravděpodobnosti $p(y|x)$ pak má následující formu:

$$p(y|x) = \frac{1}{Z(x)} exp(\sum_{j} \sum_{i} \lambda_i f_i(y_j, y_{j-1}, x, j)), \tag{3.1}$$

kde y je výstupní sekvence označení a x vstupní sekvence slov. Velikost dimenze výstupní sekvence a vstupní sekvence jsou si rovny.

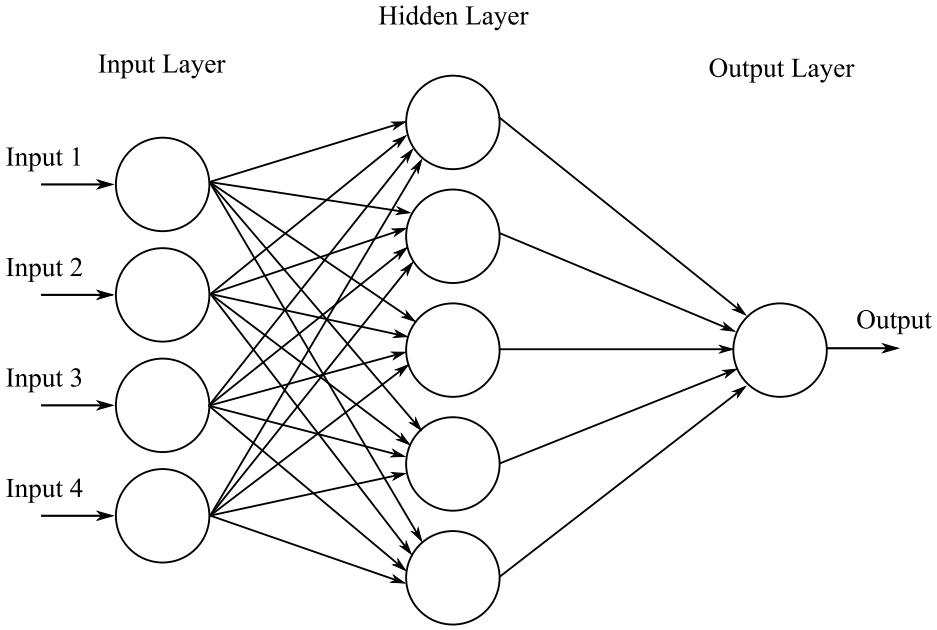
Počáteční testy zaměřené na NER úlohy byly provedeny v [18]. Od jejich zavedení je využilo mnoho systémů s velmi dobrými výsledky [10, 13] a po určitou dobu byly CRF považovány za nejúspěšnější metodu pro NER.

3.2 Dopředné neuronové sítě

Dopředná neuronová síť (také umělá neuronová síť nebo ANN) byla prvním a nejjednodušším typem navržené neuronové sítě. Jedná se o orientovaný acyklický graf, což znamená, že se v síti nevyskytují žádné zpětné vazby nebo smyčky a informace se může pohybovat jen jedním směrem. Skládá se ze vstupní vrstvy, výstupní vrstvy a skryté vrstvy. Obecně platí, že skrytých vrstev může být více. Příklad ANN můžete vidět na obrázku 3.2. [5]

3.2.1 Neuron

Základní výpočetní jednotkou neuronové sítě je neuron, který se skládá ze vstupní (přenosové) funkce, aktivační funkce a výstupu. S odkazem na obr. 3.3 je signál dat ze vstupů $x_1, ..., x_n$ považován za jednosměrný, což je indikováno šipkami, stejně jako výstupní signál neuronu. Výstupní signál neuronu O_j je dán následujícím vztahem:



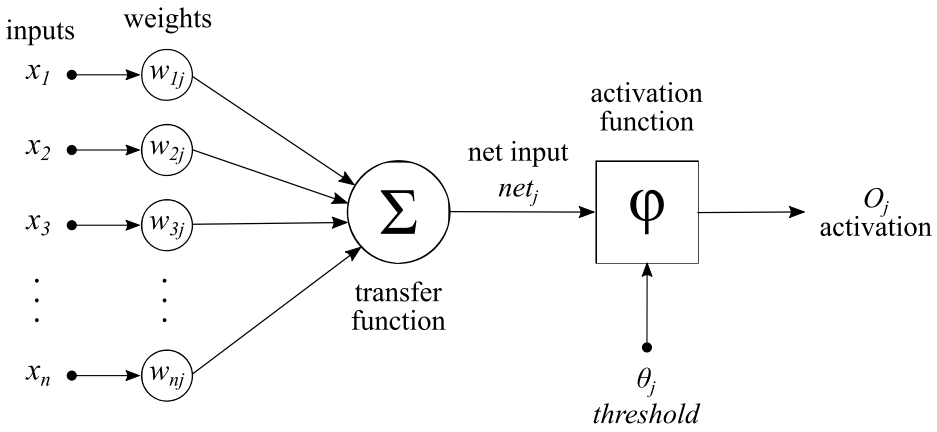
Obrázek 3.2: Umělá neuronová síť.

$$O_j = f(net_j) = f(\sum_{i=1}^{n} w_{ij} x_i), \tag{3.2}$$

kde w_{ij} je vektor váhy a funkce $f(net_j)$ je označována jako aktivační funkce. Proměnná net_j je definována jako skalární součin hmotnosti a vstupu vektorů,

$$net_j = w_j^T x = w_{1j} x_1 + ... + w_{nj} x_n, \tag{3.3}$$

kde T je transpozice matice.



Obrázek 3.3: Umělý neuron.

3.2.2 Aktivační funkce

Jednou z prvních aktivačních funkcí je kroková funkce, jejíž výstupní hodnota O_j se vypočítá jako

$$O_j = f(net_j) = \begin{cases} 1, & w_j^T x \geq 0 \\ 0, & jinak \end{cases}. \tag{3.4}$$

Kroková funkce byla použita v raných perceptronových modelech, ale nebyla spojitá, tudíž nebyla vhodná pro metody zpětné propagace založené na gradientu. Nejčastější aktivační funkcí je funkce sigmoid, známá jako logistická aktivační funkce. Používá se také ve výstupní vrstvě, kde je našim konečným cílem odhadnout pravděpodobnost. Převádí velká negativní čísla na 0 a velká kladná čísla na 1. Matematicky je reprezentována jako

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{3.5}$$

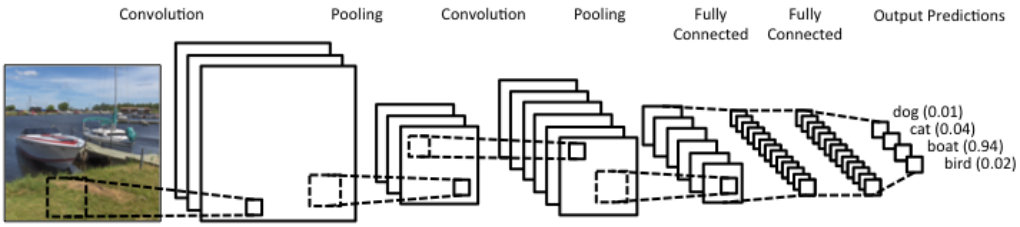
Hlavní nevýhodou sigmoidu je problém mizejícího gradientu při trénování, kdy se gradient chyb příliš přiblíží nule a nemá vliv na hlubší vrstvy. To bylo z části vyřešeno zavedením rektifikované lineární jednotky (také Rectified Linear Unit nebo ReLU) v následujícím tvaru:

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & jinak \end{cases}. \tag{3.6}$$

Tato aktivační funkce způsobí, že síť konverguje mnohem rychleji. U velkých x je derivát jednoduše 1 a chybový signál nezmizí. Existuje mnoho modifikací ReLU, jako jsou například Leaky Rectified Linear Unit (LRELU) a exponenciální lineární jednotky (ELU).

3.3 Konvoluční neuronové sítě

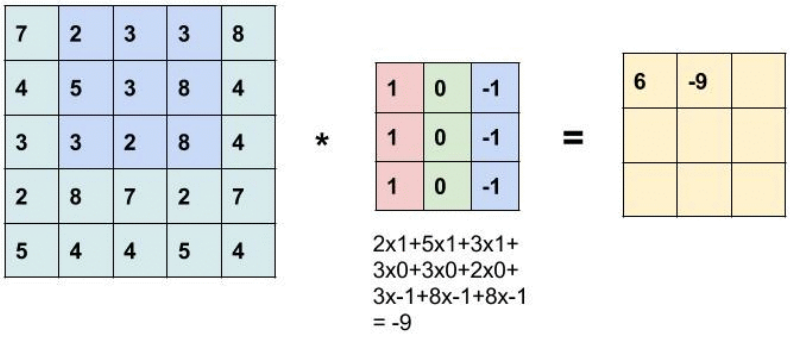
Konvoluční neuronové sítě (také Convolutional Neural Networks nebo CNN) jsou formou dopředných neuronových sítí. Níže na obrázku 3.4 je uvedeno schéma typického CNN. První část se skládá z konvolučních a poolingových (sdružovacích) vrstev, které působí jako extraktor. Druhá část se skládá z plně propojené vrstvy, která provádí nelineární transformace extrahovaných prvků a působí jako klasifikátor.



Obrázek 3.4: Schéma typické Konvoluční neuronové sítě.

3.3.1 Konvoluční vrstva

Konvoluční vrstva se skládá z několika příznakových map. Může navazovat jak na vstupní vrstvu, tak na vrstvu subsamplingovou. Neurony v této vrstvě hledají specifické rysy. Pokud tyto rysy naleznou, produkují vysokou aktivaci.



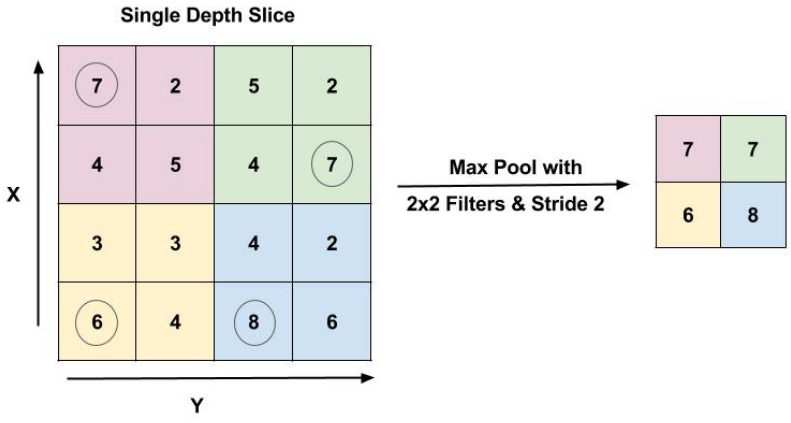
Obrázek 3.5: Příklad operace konvoluce na matici velikosti 5×5 s jádrem velikosti 3×3.

Konvoluci lze považovat za vážený součet mezi dvěma funkcemi. Při zpracování obrazu, pro výpočet konvoluce na určitém místě (x, y), vybíráme $k \times k$ rozměrný kus z obrazu centrovaného v místě (x, y). Poté vynásobíme každou hodnotu v tomto podobraze s každou hodnotou konvolučního filtru (také s rozměrem $k \times k$) a nakonec je všechny sečteme pro získání jediného výstupu. Všimněte si, že k je označeno jako velikost jádra. Příklad operace konvoluce je zobrazen na obr. 3.5.

