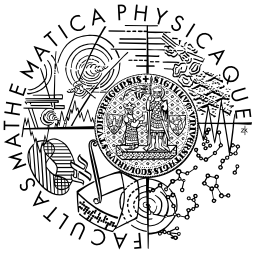


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

**DIPLOMOVÁ PRÁCE**

Bc. Martin Mareš

# Pseudonymizace textových datových kolekcí pro strojové učení

Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: doc. RNDr. Ondřej Bojar, Ph.D.
Studijní program: Informatika
Studijní obor: Umělá inteligence

Praha 2021

Prohlašuji, že jsem tuto diplomovou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne .....  .....

Podpis autora

Chtěl bych poděkovat doc. RNDr. Ondřeji Bojarovi, Ph.D. za cenné rady, trpělivost a čas, který mi věnoval po celou dobu při vedení této práce. Děkuji své rodině za dlouhodobou nekončící podporu při studiu a kolegům z práce za trpělivost a velkou podporu při psaní práce.

Název práce: Pseudonymizace textových datových kolekcí pro strojové učení

Autor: Bc. Martin Mareš

Ústav: Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: doc. RNDr. Ondřej Bojar, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: Textové datové kolekce vytvářejí prostor pro nové úlohy využívající algoritmy z umělé inteligence. Tyto kolekce často obsahují různé osobní údaje a jiné citlivé informace, které komplikují jejich sdílení a další zpracování kvůli požadavkům na ochranu osobních údajů. Hledání osobních údajů je často řešeno pouze postupným procházením celého textu. Práce si proto klade za cíl vytvořit nástroj, který napomůže anotátorům snižovat riziko úniku osobních údajů z textových datových kolekcí. Nástroj pro snížení rizika využívá pseudonymizace (tj. nahrazování slov jinými slovy pomocí nějakého klíče). V průběhu anotačního procesu nástroj automaticky označuje slova jako „veřejná“, „soukromá“ a jako „podezřelá“. Úkolem anotátora je rozhodovat o „podezřelých slovech“ a dohledávat případné chybějící neoznačené citlivé informace. „Soukromá“ slova jsou poté předmětem procesu pseudonymizace. Nástroj k automatickému označovaní využívá rozpoznávač pojmenovaných entit a databázi pravidel. Databáze pravidel se sama průběžně vylepšuje při některých rozhodnutích anotátora. V rámci práce došlo k porovnání různých rozpoznávačů pojmenovaných entit pro účel vyhledávání osobních údajů na kolekci z projektu ELITR. Při porovnávání byla nalezena metoda, která zvýšila citlivost detekce pojmenovaných entit a tím i zvýšila citlivost detekce „podezřelých slov“. Součástí práce je úvod do právní problematiky ochrany osobních údajů při vytváření textových kolekcí.

Klíčová slova: pseudonymizace, textový dataset, osobní údaje, GDPR, pojmenované entity, webový nástroj

Title: De-identification of text data collections for machine learning

Author: Bc. Martin Mareš

Institute: Institute of Formal and Applied Linguistics

Supervisor: doc. RNDr. Ondřej Bojar, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Text data collections enable the deployment of artificial intelligence algorithms for novel tasks. Such collections often contain miscellaneous personal data and other sensitive information that complicates sharing and further processing due to the personal data protection requirements. Searching for personal data is often carried out by sequential passes through the complete text. The objective of this thesis is to create a tool that helps the annotators decrease the risk of data leaks from the text collections. The tool utilizes pseudonymization (replacing a word with a different word, based on a set of rules). During the annotation process, the tool tags the words as "public", "private" and "candidate". The task of the annotator is to determine the role of the candidate words and detect any other untagged private information. The private words then become the subject of the pseudonymization process. The auto-tagging tool utilizes a named entity recognizer and a database of rules. The database is automatically improved based on the decisions of the annotator. Different named entity recognizers were compared for the purpose of personal data search on the collection of the ELITR project. During the comparison, a method was found which increased the sensitivity of the named entities detection which also increased the sensitivity of the "candidate" words detection. The work also introduces the reader to the legal issues of personal data protection during the creation of text collections.

Keywords: pseudonymization, text dataset, personal data, GDPR, named entities, web-based tool

# Obsah

**Úvod** **3**
Cíle práce . . . 4
Postup práce . . . 5
Členění práce . . . 6
Související práce . . . 6

**1 Analýza** **8**
1.1 Legislativní požadavky . . . 8
1.1.1 Ochrana osobních údajů v textových kolekcích . . . 10
1.2 Získávání informací z širšího kontextu . . . 11
1.3 Způsoby vyhledávání citlivých informací . . . 13
1.3.1 Rozpoznávání pojmenovaných entit . . . 14
1.3.2 Vliv nedokonalostí rozpoznávačů . . . 14
1.4 Možnosti automatického anotování . . . 15
1.5 Návrh aplikace . . . 17
1.5.1 Návrh architektury . . . 17
1.5.2 Použité technologie . . . 19
1.5.3 Bezpečnostní požadavky . . . 20

**2 Evaluace rozpoznávačů pojmenovaných entit** **22**
2.1 Popis datové kolekce . . . 22
2.2 Popis porovnávaných rozpoznávačů . . . 23
2.2.1 NameTag . . . 24
2.2.2 spaCy . . . 24
2.2.3 Stanford Named Entity Recognizer (NER) . . . 24
2.3 Metriky měření . . . 25
2.3.1 Vliv kvality tokenizace . . . 26
2.4 Příprava měřených hodnot . . . 28
2.5 Výsledky porovnání . . . 29
2.5.1 Citlivost po schůzkách . . . 29
2.5.2 Přesnost po schůzkách . . . 32
2.5.3 Globální výsledky . . . 33
2.5.4 Shoda úseků pojmenovaných entit s úseky citlivých informací 34
2.6 Vylepšení vyhledávání citlivých informací . . . 35
2.6.1 Výsledky vylepšeného algoritmu po schůzkách . . . 36
2.6.2 Globální výsledky vylepšeného algoritmu . . . 37
2.6.3 Použití vylepšení v rámci naší aplikace . . . 38

**3 Vývojová dokumentace** **40**
3.1 Konvence používané ve zdrojovém kódu . . . 40
3.2 Členění aplikace . . . 40
3.2.1 Členění podle místa nasazení . . . 41
3.2.2 Členění podle balíčků . . . 42
3.2.3 Adresářová struktura aplikace . . . 42
3.3 Reprezentace dat . . . 43

3.3.1 Popis databáze . . . 43
3.3.2 Ukládání do souborů . . . 45
3.4 Anotační algoritmus . . . 46
3.4.1 Automatická pravidla . . . 47
3.4.2 Fáze dokumentu . . . 47
3.4.3 Anotační proces . . . 49
3.5 Funkční popis . . . 50
3.5.1 Webové rozhraní . . . 50
3.5.2 Služby na pozadí . . . 52
3.5.3 Automatická knihovna . . . 53
3.5.4 Testy . . . 53

**4 Uživatelská dokumentace** **55**
4.1 Správa uživatelů . . . 56
4.2 Správa dokumentů . . . 57
4.3 Správa pravidel . . . 58
4.4 Správa pseudonymizačních označení . . . 59
4.5 Anotační proces . . . 60
4.5.1 Anotační okno . . . 60
4.5.2 Rozhodování . . . 61
4.5.3 Klávesové zkratky . . . 62
4.5.4 Anotace od různých anotátorů . . . 63

**Závěr** **64**

**Seznam použitých zdrojů** **66**

**Přílohy** **69**

**A Konfigurace před prvním spuštěním** **70**

**B Instrukce pro spuštění aplikace** **71**
B.1 Vytváření Docker image . . . 71
B.2 Práce s aplikací . . . 71
B.3 Spuštění pro produkci . . . 72

# Úvod

V posledních letech je možné pozorovat, jak naše společnost klade větší důraz na ochranu osobních údajů a citlivých informací. Velké softwarové společnosti v návaznosti na společenskou poptávku upravují svoje produkty pro větší ochranu soukromí [1]. Na nové společenské problémy pak v poslední době reaguje i nově přijatá legislativa.

Evropská unie v roce 2016 přijala „Obecné nařízení o ochraně osobních údajů“ častěji známé pod anglickou zkratkou „GDPR“. Toto nařízení se snaží sjednotit problematiku ochrany osobních údajů na úrovni Evropské unie a zpřehlednit jednotlivé národní úpravy [2]. Tímto nařízením se inspirovaly i některé ostatní země, například v roce 2018 začal platit v Kalifornii „California Consumer Privacy Act“ (známý pod zkratkou „CCPA“) [3].

Nařízení reagují na datové kolekce, které v posledních letech s rozvojem digitalizace vznikají skoro ve všem oborech lidské činnosti. Tyto datové kolekce většinou nějak přirozeně popisují okolní svět, takže z principu často obsahují nejrůznější osobní údaje a citlivé informace. To, co je citlivé, závisí značnou měrou na doméně daného problému. Je několik základních oblastí, které je dobré brát v úvahu:

- V kolekci se mohou vyskytovat údaje, jejichž zpracování upravuje legislativa. Podle GDPR sem patří všechny osobní údaje — tj. všechny údaje, které umožňují jednoznačně identifikovat nějakou osobu [2].
- Kolekce může svým obsahem vyzradit obchodní tajemství nebo jiné utajované informace. Odhalení takové informace pak může způsobit finanční ztrátu, ztrátu konkurenční výhody a podobně.
- Někdy i samotné přiznání existence nějaké utajované informace může představovat citlivou informaci.
- Při hledání citlivých informací je potřeba vzít v úvahu i etiku. Jako příklad lze uvést záznamy obsahující různá hodnocení, např. přepisy interní komunikace, kde jeden tým uvádí kritiku jiného týmu. I oprávněná kritika může v případě zveřejnění a identifikace způsobit škody např. ve vztazích.

Kromě osobních údajů nás mohou zajímat i další citlivé informace, které samotnou legislativní definici vždy nesplňují. Legislativa bohužel zároveň používá i pro část osobních údajů termín „citlivé osobní údaje“. Toto dělení podrobněji popíšeme v sekci 1.1. Pro samotnou detekci ale není toto rozdělení příliš důležité, takže pro potřeby tohoto textu nebudeme pojem citlivé osobní údaje používat.

Konkrétní rozhodnutí, která informace je citlivá nebo není citlivá, se může dokonce měnit i v čase. Kromě změn legislativy mohou nastat třeba i situace, kdy se z veřejných informací stávají utajované.

Datové kolekce můžeme hrubě rozdělit na strukturované a na nestrukturované (např. textové datové kolekce). Pod strukturovanou datovou kolekcí si můžeme představit například databázi. Pro zjednodušení uvažme databázi v nějaké vyšší normální formě. V strukturovaných kolekcích mají citlivé informace většinou nějaká daná místa, kde se mohou vyskytovat. V našem příkladě s databází to může

být například jeden nebo více sloupečků. Někteří autoři se už věnovali pseudonymizaci takovýchto kolekcí. Hlavní problém představuje možnost zpětné identifikace osoby z ostatních částí dat na základě nějaké unikátní vlastnosti [4]. Tuto problematiku více popíšeme v sekci 1.2.

V této práci se budeme zabývat pouze nestrukturovanými textovými datovými kolekcemi. Hlavní problém tedy pro nás představuje způsob vyhledávání citlivých informací, který je v textových kolekcích složitější. Zpětná identifikace osob, na kterou jsme narazili už u strukturovaných kolekcí, se nás ale bohužel také může dotknout.

Jak řešit samotné odstranění údajů má více možností. Pouhým odstraněním údajů z textu bychom přišli o některé důležité informace (například, které informace se týkají stejných osob). Lepším řešením ochrany údajů je proto takzvaná pseudonymizace. V tomto procesu se citlivé informace nahrazují jinými údaji podle nějakého klíče. Pokud vlastníme daný klíč, tak umíme zrekonstruovat původní dokument. Pokud klíč nemáme, tak by pro nás mělo být obtížné se vrátit k původním jménům v textu.

Motivací pro vznik práce byla potřeba zveřejňování textových datových kolekcí pro úlohu automatického psaní zápisů z jednání v rámci projektu ELITR [5]. Při řešení této úlohy byly používány hlasové záznamy z reálných schůzek. Z těchto hlasových záznamů anotátoři vytvořili přepisy a z těchto přepisů další anotátoři vytvořili zápisy. Zápisy by měly shrnovat obsah jednotlivých schůzek.

Projekt ELITR se rozhodl úlohu automatického vytváření zápisů ze schůzek koncipovat jako otevřenou vědeckou soutěž (tzv. shared task), jíž se mohou zúčastnit další akademické subjekty [6]. Kvůli potřebě sdílení vznikla textová kolekce, která obsahovala osobní údaje zúčastněných osob a názvy projektů, které bylo nutné z textu odstranit před jejich zveřejněním pro splnění legislativních a etických požadavků.

Potřeba zveřejňování textových kolekcí je teď aktuální i s Novelou zákona o soudech a soudcích,[1] která zavádí povinnost zveřejňovat soudní rozhodnutí na Internetu. Doposud rozsudky zveřejňoval jen Ústavní soud a některé nejvyšší soudy [7].

# Cíle práce

Práce si klade za cíl vytvořit anotační nástroj, který pomůže anotátorovi při vyhledávání citlivých informací v textových datových kolekcích a jejich následné pseudonymizaci. Nástroj by měl být multiplatformní a měl by umožňovat spouštění na hardwaru vlastníka dat. Anotátoři by měli mít možnost pracovat vzdáleně.

Nástroj by měl využívat dostupné existující zdroje jazykových znalostí a jazykových nástrojů a měl by cílit na úsporu času, který zaberou anotační rozhodnutí. Při provádění anotačních rozhodnutí by měl nástroj využívat znalosti z minulých rozhodnutí pro návrhy těch nových. Součástí jazykových znalostí může být například i seznam jmen získaný z nějakého externího zdroje.

V rámci projektu ELITR existují hlavně české a anglické textové kolekce. Z tohoto důvodu bude nástroj primárně cílit na tyto jazyky, ale s vhodnými jazykovými nástroji bude umět pracovat s libovolným jazykem.

[1] Zákon vyhlášen 9. 6. 2021 ve Sbírce zákonů v částce 92 pod číslem 218/2021 Sb.

Součástí práce je porovnání použitých jazykových nástrojů na datech z projektu ELITR. Tato data ale vznikala až v průběhu samotné práce, takže jejich vyhodnocení probíhalo bohužel až v posledních fázích práce. Přesto se ale podařilo pomocí těchto dat vylepšit citlivost detekce citlivých informací.

Práce se zaměřuje na hledání osobních údajů a podobných citlivých informací. Cílem práce tedy není hledat citlivé informace v delších úsecích textu. Necílíme tedy přímo na delší utajovaná sdělení. Nástroj bude i přesto umožňovat ručně označit takovou informaci jako „soukromou“.[2]

Cílem nástroje není fungovat plně automaticky ani poskytovat nějaké garance. V rámci vyhodnocení rozpoznávačů se podařilo stanovit citlivost pro záznamy ze schůzek. Pro ostatní domény se ale může citlivost lišit. Cílem nástroje je v tuto chvíli spíše anotační proces urychlit a usnadnit.

Vzhledem k ochraně osobních údajů není možné mít k dispozici veřejné datové kolekce, kde by se dal nástroj testovat. Nástroj by ale mohl napomoci, aby se tato situace v budoucnu zlepšila.