3.3.2 Sdružovací vrstva

Sdružovací vrstva se většinou používá ihned po konvoluční vrstvě ke zmenšení prostorové velikosti (pouze šířka a výška, nikoli hloubka), čímž se snižuje počet parametrů a zabraňuje se přetrénování modelu.

Nejběžnější formou převzorkování je maximální sdružování uvedené na obr. 3.6. Postup je následující: vezmeme filtr velikosti p, kterým z částí obrázku o velikosti $p \times p$ (tedy čtverce) vybereme maximální hodnotu.



Obrázek 3.6: Příklad operace sdružování s velikostí filtru 2×2 a krokem 2. Výstupem je maximální hodnota v oblasti 2×2.

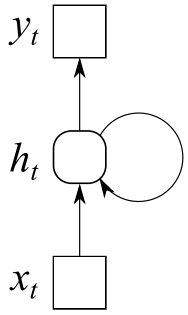
3.3.3 Plně propojená vrstva

Plně propojená vrstva znamená, že každý neuron v předchozí vrstvě je spojen s každým neuronem v další vrstvě. Výstup z konvolučních a sdružovacích vrstev představuje funkci vyššího řádu reprezentující vstupní obraz. Účelem této vrstvy je použít tyto funkce pro klasifikaci vstupního obrazu do různých tříd na základě tréninkových dat. Kromě klasifikace je přidání této vrstvy jedním ze způsobů, jak síť naučit různé nelineární kombinace těchto vlastností.

3.4 Rekurentní neuronové sítě

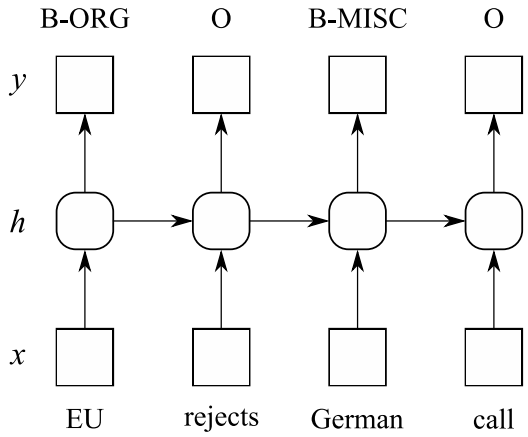
Zatímco dopředné neuronové sítě, které mapují vstupní vektory na výstupní pomocí vícevrstvých neuronových sítí, jsou nezávislé na kontextu. To zna-

mená, že stejný vstup vede vždy na stejnou odezvu sítě. Myšlenkou rekurentních neuronových sítí (Recurrent Neural Networks, RNN) je využití sekvenční informace. To je možné díky spojení mezi předchozím vnitřním stavem a aktuálním vnitřním stavem viz obr. 3.7. Jinými slovy řečeno, odezva sítě nebude dána pouze aktuálním vstupem, ale bude odrážet i vliv vstupů, které tomuto vstupu předcházely. [11]



Obrázek 3.7: Opakující se buňka v čase t odebírající vstup x_t a vracející y_t.

Pro jasnější představu na obrázku 3.8 můžete vidět jednoduchý model rekurentní neuronové sítě. Ta se skládá ze tří vrstev, vstupní vrstvy x, skryté vrstvy h a výstupní vrstvy y. V kontextu rozpoznávání pojmenovaných entit, představuje x vstupní příznaky a y značky podle BIO modelu.



Obrázek 3.8: Jednoduchý model RNN.

Jak bylo zmíněno výše, díky zavedenému spojení, umožňuje opakující se vrstva ukládání informací o historii. Hodnoty skryté a výstupní vrstvy se vypočítají podle vzorců 3.7 a 3.8.

$$h(t) = f(Ux(t) + Wh(t-1)) \tag{3.7}$$

$$y(t) = g(Vh(t)) \tag{3.8}$$

Proměnné U, W a V značí váhy spojení, které mají být vypočteny během tréninku, a $f(t)$ a $g(t)$ jsou aktivační funkce.

Teoreticky mohou RNN využívat informace v libovolně dlouhých sekvencích, ale v praxi jsou omezeny pouze na několik kroků. RNN ukázaly velký úspěch v mnoha úlohách NLP. Dnes nejčastěji používaným typem RNN jsou Long short-term memory networks (LSTM), které jsou mnohem lepší při zachování dlouhodobých závislostí než RNN. Více informací o LSTM naleznete v sekci 3.6.

3.5 Rekurzivní neuronové sítě

Dalším druhem neuronových sítí jsou rekurzivní neuronové sítě označované také jako RNN. Což může být matoucí, neboť takto se označují i rekurentní neuronové sítě, které pod tímto označením najdeme častěji. Proto, aby nedocházelo k mýlce, zůstaneme u původního celého názvu rekurzivní neuronové sítě.

Tyto sítě jsou obecnějším druhem RNN sítí umožňujících zpracovat strukturované údaje různých forem (mřížka, řetězec, graf, strom, sekvence, atd.). Například v přirozeném jazyce je běžné, že věty se stejným významem mají navzájem různou délku a strukturu. Aby byl model schopný fungovat pro různé datové struktury, rekurzivní neuronové sítě využívají sdílení parametrů, podobně jako CNN u obrázků, což jim umožňuje se zobecnit na různé formy dat, které nebyly ve cvičných datech ještě zaznamenány. Navíc objevování těchto rekurzivních struktur napomáhá nejen k identifikaci jednotlivých částí, ale i k pochopení, jak se vzájemně jednotlivé části ovlivňují, aby vytvořily celek. [23]

Matematicky vyjádřeno, přímý acyklický graf je dvojice (V, E), kde V je množina vrcholů nebo uzlů a E je množina hran mezi nimi. Pro vrchol v je $pa[v]$ množina jeho rodičů a $ch[v]$ množina jeho potomků. Graf může být použit k modelování různých druhů informací. Uzly grafu obsahují množinu doménových proměnných charakterizovaných vektorem reálných a kategorických proměnných. Navíc každý uzel kóduje část informací, o kterých se předpokládá, že hrají důležitou roli v dané úloze. Přítomnost větve (v, w) mezi dvěma uzly explicitně modeluje logický vztah mezi fragmenty informací reprezentovanými uzly v a w. [9, 19] Model rekurzivní neuronové sítě

se skládá ze stavové přechodové funkce f a výstupní funkce g daných následovně:

$$h(v^i) = f\left(\sum_{j=0}^{m} W_j x_j^i\right) \tag{3.9}$$

$$o(v^i) = g\Big(Uh(v^i)\Big), \tag{3.10}$$

kde W_j je váhová matice spojující uzel v^i s jeho m potomky a x_j^i je j-tý potomek uzlu v^i a U je váhová matice spojující skrytou aktivaci s konečným výstupem uzlu.

Funkce f obdrží jako vstup aktivace z $ch[v]$ a vynásobí každou aktivaci váhovou maticí k výpočtu aktivace pro nadřazený uzel. Funkce g označuje každý uzel v_i pomocí své skryté aktivace. Pro zpracování grafu G použijeme síť opakovaně, dokud se nedostaneme do kořenového uzlu. Uzel nemusí obsahovat vstup, ale počáteční vstup musí být v grafu znázorněn stejně tak jako v derivačních stromech vět, vstup představují listy stromu. Výstupní funkci lze použít pouze na kořenovém uzlu označujícím celý graf. [19]

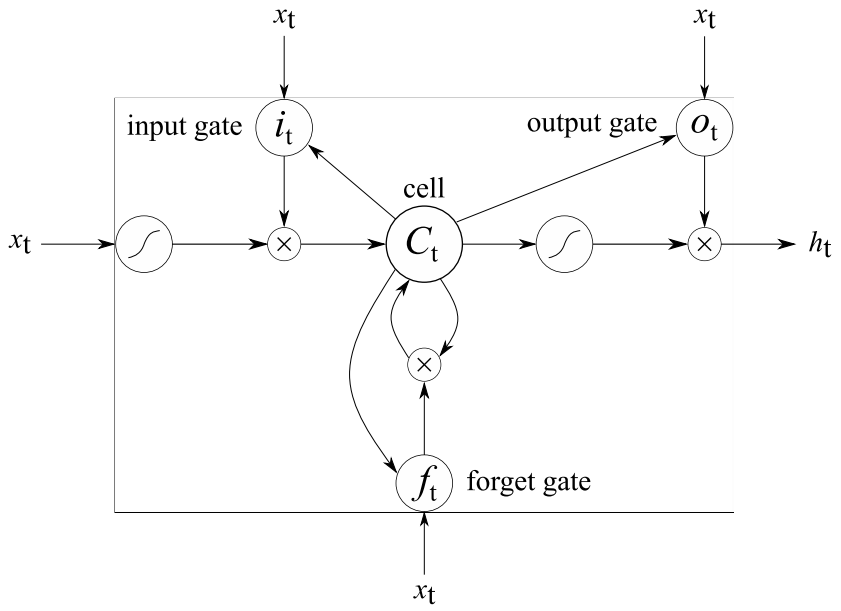
3.6 LSTM sítě

LSTM sítě jsou zvláštní druh RNN, jež jsou schopné se učit dlouhodobé závislosti. Umožňují řešit velikou škálu problémů a v dnešní době jsou široce využívány.

LSTM jsou stejné jako RNN sítě, kromě toho, že aktualizace skryté vrstvy jsou nahrazeny účelovými paměťovými buňkami. [11] Informace mohou volně procházet sítí a jsou v případě potřeby upraveny tzv. branami viz obr. 3.9, kde je znázorněna paměťová buňka LSTM.

Každá brána má svou úlohu, kterou vykonává. Vstupní brána určuje jaké hodnoty v buňce chceme aktualizovat. Další paměťová brána rozhoduje o tom, zda příchozí informaci udrží nebo zahodí. Poté přichází na řadu buňka, která má za úkol generovat nové kandidátní hodnoty. Když je proces u konce, ještě se musí rozhodnout, jaká část stavů buňky bude na výstupu. K tomu slouží výstupní brána. [4]

Sítě LSTM byly velmi úspěšné v různých úlohách, jako je strojový překlad [7], popis obrázků [26] nebo sekvenční značení v přirozeném jazyce [11]. Z tohoto důvodu jsou vhodným kandidátem pro řešení úlohy rozpoznávání pojmenovaných entit, kterou se tato práce zabývá.



Obrázek 3.9: Detail paměťové buňky LSTM.

3.6.1 Obousměrné LSTM sítě

Zatímco LSTM sítě jsou schopné ve svém skrytém stavu uchovávat informace, které již prošly skrytým stavem. Obousměrné LSTM sítě (Bidirectional LSTM, BI-LSTM) mají dva skryté stavy. Tímto způsobem můžeme efektivně využívat minulých příznaků (prostřednictvím dopředných stavů) a budoucích příznaků (prostřednictvím zpětných stavů) pro určitý časový rámec. [11]

Matematicky je vstupem do BI-LSTM vrstvy sekvence slovních příznaků (vektorů) označených jako $(x_1, x_2, ..., x_n)$. Výstupem této vrstvy je sekvence skrytých stavů pro každý vstupní vektor označených jako $(h_1, h_2, ..., h_n)$. Každý konečný skrytý stav vznikne zřetězením dopředného $\overrightarrow{h_i}$ a zpětného $\overleftarrow{h_i}$ skrytých stavů. [16]

$$\overleftarrow{h_i} = lstm(x_i, \overleftarrow{h_{i-1}}), \overrightarrow{h_i} = lstm(x_i, \overrightarrow{h_{i+1}}) \tag{3.11}$$

$$h_i = [\overleftarrow{h_i}, \overrightarrow{h_i}] \tag{3.12}$$

3.6.2 Obousměrné LSTM-CRF sítě

Obousměrné LSTM CRF sítě jsou v posledních letech velmi úspěšné a široce využívané, zejména pak v oblasti zpracování přirozeného jazyka [6, 8, 11]. Velká míra úspěšnosti je především dána kombinací LSTM sítě a CRF sítě.

Vzniklý LSTM-CRF model může efektivně využívat předešlých vstupních příznaků prostřednictvím LSTM vrstvy a infromace o značce na úrovni vět prostřednictvím CRF vrstvy. S takovou vrstvou je možné účelně předpovídat aktuální značky z předešlých a následujících značek, je to podobný princip jako u BI-LSTM sítí.

Takovýto model umožňuje využívat pouze předešlé vstupní příznaky. Obdobně jako u BI-LSTM sítí, kombinací BI-LSTM sítě a CRF sítě vznikne BI-LSTM-CRF model, který může efektivně využívat také následující vstupní příznaky.

Výkonnostní testy zaměřené na úlohy POS a NER dosáhly velmi dobrých výsledků [11]. BI-LSTM-CRF patří mezi nejpoužívanější modely pro řešení POS a NER úloh a byl použit při experimentech v kapitole 5.

3.7 Příznaky

Příznaky hrají velmi důležitou roli v NER systémech založených na metodách strojového učení, proto je velmi důležitá jejich volba. Příznak jako takový vyjadřuje konkrétní vlastnost daného slova. Například indikace přítomnosti speciálního znaku ve slově binárním příznakem (pravda či nepravda). K dosažení lepších výsledků je pak možné slovo zastoupit jedním či více příznaky, které se z hlediska strojového učení dělí na binární, kategorické, řadové, reálné, atd. [12]

Další důležitou vlastností příznaků je jazyková závislost. Platí, že příznak je nezávislý na jazyku, pokud může být použit pro jiný jazyk bez jakýchkoliv změn. Je zřejmé, že systémy NER nezávislé na jazycích potřebují používat pouze jazykově nezávislé příznaky. V opačném případě dochází k výraznému poklesu výkonnosti NER systému.

3.7.1 Znakové příznaky

Ortografické příznaky

Ortografické příznaky jsou založeny na vzhledu slova, např. první písmeno je velké, všechna písmena jsou velká, slovo se skládá z číslic nebo slovo obsahuje speciální znaky, atd. Tyto příznaky jsou velmi často používané, protože si vystačí se samostatným slovem a jsou jazykově nezávislé a přesto velmi účinné pro mnoho jazyků.

Ortografické vzory

Tyto vzory využívají podobný přístup jako ortografické příznaky - přepisují slova na speciální řetězce. Velká písmena jsou přepsána na 'A', malá písmena na 'a', číslice na '1' atd. Občas se využívá zkrácená verze, kdy více znaků stejného typu ('aaaa') je přepsáno na kratší formu ('a*' nebo 'aa'), která znázorňuje více znaků stejného typu.

3.7.2 Slovní příznaky

Slovo

I samotné slovo je možné použít jako příznak. Často se však převedou všechny znaky na malé nebo velké, aby se zamezilo duplicitám a slovo na začátku věty mělo stejný příznak jako to uprostřed.

Stematizace a Lematizace

Stematizace je úloha založená na hledání kořene slova, obvykle odstraněním sémanticky nepotřebných koncových znaků. Obdobně jako samotné slovo i kořen slova může být použit jako příznak.

Lematizace je podobná úloha jako stematizace, ale namísto kořene je výstupem základní (slovníkový) tvar slova.

Význam těchto úloh je do velké míry ovlivněn jazykem. U flektivních jazyků vyznačujících se komplikovaným systémem skloňování a časování, jako je např. čeština, je stematizace a lematizace téměř nutností, neboť je potřeba snížit vysoký počet různých tvarů slov.

N-gramy

Slovní n-gramy (sled n po sobě jdoucích slov) se používají k zachycení kontextu současného slova. Sled dvou po sobě jdoucích slov bývá často označován jako *bigram*, sled tří slov je *trigram* a od čtyř výše se používá označení *n-gram*, kde n je nahrazeno počtem za sebou jdoucích slov.

Embeddings

V dnešních systémech využívajících RNN sítě pro zpracování přirozeného jazyka se častěji místo slovních příznaků využívají tzv. embeddingy, kde jsou slova nebo fráze ze slovníku mapovány na vektory reálných čísel. Vektor má stejnou velikost jako je velikost slovníku a je tvořen jednou dimenzí. Tato technika se nazývá vkládání slov (word embedding) a může výrazně zvýšit

výkon NLP úloh. Existují různé metody vkládání slov, mezi ty nejznámější patří GloVe a Word2Vec. [21]

3.7.3 Externí příznaky

Pojem externí příznak se používá k označení příznaků, které používají externí systém nebo zdroj informací (např. Wikipedia). Existuje zvláštní podtřída externích příznaků, která se nazývá *slovník*. Tato skupina je často zpracovávána zvlášť, protože hraje důležitou roli v NER. [12]

Slovníky

Slovníky mají v úloze NER zvláštní postavení. Používáme slovníky např. křestních jmen a příjmení, firem, měst, řek, zemí, společností atd. Systémy využívající slovníkové příznaky ztrácejí možnost přímého použití v jiných jazycích nebo dokonce v doménách, protože očividně nespadají mezi jazykově nezávislé příznaky. Postup je velmi prostý, testuje se, zda je slovo obsaženo v některém ze slovníků.

4 Realizace

Cílem této práce je implementace vlastních příznaků do NER systému. Jako základ pro tuto práci byl zvolen NER systém založený na modelu BI-LSTM-CRF sítě. Vzhledem k tomu, že zmíněný model je implementačně netriviální a navíc by zabralo spoustu času a úsilí takovýto model řádně implementovat a optimalizovat, je na místě použít některou z již hotových implementací, která je uživateli ověřená a optimalizovaná.

4.1 Existující implementace

Výběr vhodného systému je nutné provést ještě před zahájením implementace, protože se může stát, že zvolený systém nemusí umožňovat přidání vlastních příznaků.

Hlavním rozhodovacím faktorem při výběru byla knihovna, na jejíchž základech je aplikace postavena. Vedoucím práce byla doporučena knihovna Torch[1]. Přestože v dnešní době existuje nespočet implementací modelů neuronových sítí, najít vyhovující implementaci se nepodařilo. Po další domluvě se do užšího výběru dostalo jen několik implementací napsaných v jazyce Python.

V následujících podsekcích si postupně představíme jednotlivé implementace, jejich výhody a nevýhody a srovnáme výsledky testovacích běhů.

4.1.1 AnaGo

Implementace AnaGo slouží pro sekvenční značení textu, umožňuje rozpoznávání pojmenovaných entit (NER), značení úseků řeči (POS tagging) a sémantické značení (SRL). Je postavena na knihovně Keras[2], což je vysokoúrovňové API pro neuronové sítě fungující jako obal knihoven, např. TensorFlow či Theano.

Systém využívá BI-LSTM-CRF model založený na architektuře pro NER [15] podporující následující možnosti:

- trénování modelu bez příznaků
- definování vlastního modelu

[1] http://torch.ch/
[2] https://keras.io/

- použití už natrénovaného modelu

Implementace systému je jazykově nezávislá, tudíž je možné úlohy řešit v mnoha jazycích. AnaGo dále také podporuje předem natrénované příznaky jako jsou vektory GloVe nebo Word2Vec.

Přes všechny zmíněné výhody implementace trpí na podporované modely vstupních dat. Zatímco s BIO modelem nebyl žádný problém, IO model implementace nebyla schopna zpracovat. Dalším úskalím bylo velké množství podpůrných programů, které bylo třeba nainstalovat, což v prostředí jako je MetaCentrum nebylo vůbec snadné, o tom ale až v podkapitole 4.5.

4.1.2 NER Tagger

Další vybranou implementací byl NER Tagger, sloužící přímo pro rozpoznávání pojmenovaných entit. Využívá nejnovější nápady a metody z oblasti NER bez využití jakékoli jazykově specifické znalosti nebo zdroje (např. slovníků). Výsledkem je možnost řešit úlohy v mnoha jazycích.

Stejně jako AnaGo, tento systém využívá BI-LSTM-CRF model uvedený v [15]. Implementace je postavena na knihovně Theano[3], která společně s NumPy[4] slouží pro vědecké výpočty.

Už v základu NER Tagger podporuje IO i BIO modely značení a možnost konfigurace programu prostřednictvím parametrů. Kvalitu implementace nekazí ani dosažené výsledky uvedené v tabulce 4.1, přesto má implementace jednu velkou nevýhodu, a to dobu běhu. Ta byla u některých jazyků (španělština, nizozemština) o poznání horší. Zatímco ostatní systémy byly schopné během 2 až 3 hodin provést 15 epoch, kdy jedna epocha znamená jeden dopředný průchod a jeden zpětný průchod všech výcvikových příkladů, tento systém provedl sotva polovinu, tedy 7 epoch.

4.1.3 NER with Tensorflow

Implementovaný systém využívá BI-LSTM-CRF model založený na knihovně TensorFlow[5]. Mezi hlavní výhody systému patří podpora IO i BIO modelů značení, minimální požadavky na potřebné knihovny a přiložená podrobná dokumentace[6]. Tím však výhody končí. Přestože implementace dosáhla na anglickém korpusu velmi dobrých výsledků, viz tab. 4.1, nebyla schopna se

[3]http://deeplearning.net/software/theano/
[4]http://www.numpy.org/
[5]https://www.tensorflow.org/
[6]https://guillaumegenthial.github.io/sequence-tagging-with-tensorflow.html

vypořádat s českými a španělskými znaky, např. ě, š, ř, atd. Tento problém nevyřešila ani oprava kódování.

4.1.4 LM-LSTM-CRF

Poslední zvolenou implementací je nástroj pro sekvenční značení, umožňující trénování, ohodnocení a predikci. Nabízí dva implementované modely LM-LSTM-CRF a BI-LSTM-CRF uvedené v [17]. Pro naše účely byl zvolen, již několikrát zmíněný, BI-LSTM-CRF model, jenž je implementován prostřednictvím knihovny PyTorch[7].

Implementovaný systém splňuje všechny potřebné požadavky, je jazykově nezávislý, podporuje modely značení IO a BIO, je konfigurovatelný prostřednictvím parametrů, podporuje natrénované příznaky (GloVe, Word2Vec) a v neposlední řadě je přiložena i kompletní programová dokumentace. Kromě toho knihovna PyTorch podporuje výpočty na grafických kartách s architekturou CUDA[8]. V praxi se to projeví výrazným urychlením výpočtů.