# Postup práce

Anotační proces bude probíhat ve webovém rozhraní. Samotná aplikace bude kontejnerizována, aby umožňovala snadné spouštění na infrastruktuře správce datových kolekcí. Toto řešení umožní anotátorům pracovat distančně a zároveň to sníží nebezpeční úniku kompletních dat. Celý anotační proces bude rozdělen do několika fází:

- Nejprve správce systému nahraje do aplikace dokumenty k anotaci. Každý nahraný dokument je nejprve automaticky zpracován v předanotační fázi.
- V předanotační fázi se využívají existující jazykové nástroje, které provedou segmentaci textu na věty a tokeny. Poté se použije rozpoznávač pojmenovaných entit, který detekuje základní „podezřelá slova“ (jako jsou jména, adresy, organizace atd. . . )
- V následné fázi dojde k aplikaci pravidel. Pravidla mohou být více typů, základní pravidla jsou založená na typu pojmenované entity a na tvaru slova („podle typu“). Každé pravidlo obsahuje váhu, které specifikuje míru spolehlivosti. Samotná pravidla pak dále označují slova jako „veřejná“, „soukromá“ a „podezřelá“.
- Po těchto automatických fázích přistupuje anotační proces. Anotátor dostane k dispozici omezené okno textu, ve kterém provádí anotace. Každým rozhodnutím anotátora přidáváme záznam o rozhodnutí na úrovni tokenů (tj. s kontextem). Anotátor zároveň může v rámci tohoto rozhodování přidávat nová pravidla. Po dokončení jednoho anotačního okna anotátor přechází na jiné okno. Pokud anotátor rozhodne, že dané slovo je „soukromé“, tak mu přiřadí nějaké pseudonymizační označení.
- Pokud anotátor označí nějaké nové slovo jako soukromé, tak dojde k dohledání tohoto slova i v ostatních kontextech. Za tímto účelem vznikne nové

[2]Hledání víceslovných osobních údajů ale bude podporováno.

pravidlo, které bude označovat tato slova jako „podezřelá“. Anotátor poté může dále v průběhu anotačního procesu všechna tato slova označit jako „soukromá“. Tím stále dochází k vylepšování výsledků v rámci celého dokumentu.

- Po skončení anotační fáze může administrátor ještě zkontrolovat výsledný dokument a zároveň má k dispozici výsledný pseudonymizovaný dokument ke stažení.

# Členění práce

Text je členěn do několika kapitol. V první kapitole popíšeme základní rozhodnutí, která byla při vývoji aplikace učiněna. Popíšeme si právní požadavky na ochranu osobních údajů a definujeme přesněji, co osobní údaje označují. Popíšeme problémy, které při pseudonymizaci vznikají v souvislosti s obecnými znalostmi kontextu. Uvedeme základní metody, jak lze hledat „podezřelá“ slova a popíšeme další rozhodnutí při tvorbě aplikace.

Ve druhé kapitole popíšeme, jak se od sebe liší jednotlivé nástroje na rozpoznávání pojmenovaných entit. Následně tyto nástroje empiricky porovnáme při hledání citlivých informací na testovací datové kolekci z projektu ELITR. Při porovnávání navrhneme vylepšený algoritmus, jak zlepšit vyhledávání a změříme, jak se výsledky zlepšily.

V následující třetí kapitole popíšeme aplikaci z programátorského hlediska. Ukážeme si členění aplikace a definujeme formáty dat, které využíváme. Popíšeme hlavní anotační algoritmus, postupy pro aplikaci pravidel a další principy a algoritmy, které aplikace využívá.

Ve čtvrté kapitole sepíšeme uživatelskou dokumentaci. Popíšeme, jak se aplikace používá a popíšeme, jak můžeme provádět anotační proces.

V první příloze popíšeme, jak se aplikace konfiguruje před prvním spuštěním. Ve druhé příloze definujeme požadavky pro vývojové a produkční prostředí. V této příloze také ukážeme postup, jak aplikaci spouštět.

# Související práce

Zveřejňování datových kolekcí bylo tématem už několika různých prací. Starší práce se ale věnují převážně zveřejňování strukturovaných datových kolekcí. Principy, které zavádějí, jsou přenositelné i do textových kolekcí.

Při zveřejňování strukturovaných kolekcí můžeme využít Statistické zveřejnění (*Statistical Disclosure*). V práci na toto téma nalezneme definice základních problémů, které způsobují úniky dat a doporučení, jak těmto únikům zabránit [8].

Na tuto práci v nedávné době navazuje další, která zavádí Diferenciální soukromí (*Differential Privacy*) [4]. Na rozdíl od předchozích prací se tato práce snaží řešit problém uniku informací i s využitím údajů, které nejsou přímo součástí datové kolekce.

Další práce poté řeší identifikaci konkrétních osob pomocí spojení několika relativně běžných údajů. V práci je ukázáno, že 63 % populace Spojených států amerických jde identifikovat pomocí kombinace pohlaví, poštovního směrovacího čísla a data narození [9].

Pro texty v českém jazyce můžeme nalézt práci Textan, která se zaměřuje na extrakci strukturovaných informací a určování jejich vazeb. Projekt pro extrakci informací využíval rozpoznávače pojmenovaných entit [10].

Pseudonymizaci textu se pak věnuje například další článek z poslední doby, který porovnává různé možnosti nahrazování citlivých informací v průběhu pseudonymizace. Porovnává prosté smazání úseku, nahrazení univerzálním slovem podle třídy až po nahrazením slova za slovo (tj. nahrazování každé citlivé informace nějakým určeným vzorem). Poslední jmenovaný přístup dosáhl nejlepších výsledků [11].

# 1. Analýza

V následující kapitole si popíšeme právní požadavky na proces pseudonymizace osobních údajů. Podle právních požadavků určíme, které funkce musí náš anotační aplikace podporovat a podle požadovaných funkcí rozebereme požadavky na architekturu a návrh aplikace.

## 1.1 Legislativní požadavky

Při pseudonymizaci textových kolekcí musíme splňovat legislativní požadavky. V následující sekci si uvedeme krátce tuto problematiku z právního pohledu a problematiku zasadíme do procesu vytváření textových datových kolekcí. Cílem následujících řádků není čtenáři poskytnout úplný právní pohled, ale spíše uvést zajímavé problémy, které je třeba vzít v úvahu.

Od roku 2016 v Evropské unii platí Obecné nařízení o ochraně osobních údajů (GDPR). Jedním z cílů tohoto nařízení bylo sjednotit Evropskou legislativu a zpřehlednit ochranu osobních údajů [2].

Podle mého názoru se navíc tomuto nařízení podařilo upozornit společnost na problematiku ochrany osobních údajů. Sám jsem ve svém okolí viděl několik pozitivních změn plynoucích z tohoto nařízení v nových softwarových projektech (například při řešení zámluv pokojů na kolejích Univerzity Karlovy). Tyto úpravy dokonce překonávaly samotné zákonné požadavky na ochranu údajů.

Z formálního hlediska nám GDPR definuje podmínky pro zpracování a předávání osobních údajů všech fyzických subjektů z Evropské unie. Zpracováním se v tomto případě rozumí jakékoliv operace s údaji (i bez pomoci automatizovaných postupů). Patří sem například ukládání, vymazání, vyhledávání, uspořádání a libovolné šíření nebo zpřístupnění dat. Zpřístupněním dat se rozumí dokonce i zobrazení.

Pro působnost nařízení není důležité, jestli údaje vznikly v rámci služby za úplatu nebo nikoliv. Není ani důležité, jestli se správce nebo zpracovatel údajů nachází v Evropské unii.

V rámci nařízení je definováno několik druhů údajů. U některých druhů jsou pak omezeny možnosti jejich zpracování:

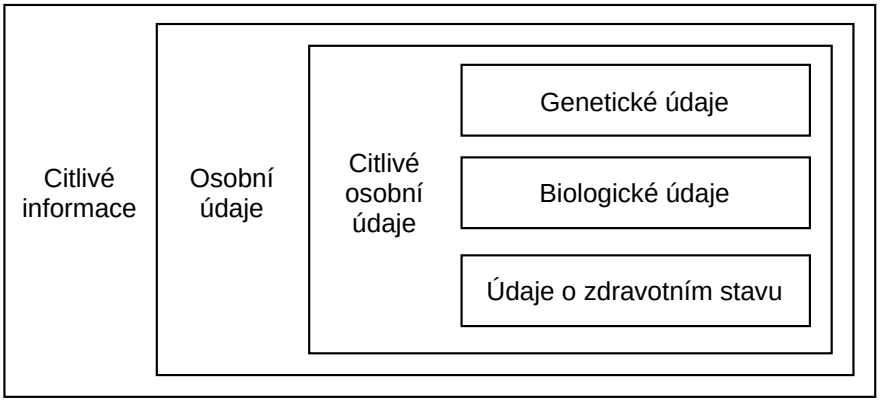
- *Osobní údaje* jsou všechny informace, které umožňují nějakou osobu přímo nebo nepřímo identifikovat. Patří sem například jméno, identifikační číslo, různé síťové identifikátory nebo i více různých prvků, které mohou dohromady identifikovat danou osobu.

- *Genetické údaje* jsou speciálním druhem osobních údajů, které se týkají zděděných nebo získaných genetických znaků fyzické osoby. Mohou vypovídat o její fyziologii nebo zdraví.

- *Biologické údaje* jsou pak druhem osobních údajů, které vyplývají z konkrétního technického zpracování týkající se fyzických či fyziologických znaků nebo znaků chování fyzické osoby. Patří sem například zobrazení obličeje nebo daktyloskopické údaje.

- *Údaje o zdravotním stavu* jsou osobní údaje, které popisují fyzické či psychické zdraví. Mezi tyto informace patří i samotná informace o poskytnutí zdravotních služeb.

Pro genetické údaje, biologické údaje a pro údaje o zdravotním stavu se někdy používá označení *citlivé osobní údaje*[1] (viz Obrázek 1.1). Pro citlivé osobní údaje platí přísnější požadavky při jejich zpracovávání. Pro potřeby naší aplikace ale není toto rozlišení tak podstatné, protože lingvisticky se jejich vlastnosti neliší, takže citlivé osobní údaje nebudeme v textu od těch ostatních odlišovat.




Obrázek 1.1: Množinové uspořádání používaných pojmů

Pro zpracování libovolných osobních údajů musí vždy existovat nějaký relevantní důvod (tj. zpracování musí být nějak opodstatněné). V samotném nařízení existuje několik vyjmenovaných důvodů pro zpracování osobních údajů. Mezi tyto důvody patří mimo jiné zpracování údajů pro účely plnění smlouvy, splnění nějaké zákonné povinnosti nebo případně zajištění nějakého veřejného zájmu nebo výkonu veřejné moci.

Pokud mezi těmito důvody účel našeho zpracování není, tak musíme vyžadovat explicitní souhlas se zpracováním osobních údajů.[2] Tento souhlas musí být oddělený od ostatních souhlasů (smluv) a musí být napsaný za použití jasných a jednoduchých jazykových prostředků. Speciální podmínky pak platí pro udělení souhlasu pro zpracování osobních údajů dítěte.

Souhlas musí vždy obsahovat kategorie osobních údajů, které jsou ukládány. Dále musí obsahovat i transparentní soupis účelů, pro které jsou osobní údaje zpracovány a musí specifikovat dobu, po kterou budou údaje správcem uloženy. Souhlas by měl pak obsahovat informací, jestli údaje budou někam předávány.

Zpracování citlivých osobních údajů je až na výjimky zakázáno. Patří sem například údaje o rasovém či etnickém původu, politických názorech, náboženském vyznání či filozofickém přesvědčení nebo členství v odborech. Zpracování genetických údajů, biometrických údajů za účelem jedinečné identifikace fyzické osoby a údajů o zdravotním stavu či o sexuálním životě nebo sexuální orientaci fyzické osoby je také zakázáno.

[1] V našem textu používáme širší pojem *citlivé informace*, kam patří všechny informace, které nechceme zveřejňovat.

[2] Nařízení dokonce zakazuje vyžadovat souhlas pro zpracování, pokud účel zpracování je dovolen přímo v nařízení.

Tento zákaz má ale několik výjimek. Citlivé osobní údaje je možné stále zpracovávat pro účely vědeckého nebo historického výzkumu. Další možností je získat výslovný souhlas pro přesně specifikované účely i pro tyto kategorie (některé kategorie pak mohou být v některých zemích Evropské unie pro zpracování úplně zapovězeny). U citlivých osobních údajů je pak na místě zachovávat transparentní přístup při komunikaci s osobami, kterých se tyto údaje týkají.

Obecně nám nařízení pro vědecké účely připouští trochu volnější definici účelu zpracování osobních údajů, protože „často není možné v době shromažďování osobních údajů v plném rozsahu stanovit účel zpracování osobních údajů pro účely vědeckého výzkumu. Subjektům údajů by proto mělo být umožněno, aby udělily svůj souhlas ohledně určitých oblastí vědeckého výzkumu v souladu s uznávanými etickými normami pro vědecký výzkum. Subjekty údajů by měly mít možnost udělit svůj souhlas pouze pro některé oblasti výzkumu nebo části výzkumných projektů v rozsahu přípustném pro zamýšlený účel“ [2].

Při zpracovávání osobních údajů je třeba brát v úvahu, že souhlas se zpracováním může být později i odvolán a my musíme mít možnost zajistit smazání části osobních údajů z naší datové kolekce (tzv. „právo být zapomenut“).

Fyzická osoba, která je subjektem osobních údajů, může dále požadovat potvrzení, jestli správce o ní nějaké údaje ještě zpracovává. Případně může požádat o opravu osobních údajů.

Jedním z účelů nařízení je zvýšit ochranu údajů před zneužitím a úniky. Samotné nařízení doporučuje pro tyto účely používat (v rozumné míře) pseudonymizaci. Zavedení pseudonymizace ale nemá za cíl předem vyloučit další opatření týkající se ochrany údajů.

Pseudonymizací se z pohledu nařízení rozumí „zpracování osobních údajů tak, že již nemohou být přiřazeny konkrétnímu subjektu údajů bez použití dodatečných informací, pokud jsou tyto dodatečné informace uchovávány odděleně a vztahují se na ně technická a organizační opatření, aby bylo zajištěno, že nebudou přiřazeny identifikované či identifikovatelné fyzické osobě“ [2].

### 1.1.1 Ochrana osobních údajů v textových kolekcích

Z pohledu textových datových kolekcí hodně záleží na doméně, kterou data popisují. Například v zápisech z projektových schůzek budeme nacházet jiné údaje než v lékařských zprávách. V každé doméně pak může záležet dále i na kontextu, ve kterém se informace vyskytuje. Pokud budeme vyhledávat informace o koronaviru, tak debata o statistice šíření koronaviru v České republice není osobním údajem, ale informace, že se nějaká osoba koronavirem nakazila už osobním údajem je.

Z tohoto důvodu je nutné pro některá rozhodnutí používat kontext. Můžeme sice všechny zmínky o koronaviru z textu odstranit, ale tím zbytečně odstraníme i informace, které nemají povahu osobních údajů. Jak už jsme si řekli, zpracování údajů o zdravotním stavu patří do omezenější skupiny osobních údajů, takže na jejich zpracování jsou kladeny větší právní nároky. Navíc v tomto případě nestačí pouze odstranit samotný název nemoci, ale i samotná informace o prodělání nějaké nemoci se dá považovat za osobní údaj.

V rámci pseudonymizace textových kolekcí dochází k náhradě osobních údajů za údaje, které bez klíče nevedou k identifikaci konkrétních osob. Osoba, která

zná kontext dané domény, většinou stále může identifikovat původní osoby z její vlastní znalosti daného kontextu. Můžeme si třeba představit situaci, kdy se jeden tým schází jako jediný v kavárně. Díky tomu můžeme snadno určit, o který tým se jedná, jenom díky informaci, že pijí kávu. Tuto problematiku více rozebereme v následující sekci.

Tato skutečnost vede k úvaze, jak „dobrá“ pseudonymizace má být. Nařízení GDPR stanovuje, že při pseudonymizaci je potřeba vzít v úvahu objektivní faktory, kolik úsilí, prostředků a času by případná zpětná identifikace vyžadovala. U toho rozhodování je potřeba vzít v úvahu dobu zpracování údajů a technologický rozvoj.

Při zpracovávání dat z projektu ELITR se osvědčilo pro zachování transparentnosti a etického přístupu (i z důvodu možné zpětné identifikace) dávat pseudonymizovaný text ke kontrole původním osobám z daných schůzek.

Etický rámec, který by měl upravovat všechny procesy na univerzitě, je na Univerzitě Karlově zakotven v Etickém kodexu univerzity [12]. Ten obsahuje principy a doporučení pro vhodnou vědeckou spolupráci a kolegialitu. Etický kodex dále nabádá k zajištění přezkoumatelnosti výzkumu a dodržování všech zákonů a dobrých mravů. Dle mého názoru, ale zatím ještě příliš neupravuje postavení subjektů výzkumu a ochranů jejich osobních údajů.

Z právního pohledu jsou tedy pro naše účely důležité všechny údaje, které umožňují přesnou identifikaci jedné osoby. Kromě těchto údajů dále potřebujeme mít možnost vyhledat údaje, které popisují zdravotní stav, genetický znak nebo biologický vzorek nějaké konkrétní osoby. Kromě osobních údajů nás v některých doménách mohou zajímat i další informace jako například názvy různých výrobků a označení projektů. Pro všechny tyto informace budeme dále používat souhrné označení *citlivé informace*.

## 1.2 Získávání informací z širšího kontextu

Jak jsme zmínili v úvodu, citlivé informace je možné dovodit i ze znalostí širšího kontextu. Prozrazení informací ze statistických datových kolekcí v širším kontextu řeší už i články, které jsou více než 40 let staré [8].

Pro nás je zajímavý například článek od T. Dalenia [8], který definuje základní terminologii. Přitom zavádí pojem tzv. *Statistického zveřejnění* (Statistical Disclosure). Tato operace se snaží najít postupy, jak poskytnout určitou rovnováhu mezi ochranou citlivých informací a užitečností výsledných datových kolekcí po aplikaci této operace. Pokud totiž budeme vracet pouze náhodné údaje nebo prázdné řetězce, tak určitě ochráníme všechny citlivé informace, ale užitečnost výsledné kolekce bude naprosto nulová.

V rámci toto článku je identifikováno 6 základních oblastí, které představují problémy. Tyto oblasti sice převážně cílí na strukturované datové kolekce, ale velmi podobné problémy nalezneme i u textových kolekcí. Jeden z těchto problémů si pro ilustraci ukážeme na příkladu. Nechť máme Tabulku 1.1, která popisuje věkovou strukturu ve fiktivní zemi „C“.

K této tabulce jsme získali další Tabulku 1.2, která popisuje dodatečné informace o kriminalitě v jednotlivých věkových skupinách.

Tyto dvě tabulky sice přímo neobsahují žádné citlivé informace, ale přesto jsme z nich schopni získat několik potenciálně citlivých informací [8]:

| Země | Věková skupina | Počet můžu | Počet žen |
|---|---|---|---|
| C | 0 | 5 | — |
| C | 1 | — | 50 |

Tabulka 1.1: *Příklad:* Rozložení počtu osob podle pohlaví a věkové skupiny [8]

| Země | Věková skupina | Trestané osoby | Netrestané osoby |
|---|---|---|---|
| C | 0 | 5 | — |
| C | 1 | — | 50 |

Tabulka 1.2: *Příklad:* Rozložení počtu trestů podle věkové skupiny [8]

- Všechny osoby ze skupiny „0“ byli trestané.
- Všechny osoby ze skupiny „0“ jsou muži.
- Všichni muži ze země „C“ byli trestáni.

Problémem tedy je, pokud ze zveřejněné kolekce umíme vyčlenit libovolnou podskupinu velikosti 1, případně pokud něco můžeme říci o nějaké celé skupině. V obou případech je v principu možné identifikovat nějaké konkrétní osoby. Původní článek doporučuje řešit tyto úniky pomocí dvou hlavních metod, jak se s tímto problémem vypořádat. Jednou možností je poskytovat data pouze za pomoci vzorkování. Další metodou je omezit minimální velikosti skupiny, kterou popisujeme, nějakou konstantou [8].

I v textových kolekcích můžeme narazit na podobný druh informací. Nestrukturovaná forma ale značně znesnadňuje hledání míst, kde tyto informace jsou. Dle mého názoru jsou navíc tyto metody automaticky těžko realizovatelné v souvislém textu. Toto téma je ale vhodné řešit s anotátory, kteří procházejí textovou kolekci. Samotný článek zmiňuje možnost sepsat si nějaká pravidla, která se budou v průběhu hledání osobních údajů dodržovat.

Starší články obecně neřeší citlivé informace, které jsme schopni dovodit za použití dodatečných informací, které nemusí vůbec být součástí samotné datové kolekce [4].

Můžeme si to ukázat na příkladu. Nechť je citlivou informací výška dané osoby. Pokud zveřejníme průměrnou výšku žen v populaci, tak nám tím sice přímo žádná informace neunikne, ale problém nastane, pokud budeme mít dodatečnou informaci, že jedna konkrétní osoba je o 10 % svojí výškou nadprůměrná. Tím se dostáváme do konfliktu s jednou z interpretací statistického zveřejnění, že „o jednotlivci nemáme mít možnost dozvědět se s daty z kolekce nic jiného než víme bez kolekce“ (v originále „nothing about an individual should be learnable from the database that cannot be learned without access to the database“) [4].

Tento problém se snaží řešit algoritmy *Diferenciálního soukromí* (Differential Privacy) [4], které přidává k výsledným datům určitý šum. Cílem je přidávat k poskytovaným datům šum s takovým rozptylem, abychom nedokázali určit, jestli se jeden konkrétní jedinec mohl podílet na určení výsledku. Opět si to můžeme ilustrovat na příkladu. Nechť nás zajímá počet osob, které byly trestány. Standardní cestou určíme výsledek, který budeme považovat za střední hodnotu normálního

rozdělení s určitým rozptylem. Samotný rozptyl je pak určen z vlastností datové kolekce. Rozptyl by měl zajistit, že nebudeme mít jistotu, kolik osob se na výsledku podílelo. Popularitu tomuto přístupu mimo jiné nedávno zajistilo i jeho použití u chytrých telefonů od společnosti Apple [13].

Na využívání externích informací pak staví další výzkumy. Podle dat z roku 2000 se ukázalo, že 63 % populace Spojených států amerických jde identifikovat pomocí kombinace pohlaví, poštovního směrovacího čísla a data narození. Podle dat z roku 1990 to bylo dokonce 87 %, případně 53 % při použití pohlaví, data narození a jména města nebo obce, ve kterém osoba aktuálně bydlí [9].

Z pohledu pseudonymizace textových kolekcí obecně nemůže ovlivnit nějakou dodatečnou informaci. Při hledání citlivých informací je nutné vzít v úvahu doménu, kterou dané texty popisují. Z pohledu GDPR můžeme při pseudonymizaci vzít v úvahu objektivní zdroje pro případnou zpětnou identifikaci s přihlédnutí k aktuálnímu technickému rozvoji [2].

Pro návrh samotné aplikace tedy pro nás plyne, že je vhodné zaměřit se na nejčastější údaje, které mohou být použity ke zpětné identifikaci. Patří sem jména, adresy a různé časové údaje. Určité riziko pak představují i některé číselné údaje (rodné číslo).

Textové datové kolekce z mého pohledu budou vždy představovat trochu větší riziko, protože samy o sobě obsahují větší množství informací, které nemusí mít přesně definovanou strukturu. V rámci pseudonymizace v projektu ELITR se proto pseudonymizované kolekce daly k dispozici osobám, které původně popisovaly. Tím si tyto osoby mohly ověřit, že ve výsledných datech nejsou informace, které by jim nebyly komfortní. To i zároveň napomohlo transparentnosti celého procesu.

## 1.3 Způsoby vyhledávání citlivých informací

V následující sekci si ukážeme možnosti, jak hledat jednotlivé citlivé informace. V předchozích částech jsme si určili, že pro splnění legislativních požadavků potřebujeme vyhledávat převážně údaje, které umožňují nějakou přímou identifikaci nějaké fyzické osoby.

Základním identifikátorem, který se pro tyto účely používá je v České republice rodné číslo případně číslo občanského průkazu. Tyto údaje samy o sobě identifikují jednu fyzickou osobu. Dále můžeme jednu osobu i identifikovat pokud spojíme více běžných údajů dohromady. V této spojitosti se jedná zejména o jméno, adresu a datum narození. Tyto údaje, které jsou dostatečné pro identifikaci jedné fyzické osoby budeme nazývat *identifikující údaje*.

Pro hledání těchto identifikujících údajů je možné využít regulárních výrazů. Regulární výrazy nám například umožní hledat číselné údaje o určité délce. Při využití velikosti písmen jsme pak schopni hledat slova sloužící k pojmenování. U jazyků jako je němčina, kde všechna podstatná jména začínají velkým písmenem, ale tento přístup neuspěje. I v češtině je pak tento přístup problematický na začátcích vět.

Další možností, jak tyto identifikující údaje hledat, je využít vlastnosti, že každý identifikující údaj popisuje vlastně nějakou pojmenovanou entitu z reálného světa. Pro hledání takových entit existují speciální nástroje pro rozpoznávání pojmenovaných entit. Tento přístup se využívá i u některých aplikací, které

v textových kolekcích hledají strukturovaná data [10].

### 1.3.1 Rozpoznávání pojmenovaných entit

Rozpoznávání pojmenovaných entit je úloha, ve které hledáme výskyty slov a slovních spojení, která označují jménem nějaké objekty reálného světa, v nestrukturované textové datové kolekci.

U každé pojmenované entity pak určujeme její třídu (jako např. jméno osoby, adresa, název instituce, atd.). Množství tříd a podrobnost jejich členění pak vždy záleží na použitém rozpoznávači.

Většinou rozpoznávače pojmenovaných entit pracují nad tokeny. Při hledání se interně využívají další indikátory jako je velikost písmen, předpony, přípony, tagy a postavení tokenu v rámci závislostního stromu. Některé rozpoznávače zvládnou rozlišovat, jestli je výsledná pojmenovaná entita tvořená více tokeny. Některé pak dokonce podporují jemnější třídy a tudíž pojmenované entity mohou být různě do sebe zanořené. Nechť nás pro příklad zajímá, jak dopadne rozpoznání fráze „Jan Novák“:

- Může od rozpoznávače dostat dvě pojmenované entity různého typu „Jan“ (jméno) a „Novák“ (příjmení).
- Může dostat dvě pojmenované entity stejného typu „Jan“ (pojmenování) a „Novák“ (pojmenování).
- Může dostat jednu pojmenovanou entitu „Jan Novák“ (pojmenování), v některých případech pak toto pojmenování může obsahovat podrobnější vnořené jmenné entity typu (pojmenování-jméno) a (pojmenování-příjmení).

Z pohledu pseudonymizace textových kolekcí může pro nás dávat smysl rozhodnout o nějaké kategorii jako o celku. To dává smysl u jmen vládních institucí, která pro nás většinou nemusí představovat citlivou informaci. U některých jmen naopak chceme mít možnost rozhodovat s kontextem ve větě. Oblíbené jméno „Jan Novák“ může být v rámci textové kolekce v různých ilustračních příkladech dovoleno, ale konkrétní osoba „Jan Novák“ by měla být pseudonymizována.

Rozhodnutí, které třídy pojmenovaných entit jsou pro nás zajímavé, záleží na doméně dané textové kolekce. Naše aplikace proto bude umožňovat upravovat rozhodnutí o pravidlech nad typy pojmenovaných entit i v průběhu anotačního procesu.

Nástrojů na rozpoznávání pojmenovaných entit existuje více (např. [14], [15] a [16]). V následující kapitole 2 je porovnáme a vybereme ten nejvhodnější pro naše účely.

### 1.3.2 Vliv nedokonalostí rozpoznávačů

Při rozpoznávání pojmenovaných entit občas narazíme na situace, kdy rozpoznávač nerozpozná celé označení jako pojmenovanou entitu. Tento případ můžeme pozorovat u delších označení jako je „Městský soud v Brně“, kde rozpoznávač v některých případech rozpozná pouze „Brno“ jako pojmenovanou entitu, která popisuje jen název města.

Z tohoto důvodu se ukázalo, že musíme v naší aplikaci podporovat i anotovaní vlastních úseků textu, které nemusí nijak souviset s nálezy rozpoznávače pojmenovaných entit.

Jak už jsme viděli v příkladu s Městským soudem v Brně, tak může docházet k zanoření nebo i průniku anotace od anotátora s návrhem od rozpoznávače pojmenovaných entit. Tento efekt může být ještě komplikovanější, pokud rozpoznávač sám o sobě generuje vnořené pojmenované entity.

Kromě nedokonalostí rozpoznávače narážíme i na nedokonalosti v samotné textové datové kolekci. Samotná slova mohou obsahovat překlepy, které komplikují přesnost rozpoznávače, ale bohužel často nepředstavují takový problém pro zpětnou identifikaci.

Zajímavý problém nastává v souvislosti s odvozováním slov. To si můžeme ilustrovat na dvojici „Madrid“ vs. „Madriďan“. Madrid pro nás může znamenat místo, kde plánujeme další rozvoj naší práce a Madriďan může pak označovat členy týmu, kteří se mají na tomto rozvoji podílet. Oba výrazy tedy pro nás mohou představovat citlivou informaci.

Pokud budeme pseudonymizovat tyto informace, tak bychom v ideálním případě chtěli zachovat vazbu u těchto slov i v pseudonymizovaném textu. Pro tyto účely lze použít lingvistické nástroje jako je DeriNet [17], které by nám umožnily hledat vazby mezi slovy. Bohužel se ukázalo, že pro méně častá slova (jako výše zmíněný Madriďan) nástroj hledanou vazbu neodhalí.

Kromě samotných identifikujících údajů nás ještě zajímají možnosti zpětné identifikace z delšího souvislého textu popisujícího obecnými slovy specifickou a tedy identifikovatelnou situaci (konference, veletrh případně jiná mimořádná situace libovolného druhu). Naše aplikace proto musí umožňovat označit i delší úsek textu jako „soukromý“.

Existují nástroje, které sice umožňují vytvářet z textových údajů strukturovaná data, ale jejich spolehlivost by záležela na doméně dané textové kolekce (případně jsou omezeny podporovanými jazyky). V některých případech se můžeme při detekci takových textů inspirovat vyhledáváním klíčových slov — ukázalo se, že některé druhy textů obsahují tyto „zajímavé údaje“ za nějakým klíčovým slovem jako je například „bytem . . . “ [10]. Dle mého názoru se tyto úseky často vyskytují v blízkosti nalezených pojmenovaných entit, takže je anotátor může nalézt při rozhodování o „podezřelých“ slovech v rámci čtení okolního textu.