4.2 Porovnání implementací

Po vybrání vhodných implementací je bylo nutné na závěr zprovoznit a podrobit výkonnostním testům. Implementace byly testovány celkem na čtyřech jazykových korpusech (angličtina, čeština, španělština a nizozemština). V některých případech právě až toto testování odhalilo pravé nedostatky, jako je podpora formátů anotace či doba běhu viz. AnaGo, NER Tagger. V následující tabulce si můžete prohlédnout, jak dopadly výkonnostní testy jednotlivých implementací.

	eng	cze	esp	ned
AnaGo	-	**70.54**	**81.90**	75.62
NER Tagger	88.30	69.35	78.58	**75.94**
NER with Tensorflow	**89.34**	-	-	-
LM-LSTM-CRF	87.41	66.62	75.43	71.38

Tabulka 4.1: Srovnání testovaných implementací. Uvedené výsledky udávají míru výkonnosti rozpoznávacích systémů pomocí harmonického průměru (F-míry), která je vyjádřena v procentech.

Z měření je patrné, že nejlepších výsledků dosahovaly implementace AnaGo a NER Tagger. O to větší je škoda, že zrovna tyto implementace dispo-

[7] http://pytorch.org/

[8] https://developer.nvidia.com/cuda-zone

nují výše zmíněnými nedostatky. Po zvážení těchto problémů a možnostech opravy, byla vybrána implementace LM-LSTM-CRF pro realizaci experimentů s vlastními příznaky.

4.3 Struktura aplikace

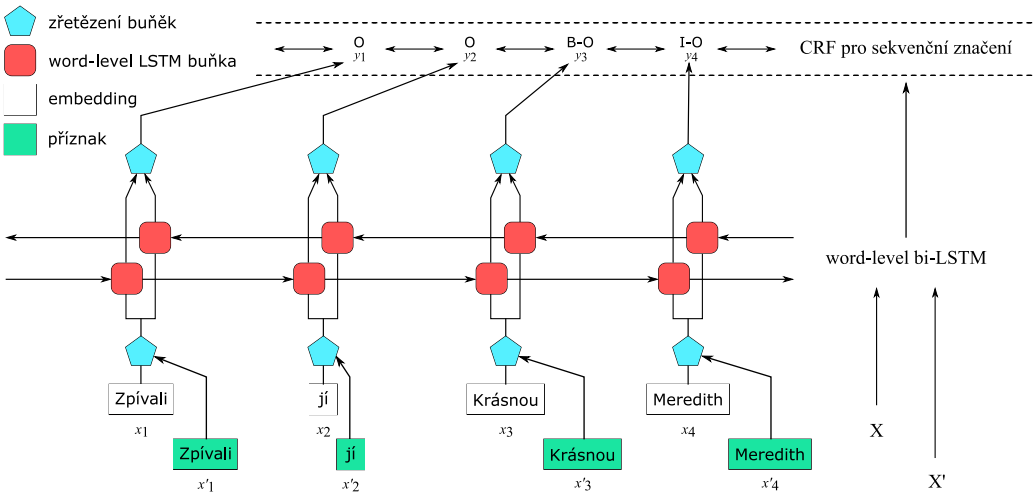
Po vybrání vhodné implementace můžeme přistoupit k popisu jednotlivých modulů a skriptů, ze kterých aplikace sestává. Detailní popis všech modulů a jejich procedur je popsán v programové dokumentaci přiložené u implementace[9] a samozřejmě v kódu programu.

V základu aplikace obsahovala dva implementované modely LM-LSTM-CRF a BI-LSTM-CRF. Pro účely této práce postačoval BI-LSTM-CRF model (viz obr. 4.1), který byl méně komplexní, protože využíval pouze slovní embeddingy na rozdíl od LM-LSTM-CRF, jenž používal i znakové embeddingy. Moduly a skripty související s nepoužitým modelem byly odstraněny. Následná struktura aplikace vypadala takto:

- **model** - tento balík sestává z následujících modulů obsahujících algoritmy sloužící pro chod NER systému, metriky pro hodnocení a pomocné nástroje
  - **crf** - modul obsahuje implementaci CRF vrstvy se všemi potřebnými procedurami
  - **evaluator** - modul pro hodnocení, poskytuje procedury pro výpočet F-míry a přesnosti
  - **features** - jedná se o nový modul, který byl do systému dodělán v rámci této práce a zahrnuje implementace všech použitých příznaků, jejichž konkrétní popis naleznete v podkapitole 4.4
  - **lstm_crf** - procedury v tomto modulu umožňují vytvářet a pracovat s LSTM-CRF modelem
  - **ner_dataset** - modul obsahuje třídy pro vytváření souborů dat využitých při trénování modelů
  - **predictor** - modul slouží pro predikci, zahrnuje procedury pro výpočet F-míry a přesnosti
  - **utils** - obsahuje širokou škálu procedur, např. pro načítání korpusů, předpřipravení dat pro CRF vrstvu, mapování slov na vektory, atd.

[9] http://lm-lstm-crf.readthedocs.io/en/latest/

- **train_w.py** - skript sloužící pro trénování modelu, obsahuje všechny nezbytné procedury uvedené v modulech modelu
- **eval_w.py** - tento skript se využívá pro hodnocení již existujících modelů, aby se ušetřil čas strávený trénováním modelu
- **seq_w.py** - skript se využívá pro anotaci neoznačených textů, kdy vstupem jsou pouze slova a výstupem je text označený entitami



Obrázek 4.1: Neuronová architektura BI-LSTM-CRF.

Aplikaci je možné spustit ve třech režimech: trénování modelu (soubor `train_w.py`), ohodnocení existujícího modelu (`eval_w.py`) a anotování neoznačených textů (`seq_w.py`). Pro implementaci vlastních příznaků a následné experimenty byl využíván pouze trénovací režim. K tomu určený skript využívá pro trénování modelu architekturu uvedenou na obrázku 4.1.

Při spuštění dojde nejprve k načtení navolených korpusů. Korpusy jsou navzájem odděleny, každý korpus se skládá ze seznamu slov $X = (x_1, ..., x_n)$ a k nim odpovídajících anotací $Y = (y_1, ..., y_n)$ uložených v seznamu anotací. Aby bylo možné se slovy pracovat, jsou vytvořeny vektorové reprezentace jednotlivých slov x_i. Pro ulehčení práce se využívají předem natrénované slovní embeddingy GloVe.

Z takto vytvořených vektorů se společně s anotacemi vytvoří soubor dat, jenž se bude následně po malých dávkách posílat na vstup LSTM-CRF modelu. Podle toho, jak velká bude jedna dávka, se určí počet trénovacích iterací jedné epochy. Nakonec se před ukončením každé epochy ještě provede vyhodnocení aktuální dosažené úspěšnosti klasifikace.

V rámci této práce bylo potřeba do systému doimplementovat vlastní způsob reprezentace slov. K tomuto účelu slouží modul `features`. Příznaky implementované v tomto modulu jsou podrobně popsány v následující sekci. Výstupem modulu je seznam slovních příznaků X' vytvořený podle seznamu vět X. Vytvořené příznaky se poté přidají do souboru dat společně s embeddingy a anotacemi. Tyto příznaky jsou v obrázku 4.1 označeny zelenou barvou a před vstupem do LSMT vrstvy jsou obě slovní reprezentace zřetězeny dohromady.

4.4 Použité příznaky

Jak již bylo několikrát zdůrazněno, výběr příznaků výrazně ovlivňuje učení klasifikátoru a tím i celkový výsledek klasifikace.

Příznak jako takový vyjadřuje určitou vlastnost daného slova. V praxi jsou pak tyto příznaky reprezentovány vektorem, který má na každé pozici definovaný specifický příznak (vlastnost). Implicitně jsou hodnoty vektoru nastaveny na hodnotu 0 (nepravda). Na základě určité vlastnosti se poté na vybraných pozicích hodnota změní na 1 (pravda).

Následuje výběr příznaků, které byly vybrány, implementovány a následně otestovány při experimentech na popsaných datech.

4.4.1 Ortografická podoba slova

Při hledání způsobů, jak co nejlépe rozpoznat pojmenovanou entitu, bude ortografická (*ortografie* - též *pravopis*, je souhrn pravidel o používání písmen a diakritických znamének při zaznamenávání jazykových projevů) podoba slova pravděpodobně jednou z prvních myšlenek. Například vlastní jména, začátky vět či geografické názvy začínají velkým písmenem. Existují však výjimky, jako jsou názvy některých firem či organizací, které používají velká písmena jinde než na první pozici, a mnoho dalších.

Nejdůležitější roli u tohoto příznaku hraje vzor. Aby bylo možné slova efektivně porovnávat, je nejlepší vytvořit seznam hned několika vzorů. K tomuto účelu se přímo nabízí využít regulárních výrazů, které umožňují definovat vzory s vysokou mírou složitosti. Dalším způsobem je možnost transformace slov do zjednodušené podoby, k čemuž je zapotřebí sada přepisovacích pravidel, a následné porovnání s předem definovanými vzory.

Při implementaci byly použity oba způsoby. První způsob je jednoduchý, nejprve se s využitím regulárních výrazů nadefinují různé vzory. V našem případě byly vytvořeny následující vzory detekující:

- velké počáteční písmeno
- kombinaci velkých a malých písmen
- přítomnost číslic
- slova psaná kapitálkami
- přítomnost speciálních znaků
- kombinaci číslic, malých a velkých znaků

Vektor vygenerovaný tímto způsobem má délku danou počtem definovaných vzorů, což je v tomto případě 6. Jak bylo zmíněno v úvodní části, prvky vektoru, které odpovídají vzoru popisujícímu dané slovo, mají hodnotu nastavenou na 1 a neshodující se prvky na 0. Například, budeme-li mít slovo „YouTube“, tak podle výše uvedených příznaků bude vygenerovaný vektor vypadat následovně $[1, 1, 0, 0, 0, 0]$.

Druhý způsob je velmi podobný. Nejprve je nutné nadefinovat seznam vzorů. Ve srovnání s prvním způsobem se seznam vzorů definuje automaticky předzpracováním vstupních dat. Jednotlivá slova obsažená ve vstupních datech se transformují do zjednodušené podoby a vytvářejí seznam unikátních vzorů seřazených sestupně podle četnosti. Následně se vezme n horních vzorů, které poslouží pro generování vektorů.

Postup transformace je následující, velká písmena jsou přepsána na 'A', malá písmena na 'a', číslice na '1' a ostatní (speciální) znaky na '-'. Aby se zabránilo odlišnostem vzniklým kvůli různým délkám sekvencí znaků se stejnou vlastností, využívá se zkrácená verze, kdy více znaků stejného typu 'aaaaa' je nahrazeno kratší formou 'aa'.

Dále už je postup stejný, generují se vektory délky n. Prvky vektoru, které odpovídají vzoru, jsou nastaveny na hodnotu 1 a neodpovídající na hodnotu 0. Pro ukázku, řetězec „YouTube“ bude po transformaci vypadat takto „AaaAaa“, řetězec „iPhone“ takto „aAaa“. Všimněte si, že zatímco předešlý způsob mohl mít více shod, tento způsob bude mít vždy pouze jednu, protože seznam obsahuje unikátní vzory. Může se zdát, že je tento způsob neefektivní, ale při experimentech dokázal úspěšnost systému pozitivně ovlivnit.

4.4.2 Přítomnost slova ve slovníku

Jedná se o další příklad příznaku aplikovaného při implementaci. Využívá slovníky obsahující pojmenované entity, v podsekci 3.7.3 již bylo naznačeno

jeho použití. Slovníky byly dodány vedoucím práce, celkem se jedná o 23 slovníků různých druhů entit, od názvů měst, dnů, organizací, až po akademické tituly či televizní stanice.

Použití je prosté. Nejprve se načtou všechny slovníky do struktury *dict*, která zajistí, že se uvnitř nebudou vyskytovat duplicitní hodnoty. Generovaný vektor délkou odpovídá počtu používaných slovníků. Hodnota 1 je nastavena u prvku, který odpovídá slovníku, ve kterém bylo slovo nalezeno.

Toto řešení umožňuje snadno identifikovat časté entity (jména, organizace, svátky atd.). V českém jazyce však narážíme na problém, který je pro flektivní (*flexe* - též *ohýbání*, tj. skloňování a časování) jazyk typický. V běžném textu se slova vyskytují různě vyskloňovaná, přičemž slovo v jakémkoliv jiném tvaru než kořenovém, je ve slovníku nemožné najít. Řešením může být využití regulárních výrazů, nicméně to může mít neblahý vliv na výkon systému. Lepším a snazším řešením je ostemování entit ve slovnících i slov v nich vyhledávaných.

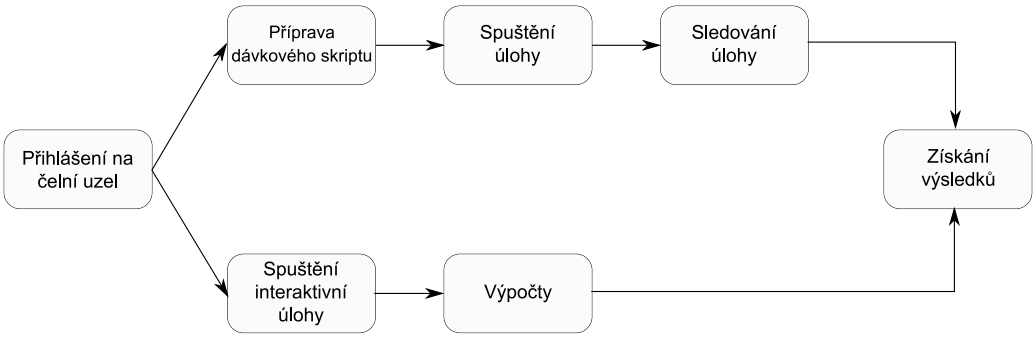
Obě varianty, jak původní data, tak i ostemovaná data, byly zahrnuty do experimentů pro demonstraci vlivu stematizace na úspěšnost klasifikace.

4.5 MetaCentrum

Velmi důležitou roli hraje také volba stroje, na kterém budou NER úlohy spouštěny, protože dnešní systémy k provozu potřebují poměrně výkonný hardware. Náročnost těchto systému je dána především potřebou implementovat netriviální modely neuronových sítí a jiné osvědčené metody, které mají velký vliv na kvalitu dosažených výsledků. Například implementace LM-LSTM-CRF, použitá při experimentech, potřebuje pro svůj běh alespoň 8 GB paměti RAM a pro urychlení výpočtů ještě grafickou kartu NVIDIA s architekturou CUDA.

Je zřejmé, že provádět jakékoliv výpočty tohoto typu, např. na noteboocích, je zcela nemožné. Proto bylo nutné nalézt alternativní řešení, které by nabízelo dostatečně výkonný hardware a ideálně neomezený přístup ke stroji. První variantou bylo využít univerzitní počítač, ale po většinu dne v učebnách probíhá výuka a počítače jsou tedy zabrané. Řešením bylo využití služeb MetaCentra, konkrétně tzv. „gridových výpočtů“. To je způsob provádění výpočtu, při kterém je pro dosažení výsledku použito více počítačů nacházejících se na různých místech.

Před samotným zahájením výpočtů je zapotřebí zažádat o vytvoření účtu. Jakmile je účet schválen, je možné začít s výpočty. Gridové výpočty mají specifický postup počítání viz obr. 4.2.



Obrázek 4.2: Schéma gridového počítání v MetaCentru.

Ze všeho nejdřív je potřeba se přihlásit na jeden z tzv. „čelních uzlů“. Jedná se o stroje, které zprostředkovávají přístup k síti MetaCentra a jsou sdíleny všemi uživateli. Přihlašuje se k nim prostřednictvím protokolu SSH. Uživatel má na čelním uzlu vlastní domovský adresář a přímý přístup k datovým úložištím pro manipulaci s velkými objemy dat. Nicméně tyto uzly slouží pouze pro manipulaci s daty, přípravu spouštěcích skriptů a zadávání úloh.

Pro potřeby provádět náročnější operace, které mohou zpomalovat čelní uzel a znepříjemňovat tak jeho používání ostatním uživatelům (například rozbalování nebo zabalování velkého množství dat, kompilace programů nebo dokonce jejich spouštění), slouží výpočetní uzly.

Pro přístup k těmto uzlům je nutné zadat úlohu a zařadit se do fronty čekajících. Úlohy se zadávají prostřednictvím plánovacího systému PBS Professional příkazem `qsub`. Společně s úlohou je potřeba specifikovat požadované zdroje (počet procesorů, množství paměti RAM či počet grafických karet) a odhadovanou dobu běhu. Dále může být uveden dávkový skript, parametry skriptu a mnoho dalších nastavení. Následující příklad demonstruje zadání úlohy pro trénování NER klasifikátoru na anglickém korpusu.

```
qsub
-l select=1:ncpus=1:ngpus=1:mem=8gb:cl_zubat=True
-q gpu -l walltime=24:00:00
-m ae -M martinm@students.zcu.cz
-v type="none" run_eng.sh
```

Takto zadaná úloha se zařadí do fronty úloh a čeká na vykonání. Jak je patrné z uvedeného schématu viz obr. 4.2, úlohy se dělí do dvou skupin - interaktivní úlohy a dávkové úlohy.

Interaktivní úlohy

Jedná se o speciální typ dávkové úlohy, kde se pracuje s přidělenými zdroji místo používání dávkových skriptů. Úlohu je možné vyžádat pomocí příkazu `qsub` s přepínačem `-I`.

Využívání této úlohy přináší oproti dávkovým úlohám jednu velkou výhodu, uživatel má k dispozici grafické prostředí a snadno vidí, co dělá. Z tohoto důvodu se tento typ úlohy využíval při experimentech pro ladění dávkových skriptů a k opravě chyb v implementaci NER systému.

Dávkové úlohy

Dávkové úlohy jsou neinteraktivní typ úloh. Pro jejich běh je potřeba mít připravený dávkový skript (=spouštěcí skript). Ten obsahuje posloupnost příkazů v určeném pořadí, které jsou vykonávány vyžádanými zdroji. Příklad spouštěcího skriptu, jenž se využíval při experimentech pro trénování NER systému, naleznete v příloze A. Připravený skript se následně přidá na konec příkazu `qsub` a úloha se zadá. Oproti interaktivním úlohám, zde se musí čekat až úloha doběhne. Stav úlohy je možné sledovat příkazem `qsub -u username`. Jakmile úloha skončí, výsledky úlohy se uloží do nastaveného adresáře.

5 Experimenty

Pro zjištění úspěšnosti implementovaných příznaků bylo provedeno několik experimentů. Úspěšnost trénovaného systému byla měřena metrikami uvedenými v části 2.1. Hlavní sledovanou mírou při ladění příznaků a finálním hodnocení byla F-míra.

Trénování a ohodnocení modelu probíhalo na třech datových sadách: CoNLL-2002, CoNLL-2003 a CNEC 2.0. Tyto datové sady byly dodány vedoucím práce. Jedná se o korpusy pro češtinu, angličtinu, španělštinu a nizozemštinu. Každý obsahuje tři soubory: trénovací, vývojový a testovací.

Níže předložené výsledky zachycují měření prováděná na jazykových korpusech CNEC 2.0 a CoNLL-2003. Pro jednotlivá měření byly nejprve použity jednotlivé modifikace samostatně. Poté následují experimenty vybraných kombinací příznaků a statistika úspěšnosti finálního systému, ve kterých byly zahrnuty všechny tři jazykové sady.

5.1 Původní řešení

Základem pro hodnocení úspěšnosti byla převzatá původní implementace NER modelu sestávající z jedné obousměrné LSTM vrstvy a CRF vrstvy. Velikost skrytého stavu byla nastavena na 300, velikost trénovací dávky na 10 a počet trénovacích epoch na 30. Pro inicializaci byly zvoleny předtrénované embeddingy GloVe velikosti 300.

Naměřené výsledky uvedené v tab. 5.1 nedopadly nejlépe pro český korpus. Důvodem byl vysoký počet různých tvarů slov vzniklých skloňováním a časováním. V porovnání s českým korpusem dopadl výrazně lépe anglický korpus, který se s tímto problémem nepotýkal.

jazykový korpus	**f-míra (%)**	**úplnost (%)**	**přesnost (%)**
CNEC 2.0 (čeština)	66.62	60.12	90.64
CoNLL-2003 (angličtina)	87.41	85.80	97.12

Tabulka 5.1: Výsledky naměřené při trénování původního NER modelu pro český a anglický korpus.

5.2 Navázání dalších příznaků

Tyto experimenty měly za cíl navázat příznaky na jednotlivé slovní embeddingy a vylepšit tak reprezentaci slova.

5.2.1 Ortografická podoba slova

Jedná se o rozpoznávání vizuálních rysů jednotlivých slov. Byly implementovány dva způsoby reprezentace, které jsou podrobně popsány v podsekci 4.4.1. První z nich hledal ve slově jednoduché vzory, např. velké počáteční písmeno, číslice, speciální znaky, atd. Naměřené výsledky byly znatelně vyšší než u původního řešení. Největší nárůst byl zaznamenán především u F-míry.

Jazykový korpus	**F-míra (%)**	**Úplnost (%)**	**Přesnost (%)**
CNEC 2.0 (čeština)	72.92	71.82	93.02
CoNLL-2003 (angličtina)	88.90	88.63	97.70

Tabulka 5.2: Výsledky měření jednoduchých vzorů rozpoznávajících vizuální rysy slov.

Druhý způsob transformoval slova do zjednodušeného tvaru, ze kterého následně vytvářel seznam nejčastějších tvarů. Jednotlivá slova poté porovnal se seznamem a hledal shody.

Výsledky měření byly velmi podobné prvnímu způsobu, český korpus dopadl v měření lépe než v předešlém experimentu, naopak anglický korpus měl výsledky nepatrně horší viz tab. 5.3.

Jazykový korpus	**F-míra (%)**	**Úplnost (%)**	**Přesnost (%)**
CNEC 2.0 (čeština)	73.45	68.78	92.91
CoNLL-2003 (angličtina)	88.61	87.35	97.58

Tabulka 5.3: Výsledky měření ortografických příznaků, které využívají seznam nejčastějších tvarů slov.