## 1.4 Možnosti automatického anotování

V rámci samotné anotační aplikace chceme mít možnost automaticky rozhodovat, které nalezené (identifikující) údaje tvoří citlivou informaci. Pro automatické rozhodování chceme šetřit množství rozhodnutí využíváním určitých společných vlastností nalezeného textu.

Pro některá rozhodnutí budeme potřebovat kontext, ale pro některé údaje budeme moci rozhodnout i bez něj. Proto dostáváme 3 druhy anotačních rozhodnutí ohledně úseku textu splňujícího nějakou určitou vlastnost:

- Úsek je pro nás *podezřelý*. To znamená, že buď o něm ještě nebylo rozhodnuto nebo konkrétní rozhodnutí závisí vždy na kontextu.

- Úsek je vždy *veřejný*. Máme jistotu, že tento údaj nikdy netvoří citlivou informaci.

- Poslední možností je, že daný úsek je *soukromý*. To pro nás představuje situaci, když vždy víme, že daný úsek tvoří citlivou informaci.

Abychom vytvořili způsob, jak automaticky spojovat úseky textu se stejnou vlastností s anotačními rozhodnutími, tak vytvoříme *automatická pravidla*. Každé pravidlo nám popisuje nějaký způsob, jak nalézt určité úseky textu a popisuje typ anotačního rozhodnutí, které s těmito úseky souvisí. V předchozím příkladu se státní institucí takto můžeme získat pravidlo, že všechny státní instituce jsou veřejný úsek textu.

Každá kategorie nalezená pomocí rozpoznávače pojmenovaných entit pro nás představuje jedno pravidlo. Ve výchozím stavu je vhodné považovat všechny tyto úseky za podezřelé. Anotátor je může případně v průběhu anotačního procesu změnit podle domény textové kolekce. Případně dává smysl, aby aplikace umožnila nastavit tato rozhodnutí dopředu.

Kromě automatických pravidel anotátor potřebuje mít možnost rozhodovat nad nějakým úsekem za použití kontextu. Takovým rozhodnutím budeme říkat rozhodnutí na *úrovni tokenu*.

Rozhodnutí anotátora na úrovni tokenu má obecně větší váhu, protože toto rozhodnutí bylo vytvořeno s větším množstvím informace a anotátor do něj vložil přidanou práci. U automatického pravidla oproti tomu nemusel anotátor vždy promyslet všechny kontexty. V některých textech pak dává smysl, aby pravidlo popisovalo většinové rozhodnutí, ale různé okrajové situace bylo možné dořešit ručně.

Z těchto důvodu nechceme měnit výsledné rozhodnutí, pokud se později objeví nové automatické pravidlo, které by bylo s rozhodnutím na úrovni tokenu v konfliktu. Abychom tuto situaci vyřešili, musíme pro každý údaj rozlišovat mezi dvěma druhy rozhodnutí:

- Rozhodnutí na úrovni tokenu, která vznikla za použití kontextu.

- Automatická rozhodnutí, která vznikla podle jednoho nebo více pravidel.

Pro vyřešení problémů s případnou nedokonalostí rozpoznávače pojmenovaných entit jsme zavedli podporu pro vytváření anotací mimo nalezené pojmenované entity.

Pokud nějaký úsek označíme jako „soukromý“, získáme informaci, že toto spojení slov je v nějakém kontextu „soukromé“, ale nemusíme mít informaci, jestli může být v jiném kontextu i „veřejné“. Abychom mohli tyto informace dále využívat, vytvoříme nový typ pravidel, který bude označovat úseky se stejným obsahem v dalších kontextech jako „podezřelé“. Tento druh pravidel budeme nazývat jako pravidla *podle typu* slova.

U jazyků se skloňováním se nabízí využít pravidel, která by hledala slova bez ohledu na konkrétní tvar slova, tj. podle základních tvarů, tzv. *lemat*. Samotný algoritmus to sice nezmění, ale při anotačním procesu musí anotátor kontrolovat zároveň i přiřazené lema. Automatická aplikace automatických pravidel na úseky textu mimo kontrolu anotátora pak zavádí další míru nejistoty, která komplikuje

rozhodování anotátora, zda použít toto pravidlo s lematem, nebo zda rozhodnout na úrovni tokenu. Další druh pravidel také komplikuje uživatelské rozhraní, protože je potřeba umět řešit situace, kdy bude lema nalezeno špatně. Dle mého názoru pravidla podle lematu spíše znepřehledňují celý anotační proces.

V aplikaci máme zavedeno několik druhů automatických pravidel, takže nastává možnost, že více pravidel bude označovat jeden úsek textu. V takovém případě existuje možnost, že pravidla budou produkovat odlišná rozhodnutí. V této situaci můžeme využít, že každé pravidlo má přirozeně jinou míru spolehlivosti, která plyne z metody, kterou pravidlo používá. Můžeme například důvěřovat více pravidlům založeným na slovníku než pravidlům, která plynou z detekce pojmenovaných entit. Abychom tuto míru jistoty mohli použít při automatickém rozhodování v naší aplikaci, budeme ke každému pravidlu uchovávat míru spolehlivosti vyjádřenou číslem. V rámci jednoho úseku pak pravidla budou hlasovat o výsledné anotaci. Pokud bude míra jistoty malá (například pravidla budou hlasovat pro opačné výsledky), necháme o výsledné anotaci rozhodnout anotátora na úrovni tokenu.

## 1.5 Návrh aplikace

Při návrhu aplikace musíme vzít v úvahu i uživatelskou přívětivost. Je důležité, aby se anotátorům s aplikací dobře pracovalo, protože pak budou produkovat kvalitnější výsledky. Zároveň musíme respektovat právní požadavky týkající se zacházení s osobními údaji.

Jak jsme ukázali v předchozích částech, existuje více možností, jak hledat osobní údaje a jak provádět samotný anotační proces. Naším cílem je proto navrhnout aplikaci takovým způsobem, aby se dala do budoucna upravovat. Oproti výkonnostním požadavkům potřebujeme upřednostňovat hlavně bezpečnost zpracovávaných dat podle požadavků plynoucích z legislativy.

### 1.5.1 Návrh architektury

Při zpracovávání osobních údajů musíme podle obecného nařízení na ochranu osobních údajů specifikovat jejich správce a rozhodnout zda budeme osobní údaje někam předávat. Z tohoto důvodu potřebujeme zajistit, že osobní údaje neopustí hardware jejich správce. Zároveň v rámci tzv. práva být zapomenut potřebujeme mít možnost údaje odstranit ze všech míst, kde je máme uložené [2].

Pro řešení tohoto problému se hodí využití architektury klient-server, abychom se vyhnuli potřebě distribuovat osobní údaje k uživatelům aplikace. Na serveru takto můžeme mít uložená všechna data a klientovi pak můžeme zpřístupňovat pouze takovou část dat, kterou bude potřebovat pro svoje rozhodování.

Tento přístup umožňuje zároveň zaměstnávat anotátory na dálku, protože nepotřebujeme, aby k běžné práci byli přítomni na jednom místě nebo vlastnili speciální hardware. Tato výhoda umožňuje snížit náklady na anotační proces.

Existuje více možností jak realizovat klient-server aplikaci. Můžeme vytvářet klasické desktopové aplikace pro klienta a se serverem můžeme komunikovat pomocí nějakého API nebo metod jako je RPC. Alternativně můžeme založit aplikaci na webových technologiích. Webové technologie jsou dnes na rozdíl od

desktopových aplikací pro běžné uživatele snadnější pro používání, protože nepotřebují prvotní instalaci. Tím zase snížíme nároky na anotátory. Navíc tím získáme podporu pro jiné operační systémy.

Některé požadované funkce vyžadují delší čas na zpracování. Při úkolech jako je rozpoznávání pojmenovaných entit, případně aplikace pravidel obecně nemůžeme očekávat, že operace bude provedena okamžitě. Abychom zabránili situacím, kdy uživatel nebude moci v takových situacích aplikaci dále používat, musí aplikace podporovat mechanismus pro provádění delších úloh na pozadí.

Použitím webových technologií ulehčíme používání aplikace koncovým anotátorům. Běžná serverová aplikace pořád potřebuje určité znalosti pro zprovoznění na straně serveru. Legislativa nám pak znesnadňuje provozování aplikace u třetí strany. Pokud aplikaci provozuje třetí strana, dochází k dalšímu zpracování údajů, které GDPR dále omezuje [2].

Pro zjednodušení zprovoznění aplikace na serveru, můžeme využít různých způsobů virtualizace a kontejnerizace. Pokud použijeme plnou virtualizaci, bude naše aplikace oddělená od ostatních aplikací na úrovni celého operačního systému. To může poskytovat větší ochranu před softwarovými útoky, ale ani to nás neochrání před chybami v hardwaru (typickým příkladem jsou bezpečnostní chyby v procesoru [18]). Kontejnerizace oproti tomu odděluje běhové prostředí jen na úrovni user space. Při kontejnerizaci totiž dochází ke sdílení jádra operačního systému mezi ostatními kontejnery. To nám přináší nižší režijní náklady (overhead) a zároveň i menší velikost výsledného kontejneru. Díky různým chybám v procesorech může být bezpečnostní přínos plné virtualizace z ohledu bezpečnosti sporný [18]. Abychom ušetřili režijní náklady využijeme raději pro naší aplikaci kontejnerizaci.

Pro použití kontejnerizace existuje opět několik nástrojů, jak ji realizovat. Obecně všechny tyto možnosti nějak propojují vestavěné funkce, které poskytuje linuxový kernel.

Nejpoužívanější technologií je bezesporu Docker, který celému tomuto odvětví dle mého názoru získal nejvíce pozornosti. Docker zastává filozofii, že každá aplikace má běžet právě v jednom Docker kontejneru. Při spuštění Docker kontejneru se spustí právě jeden program [19].

Tato filozofie je ale v rozporu s přístupem ke správě služeb, která se ve světě Linuxu prosazuje ve spojitosti se *systemd*. Systemd sjednocuje spouštění, závislosti a životní cyklus služeb v různých linuxových distribucích. V poslední době zde představuje určitý standard. Docker ho ale kvůli své filozofii nevyužívá. Tuto situaci se snaží řešit *Pod Man tool* (podman), který využívá systemd a tím umožňuje spouštět více aplikací v jednom kontejneru.

Ačkoliv přístup podmanu více zapadá do linuxového světa, tak bohužel zatím není tolik rozšířený. Naše aplikace podporuje kontejnerizaci, aby ulehčila nasazování na serveru, proto jsme zvolili lépe podporované řešení v podobě využití Dockeru.

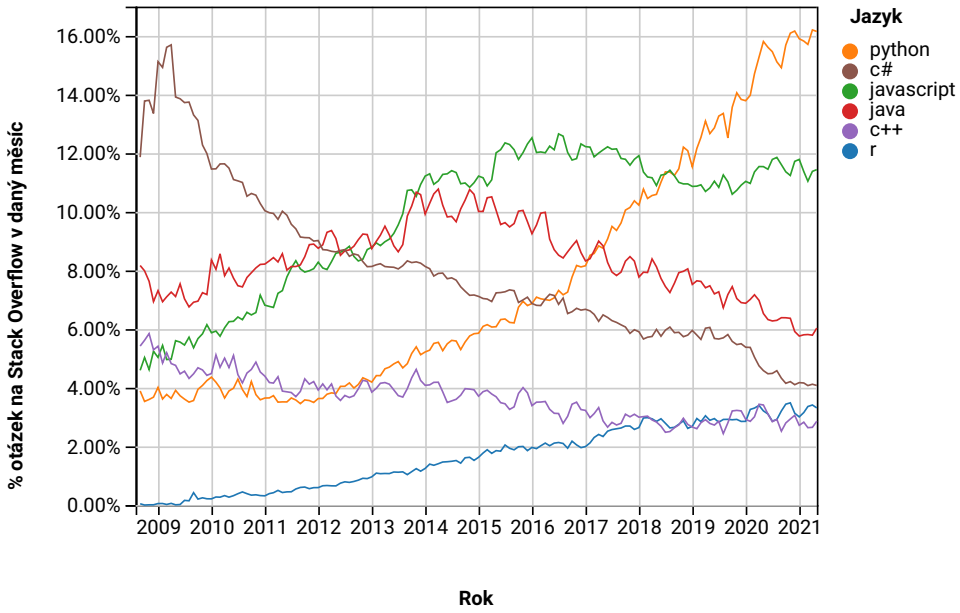
Pro splnění funkčních požadavků při použití kontejnerizace v Dockeru, musíme rozdělit aplikaci do více kontejnerů. Rozdělení na více kontejnerů komplikuje náš požadavek na jednoduché nasazení aplikace na serveru. Existuje několik nástrojů, které nám s tímto problémem mohou pomoci. V případě většího serveru je možné využít nástrojů pro orchestraci jako je Kubernetes. Pro jednodušší nasazení, nám ale plně dostačuje jednoduší nástroj Docker Compose. V případě nějakých složi-

tějších nasazení tato volba nebude představovat potřebu změny architektury.

### 1.5.2 Použité technologie

Pro návrh webové aplikace můžeme využít několik programovacích jazyků. Naše práce cílí na algoritmy z umělé inteligence, které mají většinou podporu v Pythonu. Kvůli funkčním požadavkům chceme mít možnost pracovat s rozpoznávači pojmenovaných entit. Některé jsou nativně napsány v C++ [14] případně v Javě [16]. Všechny tyto nástroje ale mají API do Pythonu [20]. Jeden z nástrojů pak primárně cílí pouze na Python [15].

Python je dnes už nejpopulárnějším jazykem (podle počtu otázek na webu StackOverflow, viz Obrázek 1.2). Navíc je snadno flexibilní díky vlastnostem jako je dynamické typování. Jazykové nástroje, které potřebujeme pro hledání citlivých údajů, pak přímo Python podporují. Naše aplikace je proto napsána za použití Pythonu.




Obrázek 1.2: Popularita programovacích jazyků na StackOverflow [21]

Při vývoji webu v Pythonu máme k dispozici dva hlavní webové frameworky, Flask a Django. Django cílí na větší aplikace s větším množstvím uživatelského rozhraní. To může v některých aplikacích ušetřit práci, ale dělá ho to komplikovanějším. Flask oproti tomu slouží spíše minimalisticky.

V naší aplikaci je nejkomplikovanější částí uživatelského rozhraní rozhraní pro anotační proces. Tato část je navíc značně specifická, proto nám s ní Django příliš nepomůže. Jeho větší složitost pak samotný vývoj spíše zkomplikuje. Flask je jednodušší a dává nám velkou svobodu v používání, takže je pro naše účely vhodnější.

Ukládání dat je v aplikaci potřeba rozdělit na dvě části. V jedné části ukládáme strukturovaná data o uživatelích a dokumentech. Kromě těchto dat musíme ukládat i nestrukturovaná data v podobě textových dokumentů.

Strukturovaná data se většinou ukládají do databáze. Tím zároveň můžeme více ochránit jejich integritu. Z pohledu požadavků jsou všechny databáze poměrně rovnocenné. Naše aplikace využívá PostgreSQL, protože má open source licenci a má dobrou podporu SQL standardu [22].

U textových kolekcí můžeme využít pro ukládání přímo souborový systém nebo databází pro ukládání dokumentů jako je MongoDB. V určité fázi anotačního procesu potřebujeme umožnit pracovat s dokumenty rozpoznávačům pojmenovaných entit. Žádný z námi uvažovaných rozpoznávačů nějakou takovou databázi přímo nepodporuje. Pro naši aplikaci by tento přístup znamenal napsat si nějaký vlastní adaptér. Tento adaptér by pro ostatní části aplikace nepředstavoval v tuto chvíli žádný větší přínos, proto naše aplikace ukládá textovou datovou kolekci přímo do souborů v souborovém systému.

### 1.5.3 Bezpečnostní požadavky

Kvůli omezení úniků údajů v průběhu anotačního procesu chceme anotátorům ukazovat pouze informace, které potřebují pro svoje rozhodnutí. Běžně by měl anotátor přístup omezený na jeden dokument, se kterým pracuje. V naší aplikaci můžeme aplikovat i přísnější omezení na určitou část dokumentu, kterou nazveme *anotační okno*.

Díky používání anotačních oken můžeme v některých doménách textových kolekcí ještě více omezit informaci, kterou anotátor získá v průběhu anotačního procesu. Samotná velikost okna je ale závislá na osobě anotátora (někomu můžeme věřit více a někomu méně) a na doméně dané datové kolekce. Tento přístup by nám u některých kolekcí mohl teoreticky umožnit i využívat anotátory, které nemáme tolik prověřené, protože z jednotlivých anotačních oken nemusí mít možnost získat celou informaci o dokumentu.

Speciální roli pak musí mít administrátor, který nahrává dokumenty. Pokud měl přístup k celému dokumentu, tak nedává smysl, aby ho anotační rozhraní omezovalo nějakým zmenšeným oknem s částí dokumentu.

Jelikož v aplikaci pracujeme s osobními údaji, potřebujeme při návrhu a implementaci dodržovat vhodné bezpečnostní zásady. Můžeme totiž narazit na to, že některá API nejsou bezpečná (např. parsování podvrženého XML v některých knihovnách) a některé špatné programátorské návyky mohou způsobovat bezpečnostní chyby (např. SQL injection).

Pro automatickou kontrolu takových bezpečnostních chyb slouží různé statické analyzátory kódu. Pro Python proto využíváme nástroje Bandit, který automaticky najde některé známé bezpečnostní problémy.

Kromě bezpečnostních chyb nám může pomoci, obzvláště u jazyků, které se dopředu nepřekládají, používat statické analyzátory na hledání různých syntaktických a sémantických chyb v kódu. Pro Python máme k dispozici například PyLint a flake8. Podle mého názoru nelze nějak objektivně říci, který analyzátor je obecně lepší. V naší aplikaci se využívá flake8, protože má lepší podporu v rámci repositáře pro správu kódu.

Ve klientské části aplikace potřebujeme mít možnost autentizovat uživatele. Pro dodržení dobrých bezpečnostních návyků potřebujeme u uživatele ukládat pouze hash hesla spojená s nějakým generovaným řetězcem (často označovaným jako „sůl“). U šifrovacích algoritmů platí více než kde jinde pravidlo, že

„není dobré znovu vymýšlet kolo“, proto pro implementaci využíváme doporučenou (vestavěnou) knihovnu z Flasku, která pro autentizaci využívá algoritmus `pbkdf2:sha256`. V tomto algoritmu kromě přidání soli několikrát aplikujeme hašovací funkci. Tím dostaneme lepší ochranu proti útokům využívající předpočítanou tabulku výsledků hašovacích funkcí (rainbow table attack) [23].

Některé operace vyžadují přístup k určité funkcionalitě i bez přihlášení. Pro tuto funkci se ve webových aplikacích používají dočasné přihlašovací tokeny.

Přihlašovací tokeny můžeme implementovat stavově a bezstavově. Při stavové variantě musíme nalézt spolehlivý způsob, jak ukládat tokeny do databáze. Další možností je vytvářet tokeny bezstavové. To znamená, že v samotném tokenu jsou uloženy všechny potřebné informace. Integritu uložených informací pak hlídá připojený digitální podpis.

Do obou variant můžeme přidat časový údaj o vytvoření tokenu, pomocí kterého můžeme hlídat platnost. Podle povahy funkcionality, kterou chceme zpřístupnit tokenem se u obou způsobů implementace dá zajistit ochrana proti opakovanému použití stejného tokenu. Stavové tokeny mají drobnou nevýhodu v podobě režie nutné pro ukládání do databáze. Naše aplikace proto využívá bezstavové tokeny. Při jejich implementaci používáme knihovnu It’s dangerous, která do Flasku přidává potřebnou podporu pro digitální podpisy a jejich kontrolu.

# 2. Evaluace rozpoznávačů pojmenovaných entit

Pro hledání osobních údajů využíváme rozpoznávače pojmenovaných entit. V následující kapitole popíšeme, jestli se pro tento účel rozpoznávače opravu hodí. Určíme, který rozpoznávač tento účel splňuje nejlépe a navrhneme (a ověříme) postup, kterým vylepšíme výsledky, které od rozpoznávače získáme.

## 2.1 Popis datové kolekce

Pro samotné měření budeme využívat textovou datovou kolekci popisující schůzky, která vznikla v projektu ELITR [5]. Samotná kolekce je členěná po schůzkách a každá schůzka obsahuje dva druhy textů:

- *Transkripty*, které jsou tvořeny doslovnými přepisy komunikace mezi osobami, které se schůzky účastnili.
- *Zápisy ze schůzek* (v originále „minutes“), který shrnuje důležité události, které se v rámci schůzky udály. Tento přepis vždy vychází z transkriptu.

Z domény kolekce obsahují texty zejména seznam účastníků, jména řečníků, názvy projektů a třeba i místa, kde se schůzky konaly.

Tato data jsou použita i pro otevřenou vědeckou soutěž (tzv. shared task) [6], proto vznikla potřeba mít možnost tato data sdílet, aniž by zároveň došlo ke sdílení osobních údajů. Pro sdílenou úlohu bylo třeba zachovat vazby mezi původními entitami (jako mezi odkazy na tutéž osobu; mezi informacemi, které popisují jeden projekt; mezi místy), bylo přistoupeno k pseudonymizaci této kolekce.

Tím vznikla větší datová kolekce, kde ke každému původnímu souboru s osobními údaji existuje i soubor druhý, který je pseudonymizovaný.

Schůzky a porady v datové kolekci popisují reálná setkání z projektů a konferencí. Větší část zúčastněných osob je z ciziny, takže většina textů je v angličtině. Pouze několik schůzek bylo v češtině (viz Tabulka 2.1). Výrazná převaha angličtiny pro potřeby našeho měření nevadí, protože pro český jazyk existuje pouze jeden rozpoznávač pojmenovaných entit, a to *NameTag* [14].

| **Jazyk** | **Počet textů** | **Počet slov**[a] |
|---|---|---|
| angličtina | 95 | 347 875 |
| čeština | 4 | 25 513 |

*Pozn:* [a] Posloupnost znaků oddělená bílým znakem.

Tabulka 2.1: Jazykové složení datové kolekce

Samotná pseudonymizace probíhala na jaře 2021, aby data byla v předstihu k dispozici pro sdílenou úlohu. Naše aplikace byla v té době ještě rozpracovaná, takže se pro anotace využívaly jiné způsoby. Z části byla pseudonymizace prováděna pomocí úpravy textů v textovém editoru a z části byla prováděna pomocí

testovacího webového anotačního rozhraní. V našich měřeních budeme používat data z tohoto rozhraní, protože tato data byla k dispozici podstatně dříve.

V obou případech anotátor dostal k dispozici celý dokument v editovatelném textovém poli a postupně označoval úseky v textu a přímo v textu je nahrazoval pseudonymizovaným označením. Úseky se vždy označovaly po znacích, takže občas samozřejmě došlo i k chybě anotátora, který neoznačil správně celá slova.

Data z webového rozhraní, která my využíváme, jsou anotována pomocí XML značek. Protože bylo anotační rozhraní pouze testovací, tak se v některých případech setkáváme s nevalidním XML. Při anotování nebyly ošetřeny „XML tagy“, které obsahoval původní dokument ve výrazech jako je `<smích>`. To vedlo pravděpodobně k potřebě některých XML knihoven tyto značky uzavírat na konci dokumentu. V některých případech pak v dokumentu byly v XML atributu použity další „vnořené XML dokumenty“. Někdy se pak dokonce v dokumentech objevovaly dva přístupy pro reprezentaci nových řádků — buď pomocí znaku pro nový řádek (tj. `\n`), nebo pomocí HTML značky (tj. `<br>`). Většinu těchto chyb bylo možné automaticky (až poloautomaticky) opravit, takže v měřených dokumentech se tyto chyby už poté nevyskytovaly.[1]

Aby porovnávání rozpoznávačů dávalo smysl, převážně nás zajímají dokumenty, ve kterých anotátoři nalezli více osobních údajů. Protože ale některé schůzky byly kratší, některé soubory neobsahovaly tolik osobních údajů.

Někdy je menší množství označených osobních údajů způsobeno chybou anotátora, protože anotátor část souboru nedokončil nebo soubor neobsahoval celou schůzku kvůli nějaké chybě v průběhu vytváření transkriptů. Tyto soubory bylo pak nutné řešit a opravovat ručně. Občas nastává zajímavá situace, kdy krátký soubor obsahuje nějaký méně běžný údaj (například nějakou přezdívku). Takový případ pak ovlivňoval úspěšnost rozpoznávačů při pohledu po souborech. Z tohoto důvodu se proto u některých druhů měření tyto soubory nepoužívají.

## 2.2 Popis porovnávaných rozpoznávačů

Pro rozpoznávání pojmenovaných entit jsme se rozhodli vyzkoušet tři nástroje a to NameTag [14], spaCy [15] a Stanford Named Entity Recognizer [16] (při využití tokenizátoru NLTK [20]). U všech těchto nástrojů nalezneme podporu angličtiny a v případě NameTagu i podporu češtiny.

Rozpoznávače pojmenovaných entit pracují nad tokenizovaným textem. Tokenizací rozdělujeme text na menší úseky, které dohromady tvoří nějakou ucelenou sémantickou jednotku. Většinou při tokenizaci dělíme text na věty a na „slova“.

Tokenizace je v nejjednodušším případě prováděna pomocí hledání bílých znaků mezi slovy. Případně pomocí hledání teček na koncích vět (pro dělení na věty). Tento přístup ale selže u složitějších textů jako je:

```
Mr. Smith didn't win the prize.
```

Už za první zkratkou „Mr.“ můžeme vidět, že tečka v tomto případě nedělí věty. Zajímavá situace pak nastává i u slova „didn't“, kde není na první pohled zřejmé, jak vypadá správná tokenizace (existuje například i validní výsledek „did

[1] Tyto dokumenty nejsou ani součástí úvodní Tabulky 2.1.

not“). Z tohoto důvodu existují i složitější přístupy, jak tokenizaci provádět za pomocí regulárních výrazů, různých slovníků tokenů a případně pomocí různých jazykově závislých pravidel.

Pro účely našeho měření používáme tokenizátory, které jsou doporučovány pro daný rozpoznávač pojmenovaných entit, abychom rozpoznávačům poskytovali tokenizaci, kterou očekávají. Tímto přístupem se snažíme minimalizovat vliv tokenizace na měřené výsledky.

### 2.2.1 NameTag

Při našich měřeních budeme pro hledání pojmenovaných entit využívat různé rozpoznávače. Prvním rozpoznávačem, který popíšeme, je *NameTag* [14]. NameTag pro rozpoznávání využívá předem natrénované modely, které jsou vždy závislé na zvoleném jazyce. Jako jediný rozpoznávač pak podporuje rozpoznávání pojmenovaných entit v českém jazyce. V případě českého jazyka dokáže nalézt 46 typů atomických jmenných entit a 4 typy složených jmenných entit. Složená jmenná entita tvoří obal nějaké skupiny atomických jmenných entit. Pro český jazyk tedy rozpoznávač nalézá i vnořené jmenné entity. Přesný popis všech podporovaných typů jmenných entit nalezneme v Obrázku 2.1.

V případě anglického jazyka nalezneme díky NameTagu 4 typy jmenných entit a to konkrétně entity pro osoby (`PER`), organizace (`ORG`), lokalizace (`LOC`) a ostatní (`MISC`).[2]

Pro vyhledávání se uvnitř využívají Long Short-Term Memory ve spojení s Conditional Random Fields pro anglický jazyk a pro český jazyk je zvolena metoda sequence-to-sequence z rekurentních neuronových sítí.

Pro tokenizaci textu NameTag využívá pro oba jazyky nástroj *UDPipe* [26]. Ten tokenizaci provádí pomocí „klasifikace dělících znaků“ přes obousměrnou rekurentní neuronovou síť s využitím *Gated Recurrent Unit* [26]. Model pro UDPipe, který sloužící k tokenizaci, je součástí modelů pro rozpoznávač.

### 2.2.2 spaCy

Dalším rozpoznávačem pojmenovaných entit je *spaCy* [15]. Ten je vytvořen jako modulární nástroj pro více lingvistických úloh. Pro rozpoznávání pojmenovaných entit v anglickém jazyce nástroj rozpoznává 18 typů jmenných entit. Pro jejich vyhledávání se používají jazykově závislé modely, které využívají rekurentní neuronové sítě. Nástroj podporuje nepřekrývající se pojmenované entity a při trénování modelu bylo upřednostněno nalézání správné délky úseků jmenných entit.[3]

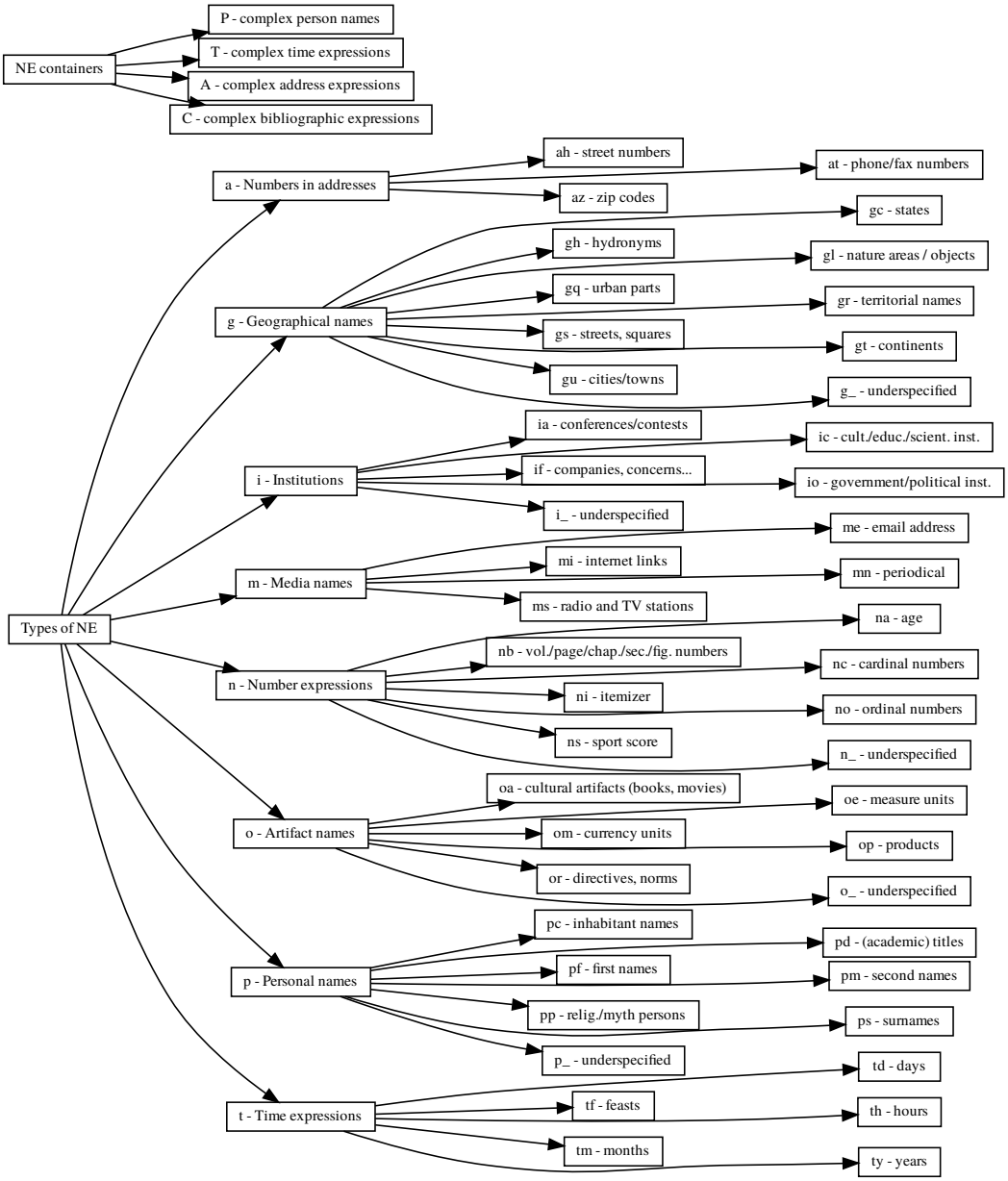
SpaCy používá pro tokenizaci ve výchozím stavu svůj vlastní vestavěný nástroj, který využívá určitá jazykově závislá pravidla.