5.2.2 Přítomnost slova ve slovníku

Další experiment byl založen na hledání slova ve slovnících pojmenovaných entit. Slovníky byly dodány vedoucím práce. Jednalo se o 23 českých slovníků, z nichž každý obsahoval jeden druh entity (např. jména žen, společností, měst, atd.). Z důvodu jazykové závislosti bylo provedeno měření

pouze na českém korpusu. Provádět měření na ostatních korpusech by nemělo smysl, protože by bylo nalezeno jen malé procento shod, které by se projevilo dramatickým poklesem celkové úspěšnosti klasifikace.

Při měření bylo dosaženo jen nepatrného zlepšení F-míry oproti původní implementaci. Důvodem je již zmíněný problém s vysokým počtem různých tvarů slov vzniklých skloňováním a časováním. Nejhůře dopadla úplnost, která klesla pod hodnotu naměřenou při původním řešení.

Jazykový korpus	**F-míra (%)**	**Úplnost (%)**	**Přesnost (%)**
CNEC 2.0 (čeština)	67.80	59.61	91.00

Tabulka 5.4: Naměřené hodnoty při použití slovníkových příznaků.

5.3 Stematizace

Při tomto experimentu byly ostemovány vstupní datové soubory (pro jednotlivá slova byl nalezen jejich kořen). V důsledku toho se razantně redukoval počet různých tvarů slov a docházelo častěji ke shodě než u vyskloňovaných a vyčasovaných slov.

Pro demonstraci účinku byl test proveden s využitím původní implementace. Získané výsledky dosáhly výrazného zlepšení na českém korpusu, naopak k velkému propadu došlo u anglického korpusu, kterému zjevně stemování vůbec neprospělo.

Jazykový korpus	**F-míra (%)**	**Úplnost (%)**	**Přesnost (%)**
CNEC 2.0 (čeština)	69.66	64.11	91.49
CoNLL-2003 (angličtina)	82.14	79.11	96.09

Tabulka 5.5: Výsledky měření ostemovaných datových sad při použití původní implementace.

5.4 Srovnání příznaků

Následující výsledky zachycují měření provedená na jednotlivých datových sadách (jednotlivých jazycích) za použití kombinace implementovaných příznaků. Dále jsou uvedeny výsledky s využitím ostemovaných datových sad a ostemovaných slovníků, které se využívaly v případě externích příznaků.

Z důvodu dlouhých čekacích front na MetaCentru nebyly následující finální výsledky počítány na grafických kartách, ale jen s využitím procesorů.

5.4.1 Čeština (CNEC 2.0)

V tomto experimentu bylo provedeno trénování NER modelů s využitím různých kombinací příznaků. Jako vstupní data byl použit český datový korpus CNEC 2.0.

Druh příznaku	**F-míra (%)**	**Trvání (h)**
none	66.62	10.49
dictionary	67.66	8.07
orthographicPatterns	72.32	10.71
orthographicPatterns + dictionary	73.42	8.68
dictionary + orthographicPatterns	73.44	9.67
orthographicFeatures + dictionary	73.88	8.53
orthographicFeatures	74.44	11.01
orthographicFeatures + orthographicPatterns	74.74	8.57
orthographicPatterns + orthographicFeatures	74.96	8.58
dictionary + orthographicFeatures	75.59	8.56

Tabulka 5.6: Srovnání výsledků dosažených různou kombinací příznaků při trénování modelů na českém korpusu.

Výsledky uvedené v tabulce 5.6 potvrdily vysokou míru úspěšnosti u příznaků založených na ortografické podobě slova, především pak kombinaci obou způsobů. Nejlepšího výsledku nakonec dosáhly překvapivě slovníkové příznaky v kombinaci s ortografickými.

Pro dosažení ještě vyšších výsledků byly vstupní datové soubory a slovníky ostemovány. Výsledkem bylo ještě drobné vylepšení, které mělo vliv na všechny testované případy. Následující naměřené hodnoty v tabulce 5.7 potvrzují důležitost stemování u ohebných jazyků, mezi které patří i čeština.

Na rozdíl od předchozích experimentů se zde naplno projevil význam slovníkových příznaků, které jasně dominují na nejvyšších příčkách v kombinaci s ortografickými příznaky.

5.4.2 Angličtina (CoNLL-2003)

Experiment kombinoval různé druhy příznaků pro trénování NER modelů, přičemž jako vstup byl použit anglický korpus, který je součástí CoNLL-2003. Do experimentu nebyly zahrnuty možnosti se slovníkovými příznaky, protože byly k dispozici pouze české slovníky.

Tento experiment dosáhl při měření nejlepších výsledků, které byly zaznamenány. Dosažené výsledky jsou uvedeny v tabulce 5.8.

Druh příznaku	F-míra (%)	Trvání (h)
none	69.27	14.07
dictionary	70.58	14.37
orthographicPatterns + dictionary	75.73	11.84
orthographicPatterns	75.93	11.02
orthographicFeatures + orthographicPatterns	76.38	10.36
orthographicFeatures	76.64	14.5
orthographicPatterns + orthographicFeatures	76.68	12.27
dictionary + orthographicFeatures	76.79	14.57
dictionary + orthographicPatterns	77.01	11.86
orthographicFeatures + dictionary	77.16	14.15

Tabulka 5.7: Srovnání výsledků dosažených stemováním a různou kombinací příznaků při trénování modelů na českém korpusu.

Druh příznaku	F-míra (%)	Trvání (h)
none	87.41	17.17
orthographicPatterns	88.80	17.34
orthographicFeatures	89.17	17.21
orthographicPatterns + orthographicFeatures	89.62	15.67
orthographicFeatures + orthographicPatterns	89.92	17.33

Tabulka 5.8: Naměřené hodnoty udávající úspěšnost klasifikace a dobu trénování systémů na anglickém korpusu.

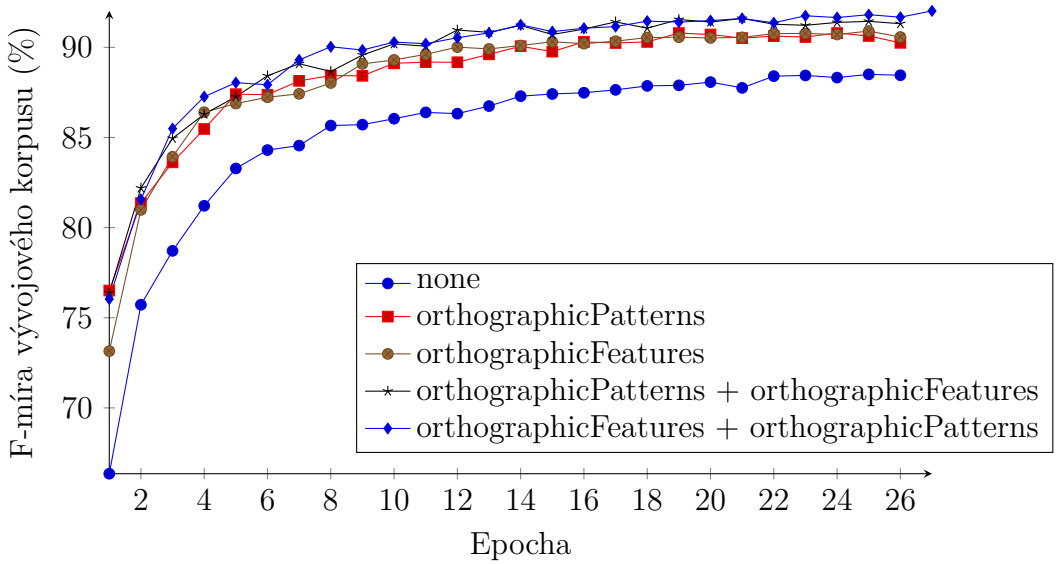
Experiment s ostemovanými korpusy už dosažené výsledky dále nezlepšil, naopak došlo k viditelnému poklesu v úspěšnosti klasifikace viz tab. 5.9. Není však zcela jisté, zda mělo stemování negativní vliv na naměřené hodnoty, protože úlohy byly ukončeny předčasně z důvodu vyčerpání (maximálního možného) vymezeného času. Na následujícím grafu 5.1 je znázorněn průběh trénování NER modelů, na kterém je možné vidět kromě úspěšnosti v jednotlivých epochách také epochu, ve které byly jednotlivé úlohy přerušeny (standardně trénování končí po 30 epochách).

5.4.3 Španělština (CoNLL-2002)

Další experiment byl proveden na španělském korpusu z datové sady CoNLL-2002. Implementované příznaky pozitivně ovlivnily dosažené výsledky viz tab. 5.10. Podle naměřených hodnot úspěšnosti se může zdát, že jednotlivá měření dosáhla podobných výsledků, nicméně doba trénování nasvědčuje tomu, že byly úlohy předem ukončeny. To potvrzuje i průběh trénovaných

Druh příznaku	F-míra (%)	Trvání (h)
none	81.68	23.96
orthographicPatterns	85.02	23.96
orthographicFeatures	85.41	23.96
orthographicPatterns + orthographicFeatures	85.86	23.96
orthographicFeatures + orthographicPatterns	86.2	23.96

Tabulka 5.9: Úspěšnost použitých příznaků při trénování modelů na ostemovaném anglickém korpusu.

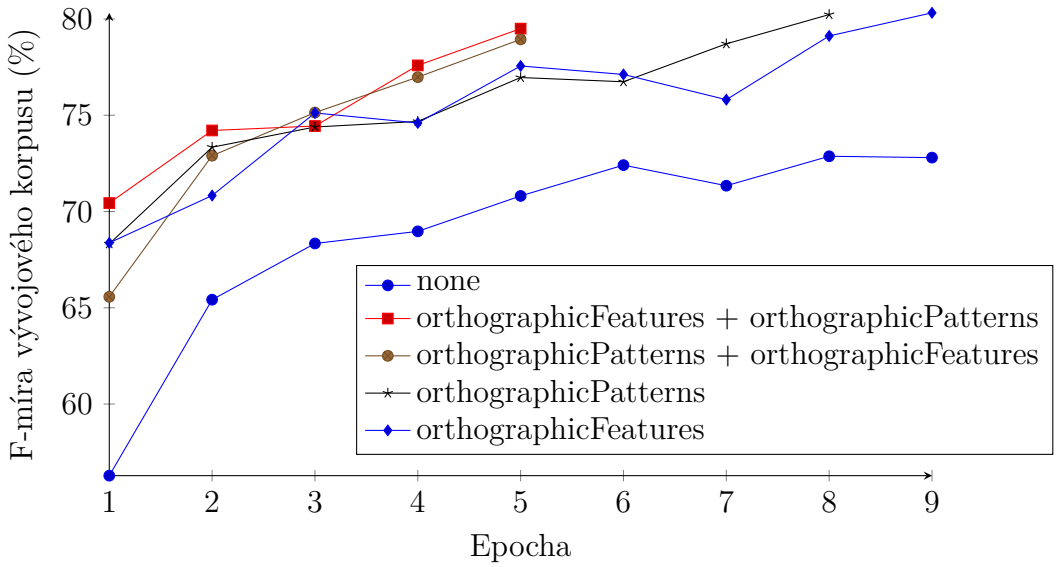


Obrázek 5.1: Průběh trénování modelů znázorňující konvergenci k výsledné úspěšnosti. Uvedené výsledky byly měřeny na ostemovaném anglickém korpusu určeném pro ladění parametrů systému (nejedná se o testovací korpus).

modelů uvedený v grafu 5.2. Z průběhu je patrné, že úlohy skončily v rané fázi trénování, ve které výslednou úspěšnost trénovaných modelů ještě nelze zcela jistě určit.