### 2.2.3 Stanford Named Entity Recognizer (NER)

Stanford NER také známý jako *CRFClassifier* je implementace rozpoznávače pojmenovaných entit v Javě. V angličtině rozpoznává pojmenované entity ve

[2]Při trénování tohoto rozpoznávače se využíval korpus „CoNLL-2003 NER annotations“ [25], ze kterého plynou podporované typy. Stejný korpus nalezneme i u Stanford NERu.

[3]Pro vyhledávání citlivých informací bychom pro naše účely spíše upřednostnili, kdyby nástroj vždy rozpoznal alespoň libovolnou část jmenné entity.




Obrázek 2.1: Typy pojmenovaných entit pro český jazyk v NameTagu [24]

třech typech (obdobně jako NameTag) a to konkrétně osoby, organizace a lokace. Vnitřně k vyhledávání využívá jazykově závislé modely využívající lineární řetězce Conditional Random Fields [16].

Stanford NER nemá přímé API pro Python, takže pro práci s ním využíváme *Natural Language Toolkit* (NLTK). Z NLTK pak využíváme i doporučený tokenizér, kterým v době měření byl `TreebankWordTokenizer`. Ten pro tokenizaci užívá regulární výrazy, které byly vytvořeny podle korpusu Penn Treebank [20].

## 2.3 Metriky měření

Při hledání citlivých informací potřebujeme určit metriky, pomocí kterých budeme jednotlivé rozpoznávače porovnávat. Běžně se rozpoznávače pojmenovaných

entit porovnávají pomocí (kombinace) dvou kriterií:

- Nalezení správného úseku textu, kde se pojmenovaná entita nachází.
- Určení správného typu pojmenované entity pro nalezený úsek.

Podle zvolené metriky se pak například započítávají pouze nálezy, které splňují správně obě dvě kritéria. My z testovací datové kolekce ale nezískáme přímo typ citlivé informace. Získáme pouze informaci o úsecích textu, kterou anotátor označil jako „soukromou“.

Z tohoto důvodu při našem měření nebudeme brát v úvahu, jaký typ má daná pojmenovaná entita. Tím zároveň vyřešíme i problém, kdy se v různých rozpoznávačích liší druhy (a množství) typů pojmenovaných entit, které rozpoznávače rozpoznávají.

Z legislativních požadavků je naší povinností převážně ochraňovat osobní údaje a zabránit jejich případnému úniku. Proto pro nás představuje prioritu nalezení, co největšího množství osobních údajů, abychom zabránili případným zneužitím. V rámci kontextu naší aplikace to znamená, že potřebujeme detekovat, co nejvíce „podezřelých slov“, takže potřebujeme detekovat širokou škálu typů pojmenovaných entit.

Naše aplikace pak umožňuje (za použiti automatického pravidla) určit nějaký typ pojmenovaných entit jako „veřejný“. To prakticky pro anotátora znamená, že se tímto typem nemusí už více zabývat při dalších anotacích.

### 2.3.1 Vliv kvality tokenizace

Protože každý rozpoznávač používá jiný nástroj na tokenizaci, můžeme se setkat se situací, kdy se jednotlivé tokenizace budou při vyhodnocování lišit. Použitá testovací datová kolekce neobsahuje informace o správné tokenizaci, proto nemáme ani žádné správné (zlaté) hodnoty, které bychom mohli používat při vyhodnocování.

Rozpoznávače pojmenovaných entit zároveň různě přistupují k označování pojmenovaných entit, které se skládají z více tokenů. NameTag a spaCy poskytuje informaci, jestli daná posloupnost tokenů tvoří souvislou pojmenovanou entitu, ale Stanford NER tuto informaci neposkytuje.

Abychom vyřešili rozdílné vlastnosti tokenizace, budeme porovnávat jednotlivé nálezy podle absolutní pozice v dokumentu. Absolutní pozici v dokumentu definujeme jako počet znaků od začátku dokumentu do začátku (případně konce) pojmenované entity. Zároveň, abychom získali představu, jak dobře nalezené pojmenované entity odpovídají anotacím anotátora z testovacích kolekce, zavedeme při vyhodnocování 3 druhy shody:

- **Přesná** — označuje nález rozpoznávače pojmenovaných entit, který se přesně shoduje v začáteční a koncové pozici v dokumentu s úsekem, který anotátor označil jako „soukromý“. Tyto výsledky jsou nejvhodnější pro automatická pravidla, protože anotátor nemusí provádět žádnou úpravu, aby získal správný výsledek.
- **Uvnitř** — označuje nález rozpoznávače pojmenovaných entit, který obklopuje nějakou citlivou anotaci podle anotace od anotátora. Pokud budou

tyto nálezy řešeny pouze automatickými pravidly, tak nám neunikne žádná citlivá informace, ale je možné, že z textu odebereme některá slova navíc.

- **Částečná** — označuje nález rozpoznávače pojmenovaných entit, který částečně pokrývá nějaký úsek citlivých informací od anotátora. Část úseku ale není pokryta, takže pro nalezení všech citlivých informací musí anotátor ručně rozšířit označený úsek. Rozpoznávač pojmenovaných entit ale v tomto případě správně najde část textu, který je třeba podrobit „větší kontrole“.

Libovolná z těchto kategorií je dostačující, aby anotátor obdržel úsek textu, který vyžaduje nějaké jeho rozhodnutí. První kategorie představuje minimální množství práce pro anotátora, další dvě vyžadují určitou ruční úpravu délky úseku. Pokud citlivá informace není nalezena přímo touto cestou, tak je nutné jí dohledat průchodem celého textu.[4]

Bez ohledu na druh shody budeme při měření používat dvě metriky pro porovnávání výsledků rozpoznávačů pojmenovaných entit.

- **Citlivost** pro nás bude označovat procento citlivých údajů, které označil anotátor jako citlivé a pro které rozpoznávač pojmenovaných entit našel nějakou shodu.

$$\text{citlivost} = \frac{\text{Počet nalezených shod}}{\text{Celkový počet citlivých údajů}}$$

  Zjednodušeně tedy toto číslo říká, kolik citlivých údajů jsme schopni najít automaticky.

- **Přesnost** pro nás bude označovat procento údajů, které anotátor skutečně označil jako citlivé z údajů, které rozpoznávač pojmenovaných entit našel.

$$\text{přesnost} = \frac{\text{Počet nalezených shod}}{\text{Celkový počet nalezených pojmenovaných entit}}$$

  Z tohoto čísla jsme zjednodušeně schopni určit kolik práce musí anotátor udělat navíc.

Při měření budeme rozlišovat dva pohledy na výpočet měřených metrik. Testovací datová kolekce popisuje schůzky z různých druhů událostí. Tyto události se liší svojí délkou a tématem. Některá témata mohou být pro rozpoznávače pojmenovaných entit obtížnější. Proto zavedeme dva pohledy na výsledky:

- **Po schůzkách** je pohled na výsledky, kde je vyhodnocení prováděno pro každý soubor (schůzku) zvlášť. Pohledy na tato data nám pomohou, jak se výsledky rozpoznávače liší nad různými soubory (tj. tématy schůzek). Tento pohled nám tedy ilustruje, jak je rozpoznávač univerzální na různá témata.

- **Globální** pohled na výsledky, kde výhodnocení provádíme nad všemi daty bez ohledu na soubor (schůzku), ze kterého pocházejí.

Pohledem na data po schůzkách můžeme určit, jak rozpoznávače pracují nad různými druhy schůzek bez ohledu na různá zastoupení délek schůzek s odlišnými tématy.

[4] V sekci 2.6 si ukážeme, že jsme schopni dohledat další (nenalezené) citlivé informaci automaticky díky rozhodnutím v průběhu anotačního procesu.

## 2.4 Příprava měřených hodnot

Pro potřeby našeho měření tedy vytvoříme několik pomocných souborů. Pro každý textový soubor z testovací datové kolekce vyextrahujeme citlivé informace spolu se souvisejícími pseudonymizovanými označeními (viz Tabulka 2.2). Tím získáme přehled o všech citlivých informacích v testovací datové kolekci.

Ukázkové tabulky v následující sekci jsou vytvořeny (pro přehlednost) nad částí jednoho ze souborů z testovací datové kolekce.

| První znak | Poslední znak | Citlivá informace | Označení |
|---:|---:|---|---|
| 2 | 7 | `ELITR` | `[PROJECT2]` |
| 94 | 100 | `Ondřej` | `[PERSON5]` |
| 700 | 708 | `- Martin`[a] | `[PERSON3]` |

*Pozn:* [a] Označený interval vznikl chybou anotátora.

Tabulka 2.2: Ukázka části vstupní tabulky pro jeden soubor z testovací datové kolekce

Celkově v naší datové kolekci takto nalezneme 39 696 citlivých údajů. Citlivým údajem v tomto případě rozumíme souvislý úsek textu, který označil anotátor za citlivý.

Každý rozpoznávač pojmenovaných entit spustíme na všechny soubory z naší testovací datové kolekce a pro každý rozpoznávač pojmenovaných entit vytvoříme jeden soubor s nálezy. Každý řadek poté popisuje úsek textu, který tvoří pojmenovanou entitu a typ pojmenované entity, který rozpoznávač určil (viz Tabulka 2.3).

| První znak | Poslední znak | Pojmenovaná entita | Typ |
|---:|---:|---|---|
| 2 | 13 | `ELITR Surge`[a] | `ORG` |
| 94 | 100 | `Ondřej` | `PER` |
| 702 | 708 | `Martin`[b] | `PER` |

*Pozn:* [a] Rozpoznávač nalezl (chybně) delší úsek.

*Pozn:* [b] Rozpoznávač nalezl správný úsek, ač anotátor označil chybně delší.

Tabulka 2.3: Ukázka části výsledků z NameTagu pro jeden soubor z testovací datové kolekce

Od rozpoznávačů NameTag a spaCy přímo dostáváme informaci, kolik tokenů tvoří jednu pojmenovanou entitu. V případě Stanford NERu tuto informaci nemáme, takže použijeme určité zjednodušení — více tokenů bude tvořit jednu pojmenovanou entitu v případě, že rozpoznávač detekoval více entit za sebou a všechny mají stejný typ (pojmenované entity). Protože při vyhodnocení měříme i shodu v podobě průniku a vnoření, tak vliv tohoto zjednodušení umíme na výsledcích případně popsat.[5]

Pomocí testovacích souborů od anotátora (obsahující označené citlivé informace) a výsledků jednotlivých rozpoznávačů (obsahující pojmenované entity) vytvoříme souhrný soubor, který popisuje výsledky jednotlivých rozpoznávačů pro

[5]Podle měření toto zjednodušení výrazně výsledky neovlivnilo.

jednotlivé citlivé údaje. Pro každý citlivý údaj vytvoříme jeden řádek, který popisuje druh shody s příslušnou pojmenovanou entitou pro jednotlivé rozpoznávače (viz Tabulka 2.4).

| Úsek | | Citlivý údaj | NameTag | spaCy | StanfordNer |
|---|---|---|---|---|---|
| 2 | 13 | `ELITR` | Uvnitř | — | — |
| 94 | 100 | `Ondřej` | Přesná | Přesná | Přesná |
| 702 | 708 | `- Martin` | Částečná | — | Částečná |
| 3 257 | 3 263 | `Ondřej` | — | — | Přesná |

Tabulka 2.4: Ukázka části souhrnu shod pro jeden soubor z testovací datové kolekce

Pro některé citlivé informace rozpoznávač nenajde žádnou korespondující pojmenovanou entitu, takže pro tyto dvojice nedostaneme ani žádnou shodu.

Z těchto výsledných souhrnů pak následně vytvoříme celkový souhrn přes celou testovací datovou kolekci. Ten obsahuje shrnutí počtu citlivých údajů a nalezených druhů schod od rozpoznávačů pojmenovaných entit (viz Tabulka 2.5). Každý řádek popisuje jednu dvojici testovacího souboru a rozpoznávače pojmenovaných entit.

| Soubor | NER | C | P | U | Č | E |
|---|---|---|---|---|---|---|
| `meeting006_en-transcript` | NameTag | 83 | 62 | 3 | 2 | 105 |
| `meeting006_en-transcript` | StanfordNer | 83 | 33 | 2 | 1 | 39 |
| `meeting006_en-transcript` | spaCy | 83 | 55 | 4 | 2 | 101 |

*Pozn:* **C** — počet citlivých údajů; **P** — počet přesných shod; **U** — počet shod uvnitř; **Č** — počet částečných shod; **E** — počet pojmenovaných entit

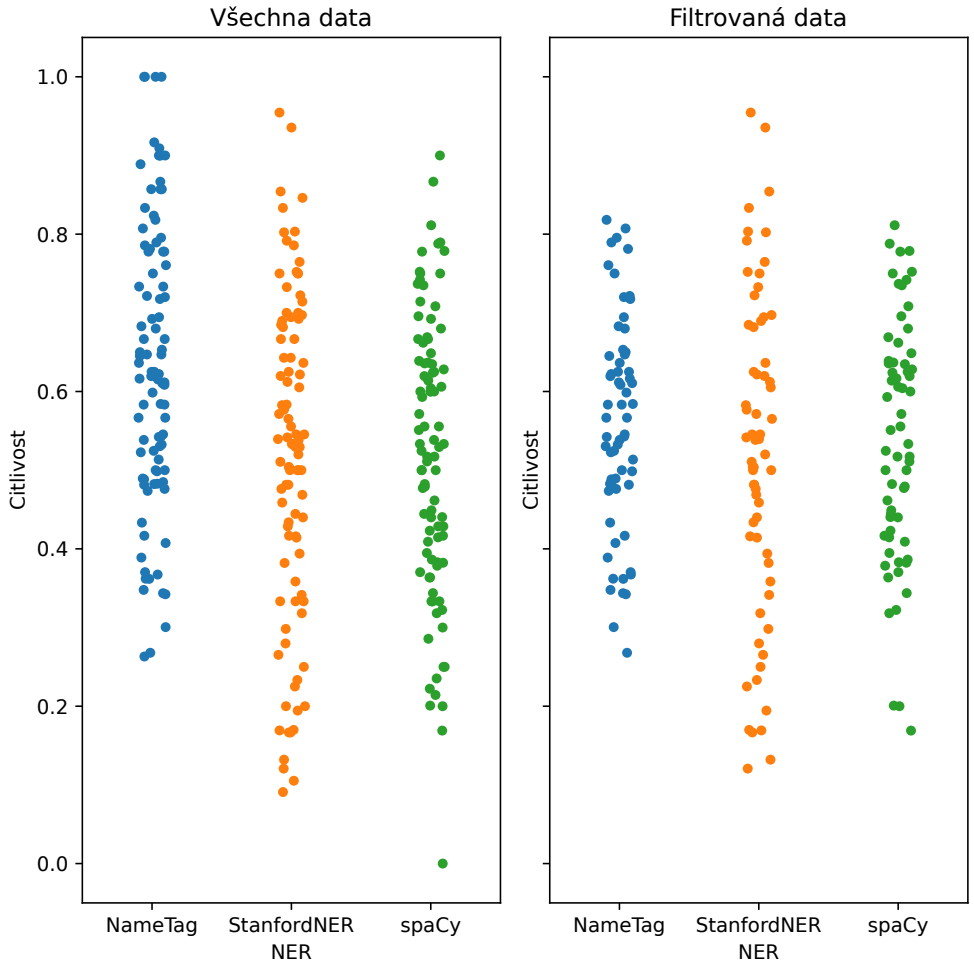
Tabulka 2.5: Ukázka části souhrnu pro celou testovací datovou kolekci

## 2.5 Výsledky porovnání

V následující sekci popíšeme výsledky jednotlivých rozpoznávačů pojmenovaných entit podle metrik, které jsme si zavedli v předchozí části. Pro určení citlivosti a přesnosti rozpoznávačů nad různými druhy schůzek budeme nejprve postupovat „po schůzkách“. Výsledky každé schůzky si můžeme představit jako nějaký nezávislý experiment. Vyhodnocováním po schůzkách získáme větší představu o kvalitě vstupních dat a o fungování rozpoznávačů nad různými schůzkami.