Druh příznaku	F-míra (%)	Trvání (h)
none	75.43	23.97
orthographicFeatures + orthographicPatterns	80.96	23.95
orthographicPatterns + orthographicFeatures	81.38	23.96
orthographicPatterns	81.72	23.98
orthographicFeatures	82.24	23.98

Tabulka 5.10: Výsledky trénování NER modelů na španělském korpusu.



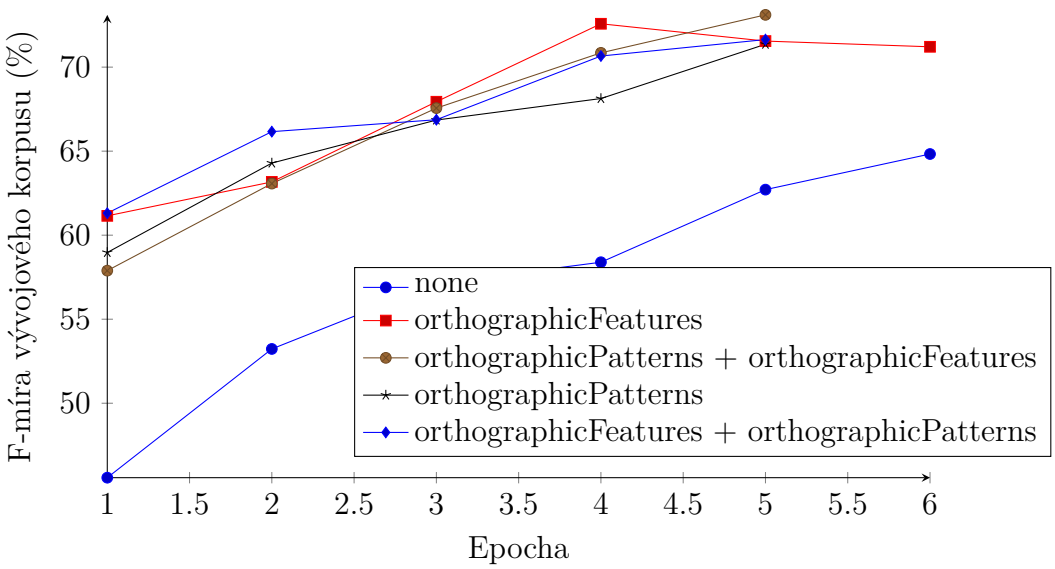
Obrázek 5.2: Průběh trénování modelů na španělském korpusu určeném pro ladění sytému (nejedná se o testovací korpus).

Druhé měření bylo provedeno na ostemovaném španělském korpusu. Stejně jako při prvním experimentu byl problém s předčasně ukončenými úlohami. Naměřené výsledky jsou o něco horší, než při využití neostemovaného korpusu, nicméně oproti původní implementaci přináší lehké zlepšení viz. tab 5.11. Z průběhu trénování modelů uvedeném v grafu 5.3 je zjevné, že v nadpoloviční většině případů úloha skončila v páté iteraci, zbytek úloh v šesté. Z tohoto důvodu není opět možné zcela jistě určit úspěšnost systému.

Při bližším prozkoumání se jako nejpravděpodobnější důvod pomalého trénování modelů jevila velikost trénovacího korpusu. V případě CNEC 2.0 korpus obsahoval 166 tisíc záznamů a doba trénování se pohybovala v průměru kolem 9 hodin, zatímco trénovací korpus datové sady CoNLL-2002 obsahoval v průměru 240 tisíc záznamů a trénování modelu trvalo maximální povolenou dobu, tedy 24 hodin, a přesto nebyl model zcela natrénován.

Druh příznaku	**F-míra (%)**	**Trvání (h)**
none	69.8	23.95
orthographicFeatures	75.48	23.95
orthographicPatterns + orthographicFeatures	76.44	23.95
orthographicPatterns	77.13	23.96
orthographicFeatures + orthographicPatterns	78.42	23.94

Tabulka 5.11: Výsledky trénování NER modelů na ostemovaném španělském korpusu.



Obrázek 5.3: Úspěšnost modelů při trénování na ostemovaném španělském korpusu, který je určen k optimalizaci modelu.

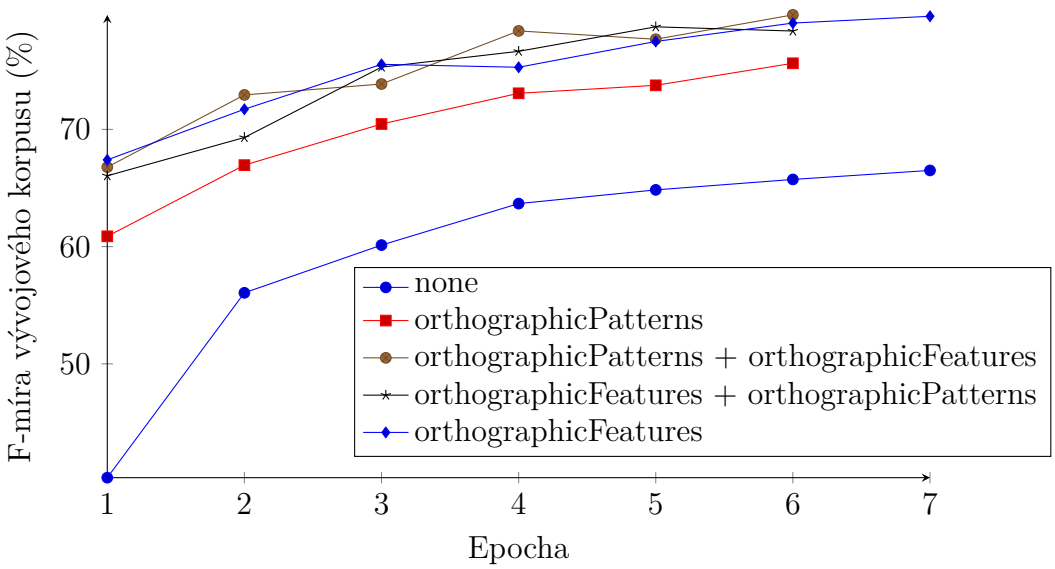
5.4.4 Nizozemština (CoNLL-2002)

Poslední experiment byl proveden na nizozemském datovém korpusu pocházejícím z datové sady CoNLL-2002. V tabulce 5.12 jsou uvedeny nejlepší dosažené výsledky jakých modely během experimentu dosáhly. Oproti původní implementaci bylo vidět znatelné zlepšení. Doba trénování u jednotlivých modelů se blíží jednomu dni, což nasvědčuje tomu, že úlohám vypršel vymezený čas. Toto tvrzení je možné ověřit z grafu 5.4, odkud je patrné, že modely nebyly trénovány 30 epoch, ale jen 6 až 7 epoch. Uvedené výsledky v tabulce proto nelze považovat za výslednou úspěšnost modelů.

Druh příznaku	**F-míra (%)**	**Trvání (h)**
none	71.38	23.96
orthographicPatterns	76.09	23.95
orthographicPatterns + orthographicFeatures	78.35	23.95
orthographicFeatures + orthographicPatterns	79.31	23.95
orthographicFeatures	79.7	23.96

Tabulka 5.12: Uvedené výsledky udávají úspěšnost klasifikace a dobu trénování jednotlivých NER modelů na nizozemském testovacím korpusu.

Následně bylo provedeno ještě měření úspěšnosti klasifikace s použitím ostemovaného nizozemského korpusu. Dosažené výsledky s využitím stemo-



Obrázek 5.4: Průběh trénování NER modelů na nizozemském korpusu sloužícímu k ladění modelu.

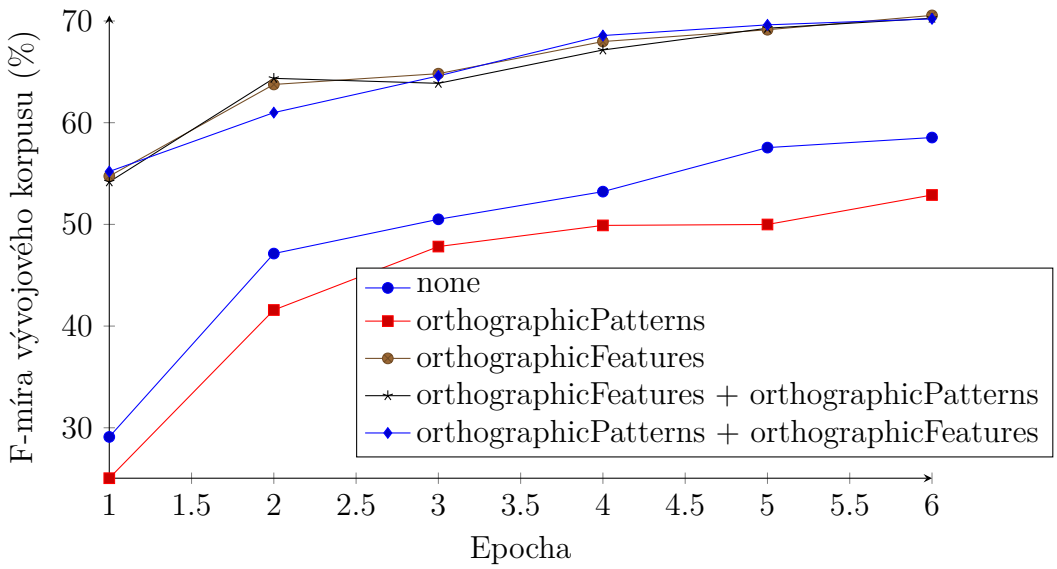
vání nepřinesly žádné zlepšení, naopak, podle naměřených výsledků byla úspěšnost o něco nižší viz tab. 5.13. Implementované příznaky pozitivně ovlivnily úspěšnost jednotlivých modelů. Dá se předpokládat, že by úspěšnost mohla ještě vzrůst, protože podle trvání úloh došlo pravděpodobně k předčasnému ukončení. Tuto domněnku potvrzuje graf 5.5, ze kterého je patrné, že úlohy skončily v šesté epoše. Uvedené úspěšnosti tak nelze považovat za výsledné, neboť modely byly trénovány jen na několika epochách.

Druh příznaku	**F-míra (%)**	**Trvání (h)**
none	62.82	23.96
orthographicPatterns	71.23	23.96
orthographicFeatures	72.09	23.95
orthographicFeatures + orthographicPatterns	73.13	23.96
orthographicPatterns + orthographicFeatures	73.29	23.96

Tabulka 5.13: Úspěšnost použitých příznaků při trénování modelů na ostemovaném nizozemském korpusu.

5.5 Zhodnocení výsledků

Na základě dosažených výsledků lze říci, že implementované příznaky měly velký vliv na výslednou úspěšnost modelů. Na neostemovaných korpusech



Obrázek 5.5: Průběh trénování modelů na nizozemském korpusu určeném pro ladění sytému (nejedná se o testovací korpus).

bylo průměrné zlepšení o 6.7 % u češtiny a o 2 % u angličtiny, na ostemovaných korpusech pak o 6.6 % u češtiny a o 3.9 % u angličtiny. Zbylé jazyky (španělština a nizozemština) nebyly dostatečně natrénovány, aby bylo možné z výsledků udělat závěr.