### 2.5.1 Citlivost po schůzkách

Nejprve určíme pro každý testovací soubor citlivost všech rozpoznávačů při započítávání libovolného druhu shody (tj. uvažujeme přesné, vnořené i částečné shody). Tím určíme, kolik citlivých informací rozpoznávač najde pomocí pojmenovaných entit. Výsledky si ukážeme v Obrázku 2.2 pomocí *Strip Plotu*. Strip Plot je druh grafu, kde pro každý výsledek zaneseme jeden bod a lze na něm díky tomu dobře ilustrovat, jak se liší výsledky pro různé soubory a rozpoznávače.



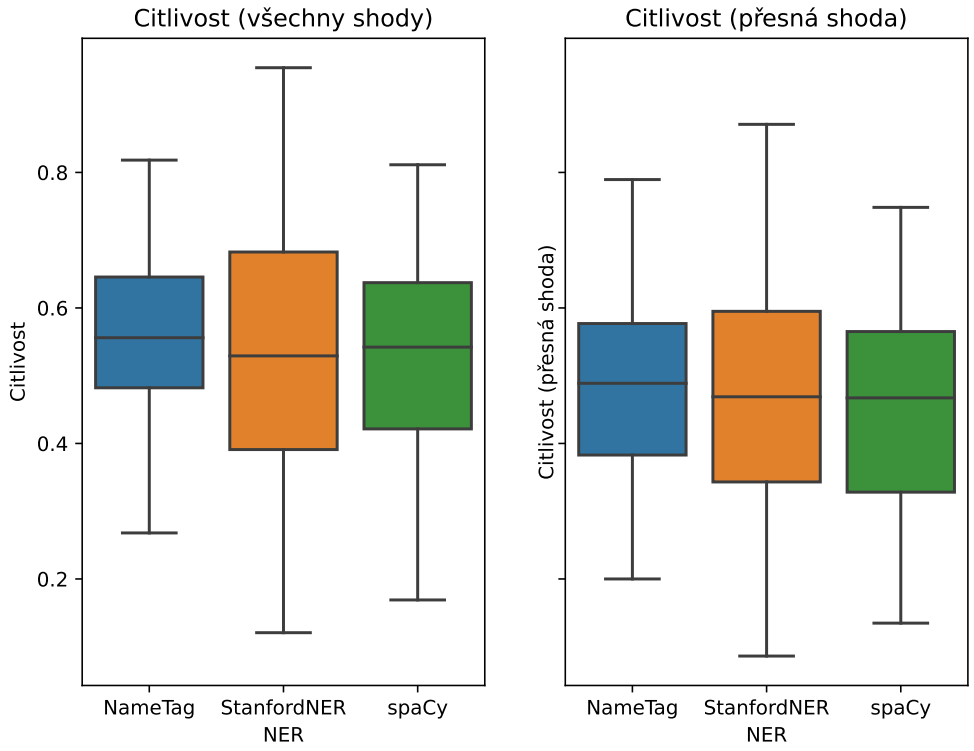

Obrázek 2.2: Citlivost rozpoznávačů pojmenovaných entit po schůzkách

U několika souborů pro každý rozpoznávač nalezneme nějaké extrémní hodnoty z hlediska citlivosti na obou koncích stupnice. Při bližší analýze těchto souborů zjistíme, že tyto soubory jsou kratší a obsahují jen několik málo citlivých informací. Některé schůzky obsahovaly pak třeba i nějakou netypickou citlivou informaci (např. název projektu bez nějakého pomocného kontextu). V jednom případě dokonce takto nalezneme špatně anotovaný soubor.

V pravé části Obrázku 2.2 jsme tyto kratší schůzky odfiltrovali, abychom minimalizovali jejich vliv. Schůzku jsme odfiltrovali pokud obsahovala méně než 20 citlivých informací. Výsledek tohoto odfiltrování můžeme pak také pozorovat v Obrázku 2.3.

Pro následující Obrázek 2.3 využíváme *Box Plot*. Tento graf přímo znázorňuje pomocí konců obdélníku první a třetí kvartil, čára uvnitř obdélníku ukazuje medián (druhý kvartil). Pomocí tohoto grafu můžeme často pozorovat i takzvané odlehlé body (častěji označované *outliery*). Outliery znázorňují nějaké extrémní hodnoty (občas se tyto hodnoty považují za místa, kde mohlo dojít k chybě měření) [27]. Protože graf vykreslujeme už nad odfiltrovanými daty, žádné outliery nemáme.[6] Podle grafů můžeme pozorovat, že NameTag dosahuje nejvyšší citli-

[6]Při vykreslování nad všemi daty se několik outlierů objevilo. Příklad outlierů nalezneme v Obrázku 2.4.




Obrázek 2.3: Citlivost rozpoznávačů pojmenovaných entit podle různých druhů shod po schůzkách

vosti, protože jeho medián je nejvyšší a nemá ani příliš velký rozptyl, takže jeho výsledky jsou nad různými schůzkami nejvíce konzistentní. Stanford NER oproti tomu příliš konzistentních výsledků nedosahuje. Existují testovací soubory, kde sice dává lepší výsledky než ostatní, ale v jiných případech je zase podstatně horší.

Zajímavou alternativou k NameTagu je pak spaCy, které má sice trochu horší výsledky, ale má svobodnou licenci pro použití oproti modelům NameTagu, které jsou pouze pro nekomerční využití.

V pravém grafu je dále znázorněno, jak se sníží citlivost rozpoznávačů, pokud zvýšíme nároky na nalezenou shodu a budeme započítávat pouze pojmenované entity s přesnou shodou s citlivou informací z testovacího souboru. Všechny rozpoznávače s tímto požadavkem dosahují horších výsledků, ale toto zhoršení není výrazné a nemění ani pořadí rozpoznávačů. Můžeme říci, že je u všech rozpoznávačů docela srovnatelné.

Průměrnou citlivost (po souborech) pro jednotlivé rozpoznávače pojmenovaných entit můžeme také porovnat v Tabulce 2.6.

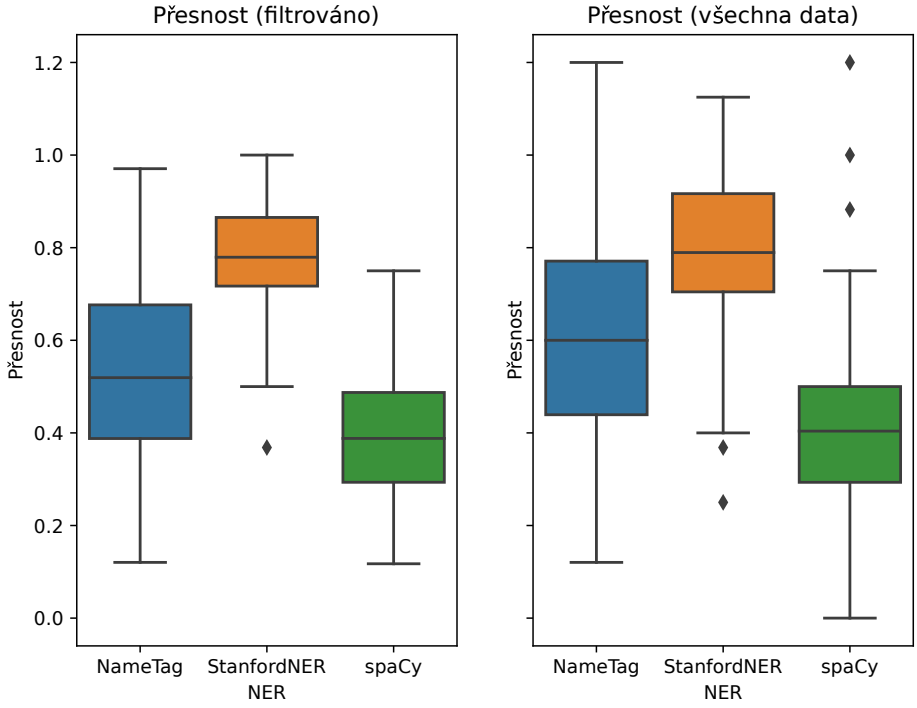
| **Rozpoznávač** | **Libovolná shoda**[a] | **Přesná shoda**[a] |
|---|---|---|
| NameTag | 0,638 (0,182) | 0,554 (0,179) |
| StanfordNer | 0,517 (0,205) | 0,454 (0,194) |
| spaCy | 0,519 (0,178) | 0,404 (0,177) |

*Pozn:*[a] Průměrná citlivost, standardní odchylka v závorce.

Tabulka 2.6: Citlivost rozpoznávačů pojmenovaných entit po schůzkách

### 2.5.2 Přesnost po schůzkách

Kromě citlivosti potřebujeme určit i přesnost rozpoznávačů pojmenovaných entit. Přesnost nám říká, kolik nalezených pojmenovaných entit tvoří zároveň citlivou informaci. Tím si ilustrujeme, kolik „práce navíc“ musí anotátor při svém rozhodování udělat. U přesnosti už můžeme pozorovat vliv tokenizace (a správného určení úseku pojmenované entity). Pro některé testovací soubory totiž přesnost přesahuje 100 %, protože v některých případech může jedna nalezená pojmenovaná entita obsahovat více citlivých informací. Tuto situaci můžeme nejvíce pozorovat na Obrázku 2.4 v rámci nefiltrovaných dat, kde k tomu napomáhá menší množství citlivých informací.




Obrázek 2.4: Přesnost rozpoznávačů pojmenovaných entit po schůzkách

Nejvíce přesný z námi porovnávaných rozpoznávačů je Stanford NER. Pokud se podíváme na výsledky citlivosti, tak byl ale zase nejhorší. Tento rozpoznávač tedy nenalezne nejméně citlivých informací, ale největší část pojmenovaných entit, které nalezne, tvoří citlivou informaci. NameTag a spaCy mají podobné výsledky z pohledu citlivosti, ale z pohledu přesnosti je NameTag lepší. Z porovnání po souborech tedy vychází nejlépe NameTag. Podrobné výsledky pro citlivost rozpoznávačů můžeme porovnat v Tabulce 2.7.

| **Rozpoznávač** | **Přesnost**[a] |
|---|---|
| NameTag | 0,602 (0,230) |
| StanfordNer | 0,787 (0,167) |
| spaCy | 0,417 (0,185) |

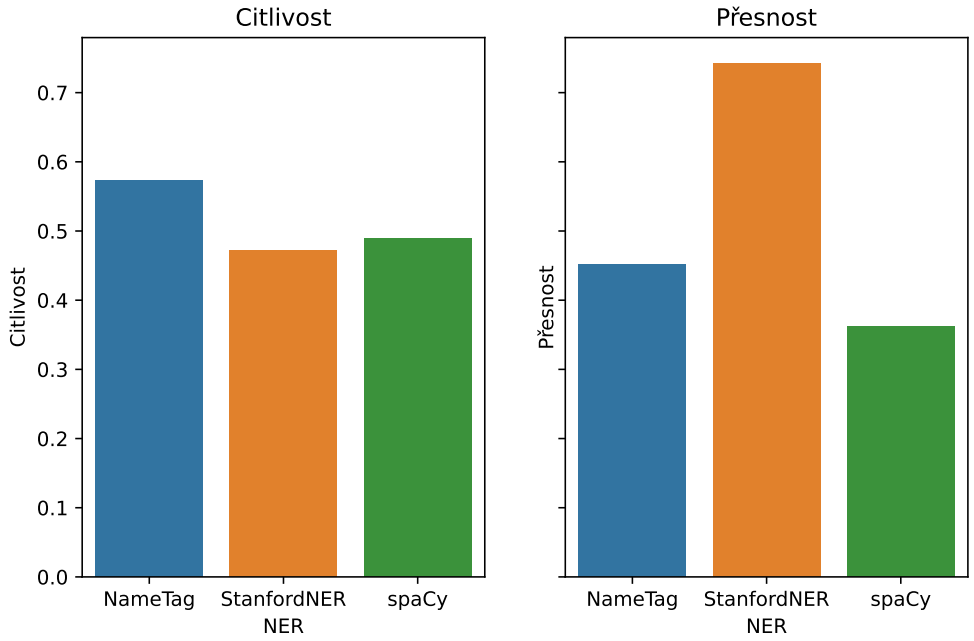
*Pozn:*[a] Průměrná citlivost, standardní odchylka v závorce.

Tabulka 2.7: Přesnost rozpoznávačů pojmenovaných entit po schůzkách

Obecně je možné zlepšovat citlivost na úkor přesnosti a naopak. Pro naši aplikaci je ale podstatně významnější citlivost, protože vyšší citlivost napomůže k nalezení více úseků s citlivou informací, díky čemuž můžeme zabránit úniku těchto údajů. Proto je pro nás NameTag celkovým vítězem, protože v přesnosti sice dosahuje průměrných výsledků, ale dosahuje nejvyšší citlivosti.

### 2.5.3 Globální výsledky

Kromě měření po schůzkách porovnáme rozpoznávače i v globálním měřítku bez ohledu na jednotlivé testovací soubory (schůzky). V tomto případě při porovnávání použijeme všechny citlivé informace přes všechny dokumenty. Do výsledků budeme započítávat všechny druhy shod mezi pojmenovanými entitami a citlivými informacemi.




Obrázek 2.5: Globální výsledky přes celou testovací datovou kolekci

Jak můžeme vidět na Obrázku 2.5, v globálním měřítku dosahuje nejvyšší citlivosti vyhledání citlivých informací NameTag. NameTag sice zaostává v přesnosti za Stanford NER, ale ten má pak zase nejnižší citlivost při hledání citlivých informací. Z pohledu naší aplikace je kvůli legislativním požadavkům mnohem důležitější nalezení všech citlivých informací před snížením množství falešně označených „podezřelých slov“. Podrobné výsledky rozpoznávačů pojmenovaných entit nalezneme v Tabulce 2.8.