U českého korpusu hrála zásadní roli také stematizace. Výsledná nejlepší úspěšnost 77.16 % dosažená stemováním byla v porovnání s neostemovaným korpusem o 1.57 % lepší. U ostatních korpusů měla stematizace spíše opačný vliv na úspěšnost modelů.

Obecně nejlepších výsledků dosahovaly kombinace ortografických příznaků, u českého korpusu pak kombinace slovníkových příznaků s ortografickými. Úplně nejlepšího výsledku bylo dosaženo na neostemovaném anglickém korpusu s kombinací ortografických příznaků, kdy byla naměřena úspěšnost 89.92 %. V následující tabulce 5.14 bylo provedeno srovnání výsledků s ostatními systémy.

	en (%)	**es (%)**	**nl (%)**	**cz (%)**
Lample a kolektiv [15]	**90.94**	85.75	**81.74**	-
Straková a kolektiv [25]	89.16	-	-	**82.82**
Konkol a Konopík [13]	83.24	81.39	75.97	74.08
tato práce	89.92	**86.20**	79.70	77.16

Tabulka 5.14: Srovnání dosažených výsledků.

6 Závěr

Cílem této práce bylo seznámit se s problematikou rozpoznávání pojmenovaných entit a prozkoumat vhodné metody pro řešení této problematiky, zejména pak metody založené na strojovém učení. Dalším úkolem bylo zvolit jednu z těchto technik společně s množinou příznaků a vybrané řešení implementovat. Nakonec byl vytvořený systém natrénován na standardních datech a proběhlo vyhodnocení dosažených výsledků.

Teoretická část tohoto textu se zabývala popisem problematiky rozpoznávání pojmenovaných entit, metrikami pro výpočet hodnocení systémů a používanými daty, jež se využívají pro trénování, optimalizaci a testování systémů. Dále podrobně rozebírá techniky, které jsou pro řešení této problematiky využívány a způsoby, jak lze reprezentovat vlastnosti zpracovávaných slov.

Součástí praktické části byla analýza již existujících systémů pro rozpoznávání pojmenovaných entit s následným detailním popisem zvolené implementace. Nechybí popis implementovaných příznaků a prostředí, ve kterém se systém spouštěl. Nakonec je uveden popis experimentů, dosažené výsledky a jejich zhodnocení.

Implementované příznaky dosáhly na standardizovaných datech slibných výsledků. Případná navazující práce by mohla spočívat v optimalizaci úspěšnosti implementací dalších příznaků či úpravou těch stávajících. Od věci není ani možnost optimalizace trénovacího mechanismu.

Nejproblematičtější a časově velmi náročnou částí práce bylo zprovoznění existujících implementací v prostředí MetaCentra, především kvůli rozdílným požadavkům jednotlivých systémů na knihovny a komplikovanému způsobu jejich přidání.

Seznam zkratek

ANN	Artificial Neural Networks
BI-LSTM	Bidirectional Long-Short Term Memory
CNEC	Czech Named Entity Corpus
CNN	Convolutional Neural Networks
CoNLL	Conference of Computational Natural Language Learning
CRF	Conditional Random Fields
IE	Information Extraction
LSTM	Long-Short Term Memory
NE	Named Entity
NER	Named Entity Recognition
NLP	Natural Language Processing
NN	Neural Networks
RNN	Recurrent Neural Networks
SSH	Secure Shell

Literatura

[1] *Czech Named Entity Corpus 2.0* [online]. Dostupné z: https://ufal.mff.cuni.cz/cnec/cnec2.0.

[2] *The CoNLL-2002 shared task data files* [online]. Dostupné z: https://www.clips.uantwerpen.be/conll2002/ner/.

[3] *The CoNLL-2003 shared task data files* [online]. Dostupné z: https://www.clips.uantwerpen.be/conll2003/ner/.

[4] *Understanding LSTMs* [online]. Dostupné z: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[5] Abraham, A. *Artificial Neural Networks*. American Cancer Society, 2005. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471497398.mm421. ISBN 9780471497394.

[6] Anh, L. T. – Arkhipov, M. Y. – Burtsev, M. S. Application of a Hybrid Bi-LSTM-CRF model to the task of Russian Named Entity Recognition. *CoRR*. 2017.

[7] Bahdanau, D. – Cho, K. – Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv*. 2014.

[8] Chalapathy, R. – Zare Borzeshi, E. – Piccardi, M. Bidirectional LSTM-CRF for Clinical Concept Extraction. In *Proceedings of the Clinical Natural Language Processing Workshop (ClinicalNLP)*. The COLING 2016 Organizing Committee, 2016. Dostupné z: http://www.aclweb.org/anthology/W16-4202.

[9] Chinea, A. Understanding the Principles of Recursive Neural Networks: A Generative Approach to Tackle Model Complexity. In Alippi, C. et al. (Ed.) *Artificial Neural Networks – ICANN 2009*. Springer Berlin Heidelberg, 2009.

[10] Georgiev, G. et al. Feature-Rich Named Entity Recognition for Bulgarian Using Conditional Random Fields. 2009.

[11] Huang, Z. – Xu, W. – Yu, K. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR*. 2015, abs/1508.01991.

[12] Konkol, M. *Rozpoznávání pojmenovaných entit.* PhD thesis, Západočeská univerzita v Plzni, 2016. Dostupné z: http://hdl.handle.net/11025/23711.

[13] Konkol, M. – Konopík, M. CRF-based Czech Named Entity Recognizer and Consolidation of Czech NER Research. In Habernal, I. – Matoušek, V. (Ed.) *Text, Speech and Dialogue*, 8082 / *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013.

[14] Lafferty, J. D. – McCallum, 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*. Morgan Kaufmann Publishers Inc., 2001. ISBN 1-55860-778-1.

[15] Lample, G. et al. Neural Architectures for Named Entity Recognition. In Knight, K. – Nenkova, A. – Rambow, O. (Ed.) *HLT-NAACL*. The Association for Computational Linguistics, 2016. ISBN 978-1-941643-91-4.

[16] Lin, B. Y. et al. Multi-channel BiLSTM-CRF Model for Emerging Named Entity Recognition in Social Media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Association for Computational Linguistics, 2017.

[17] Liu, L. et al. Empower Sequence Labeling with Task-Aware Neural Language Model. *CoRR*. 2017.

[18] McCallum, A. – Li, W. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. Association for Computational Linguistics, 2003.

[19] Nomer, H. A. A. Recursive Neural Networks Review. 2016.

[20] Piskorski, J. – Yangarber, R. *Information Extraction: Past, Present and Future*. Theory and Applications of Natural Language Processing. Springer Berlin Heidelberg, 2013. Dostupné z: http://dx.doi.org/10.1007/978-3-642-28569-1_2. ISBN 978-3-642-28568-4.

[21] Seok, M. et al. Named Entity Recognition using Word Embedding as a Feature. 2016.

[22] Ševčíková, M. – Žabokrtský, Z. – Krůza, O. Named Entities in Czech: Annotating Data and Developing NE Tagger. In Matoušek, V. – Mautner, P. (Ed.) *Text, Speech and Dialogue*. Springer Berlin Heidelberg, 2007.

[23] Socher, R. et al. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2011.

[24] Straková, R. J. *Rozpoznávání pojmenovaných entit pomocí neuronových sítí.* PhD thesis, Matematicko-fyzikální fakulta Univerzity Karlovy, 2017.

[25] Straková, J. – Straka, M. – Hajic, J. A New State-of-The-Art Czech Named Entity Recognizer. In *TSD*, 2013.

[26] Wang, C. et al. *Image Captioning with Deep Bidirectional LSTMs.* ACM, 2016. ISBN 978-1-4503-3603-1.

A Dávkový skript pro trénování NER systému

```
#!/bin/bash

# nastavení proměnné domovského adresáře
DATADIR="/storage/plzen1/home/$LOGNAME/LM-LSTM-CRF-master/"

# vstoupí do uživatelského adresáře
cd $DATADIR || exit 1

# potřebné moduly pro spuštění
module add python27-modules-gcc
module add python-2.7.6-gcc python-2.7.6-intel
module add cuda-7.5 cudnn-6.0

# nastavení proměnných pro příkaz pip
export PYTHONUSERBASE=/storage/plzen1/home/martinm/.local
export PATH=$PYTHONUSERBASE/bin:$PATH
export PYTHONPATH=$PYTHONUSERBASE/lib/python2.7/site-packages:$PYTHONPATH

# instalace potřebných závislostí
pip install -r requirements.txt --user --process-dependency-links

# nastavení úklidu SCRATCHE v případě chyby
trap 'clean_scratch' TERM EXIT

# trénování NER modelu
python ./train_w.py \
        --train_file ../data/eng/eng.train \
        --dev_file ../data/eng/eng.testa \
        --test_file ../data/eng/eng.testb \
        --emb_file ../data/glove.6B/glove.6B.300d.txt \
        --epoch 30 --caseless \
        --fine_tune --embedding_dim 300 \
        --hidden 300 --feature_modif $type
```

B Přílohy na DVD

Na přiloženém DVD se nachází tato práce, návod pro spuštění, zdrojové kódy NER systému, dosažené výsledky a data pro otestování.

Z důvodu autorských práv nejsou uvedeny datové sady CoNLL-2002 a CoNLL-2003, je uveden pouze český korpus CNEC 2.0, který je dostupný pod licencí CC BY-NC-SA 3.0 a byl pro účely této práce upraven.

- *bp_src* - Zdrojový kód této práce.
- *cze_dictionaries* - Slovníky pojmenovaných entit.
- *cze_dictionaries_stemmed* - Ostemované slovníky pojm. entit.
- *data* - Data nezbytná pro trénování modelu.
- *results* - Dosažené výsledky při experimentech.
- *src* - Zdrojové kódy NER systému.
- *BP-Matas.pdf* - Elektronická verze této práce.
- *metacentrum_prikazy.txt* - Příklady spuštění systému na MetaCentru.
- *pc_prikazy.txt* - Příklady spuštění systému na počítači.
- *README.txt* - Návod pro spuštění systému.

C Uživatelský manuál

C.1 Systémové požadavky

Pro spuštění je zapotřebí operačního systému *Linux* a knihoven *python 2.7.13* a *pip 9.0.1*. Vzhledem k náročnosti na operační paměť je doporučeno spouštět systém na stroji alespoň s 8 GB RAM.

C.2 Spuštění aplikace

Aplikaci je možné spustit dvěma způsoby: na počítači nebo prostřednictvím gridových výpočtů na MetaCentru. Oba způsoby jsou podrobně popsány v přiloženém souboru *README.txt*. Ke spuštění dochází z adresáře *src* pomocí spouštěcích skriptů s prefixem *run_*.

Příklad spuštění na počítači:

```
./run_cze_pc.sh [--type="dictionary"]
```

Příklad spuštění na MetaCentru:

```
qsub \
-l select=1:ncpus=1:mem=8gb:scratch_local=10gb \
-l walltime=16:00:00 \
-v type="dictionary" \
run_cze_metacentrum.sh
```