Pro porovnání vztahu mezi přesností a citlivostí můžeme využít tzv. *F1 skóre*. Toto číslo určíme jako harmonický průměr „přesnosti“ a „citlivosti“:

$$F_1 = \frac{2}{\text{přesnost}^{-1} + \text{citlivost}^{-1}}$$

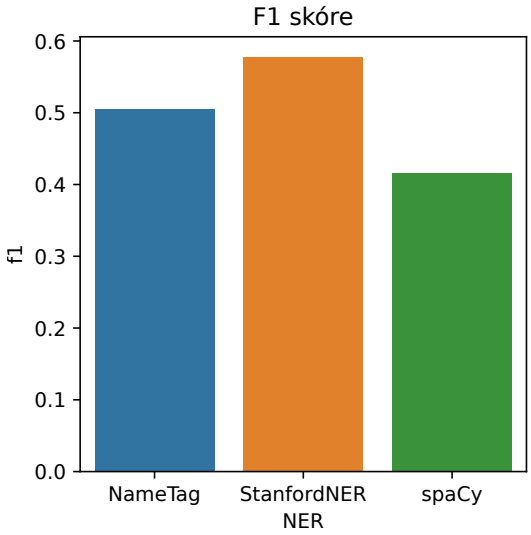
To, jak jsme si zavedli přesnost a citlivost v našich měřeních, ale plně nekoresponduje s nejběžnějším užitím. Navíc z pohledu naší aplikace je důležitější

| Rozpoznávač | Citlivost (ls)[a] | Citlivost (ps)[b] | Přesnost |
|---|---|---|---|
| NameTag | 0,573 | 0,495 | 0,452 |
| StanfordNer | 0,472 | 0,429 | 0,742 |
| spaCy | 0,489 | 0,419 | 0,363 |

*Pozn:* [a] libovolná shoda, [b] přesná shoda

Tabulka 2.8: Globální výsledky přes celou datovou kolekci

samotná citlivost než poměr mezi citlivostí a přesností. V Obrázku 2.6 můžeme pozorovat, jak se liší rozpoznávače z hlediska poměru obou vlastností. Podle F1 skóre má nejlepší výsledek Stanford NER.



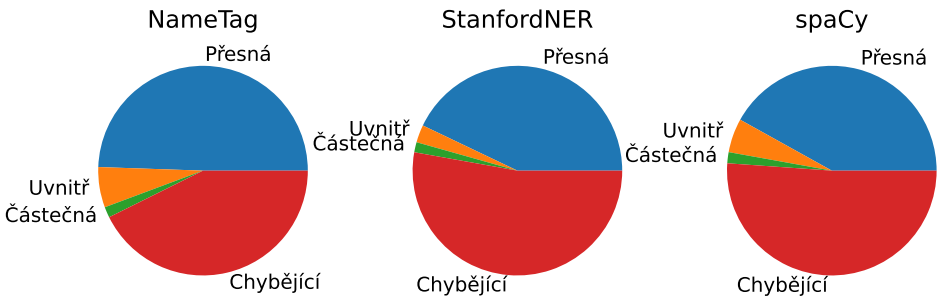

Obrázek 2.6: Porovnání globálních výsledků podle F1 skóre

### 2.5.4 Shoda úseků pojmenovaných entit s úseky citlivých informací

Kromě nalezení citlivé informace můžeme ještě rozpoznávače pojmenovaných entit porovnávat podle toho, jak dobře vyhledávají konkrétní úseky citlivých informací. Pro toto porovnávání jsme si už v dřívější části zavedli 3 druhy shody (přesnou, vnořenou a částečnou). Na Obrázku 2.7 můžeme pozorovat, jaké druhy shod u jednotlivých rozpoznávačů převažují. Pro úplnost jsme do grafu přidali i chybějící citlivé údaje (tj. citlivé údaje, které se neprotínají s žádnou nalezenou pojmenovanou entitou).

Špatná detekce úseku je často spojena s místy, kde tokenizátor naráží na znak nového řádku. V naší testovací datové kolekci můžeme nalézt dva druhy souborů — soubory s transkripty a soubory s poznámkami ze schůzek. Právě v poznámkách ze schůzek se nový řádek používal pro oddělení jednotlivých „odrážek“ popisující jednotlivé události ve schůzce.

S dělením informací po řádcích některé tokenizátory nepočítají, takže považují slova na dalším řádku za pokračování předchozí věty, což pak ovlivňuje rozhodnutí




Obrázek 2.7: Porovnání druhů shody citlivých údajů s pojmenovanou entitou

rozpoznávače. Toto chování můžeme nalézt na některých výsledcích NLTK, které se používá pro tokenizaci v rámci Stanford NER.

Jak už bylo vidět z globálních výsledků, tak NameTag produkuje nejvíce shod s pojmenovanými entitami. Můžeme dále pozorovat, že spaCy a NameTag mají podobné množství pojmenovaných entit, které obklopují nějaké citlivé údaje. Stanford NER má oproti těmto dvěma rozpoznávačům nejmenší množství vnořených citlivých informací. Podobnou vlastnost jsme u něj pozorovali v souvislosti s přesností, kde dosahoval nejlepších výsledků. Dle mého názoru tedy můžeme říci, že Stanford NER převážně najde jen jednoznačnější pojmenované entity tvořící citlivé informace.

Podrobné výsledky, jak dopadly jednotlivé rozpoznávače pojmenovaných entit, pro různé druhy shody můžeme porovnávat v Tabulce 2.9.

| **Rozpoznávač** | **Přesná** | **Uvnitř** | **Částečná** |
|---|---|---|---|
| NameTag | 3272 | 412 | 109 |
| StanfordNer | 2839 | 180 | 103 |
| spaCy | 2774 | 347 | 113 |

Tabulka 2.9: Porovnání druhů shody citlivých údajů s pojmenovanou entitou

## 2.6 Vylepšení vyhledávání citlivých informací

Podle výsledků měření testovací datové kolekce jsme schopni s nejcitlivějším rozpoznávačem pojmenovaných entit (NameTag) identifikovat 57,3 % citlivých údajů pomocí vyhledávání pojmenovaných entit.

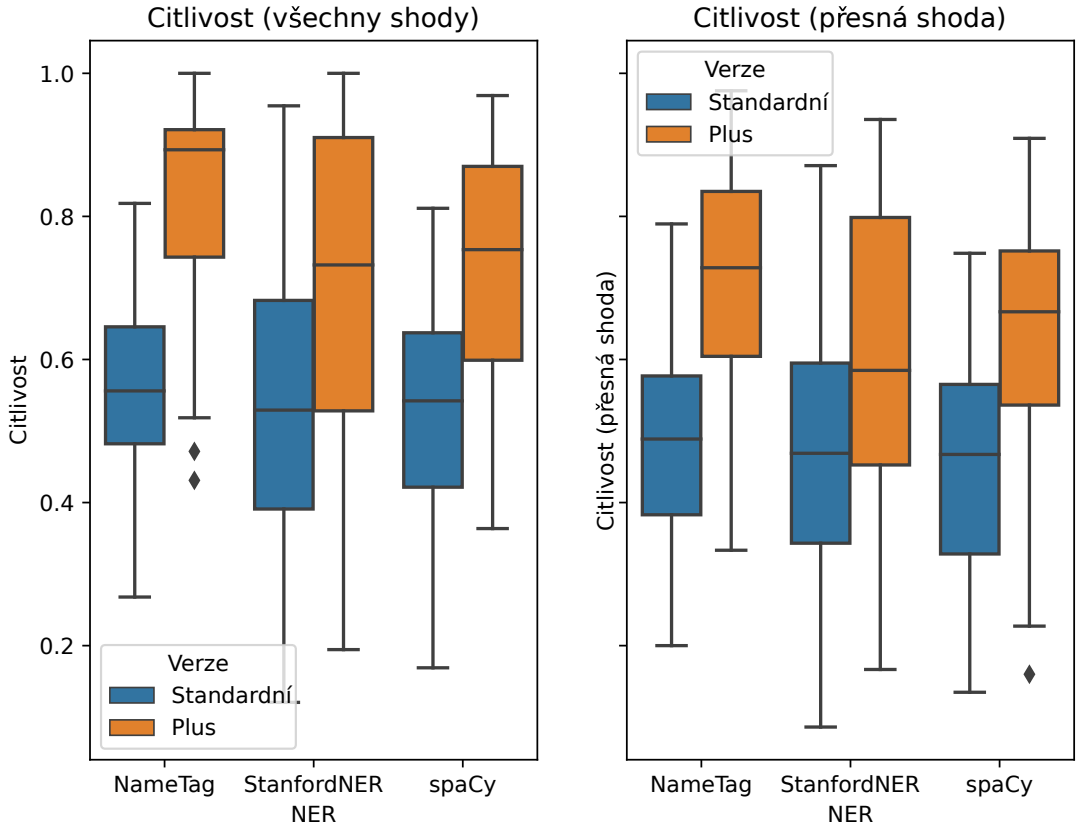
V některých případech tedy citlivou informaci díky samotnému rozpoznávači nenalezneme automaticky. V Tabulce 2.4 (kde ilustrujeme výstup z rozpoznávače pojmenovaných entit) můžeme pozorovat jednu takovou situaci. V dáném případě rozpoznávač pojmenovaných entit nalezl nějakou citlivou informaci v rámci jednoho kontextu, ale v jiném kontextu se mu už tutéž citlivou informaci nalézt nepodařilo.

Tato slova se dokonce shodují „podle typu“, což vede k úvaze: „Jak se zlepší výsledky měření, pokud budeme označovat pojmenované entity z jednoho kontextu i v ostatních kontextech.“ Pro účely našeho měření si zaznamenáme všechny nalezené pojmenované entity v daném dokumentu a poté budeme označovat jejich

další výskyty i v jiných částech dokumentu. Dalším výskytem rozumíme shodu „podle typu“.[7]

### 2.6.1 Výsledky vylepšeného algoritmu po schůzkách

Na Obrázku 2.8 můžeme pozorovat, jak tento přístup vylepší citlivost vyhledávání citlivých informací. V následujících grafech budeme vylepšený algoritmus označovat jako verzi „plus“ a původní algoritmus, který fungoval pouze s rozpoznávačem pojmenovaných entit, budeme označovat jako „standardní“ verzi.



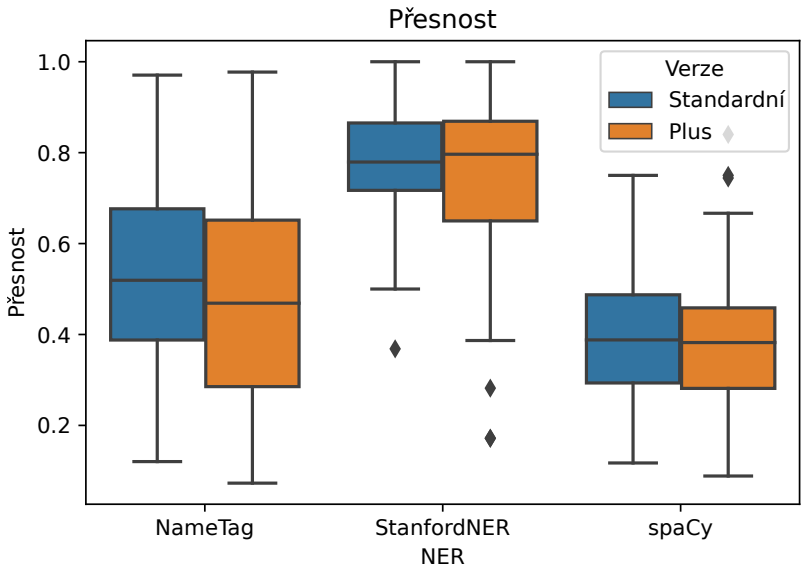

Obrázek 2.8: Vylepšená citlivost po souborech

Z grafů můžeme pozorovat, že pro všechny rozpoznávače pojmenovaných entit tento vylepšený algoritmus znamená podstatné zlepšení. Pořadí rozpoznávačů podle citlivosti se ani s vylepšeným algoritmem nezměnilo, ale rozdíly mezi rozpoznávači jsou více patrné. Citlivější rozpoznávače totiž najdou pojmenované entity ve více případech, takže při hledání bez kontextu máme více informací o pojmenovaných entitách, které je možné při dalším hledání využívat.

Zajímavou otázkou je, jak vylepšený algoritmus ovlivňuje přesnost. To můžeme pozorovat na Obrázku 2.9. Díky vyhledávání pojmenovaných entit bez kontextu získáme větší množství nálezů, což se musí projevit na množství falešně pozitivních označení.

Je patrné, že pro všechny rozpoznávače pojmenovaných entit se zvětšil rozptyl hodnot. Pro NameTag a spaCy přesnost klesla, ale v případě Stanford NERu se

[7]Podobný algoritmus jsme zavedli v sekci 1.4, pro vytváření automatických pravidel po rozhodnutí na úrovni tokenu.

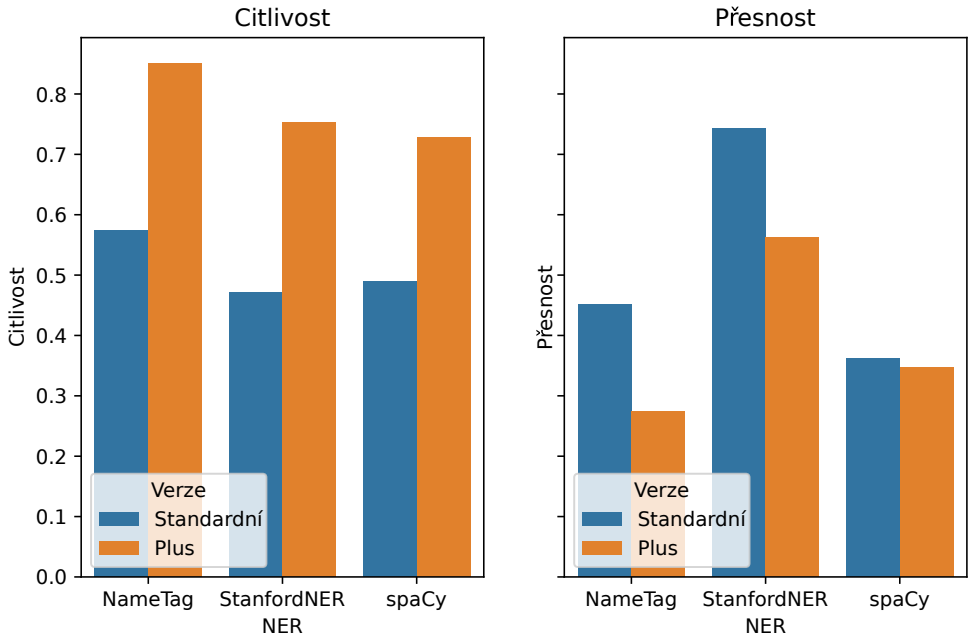



Obrázek 2.9: Vylepšená přesnost po souborech

lehce zvýšila. To je dle mého názoru způsobeno tím, že model Stanford NERu obecně ve všech měřeních vykazoval spíše vyhledávání kvalitnějších výsledků (z pohledu přesnosti a druhu shody) před kvantitou.

### 2.6.2 Globální výsledky vylepšeného algoritmu

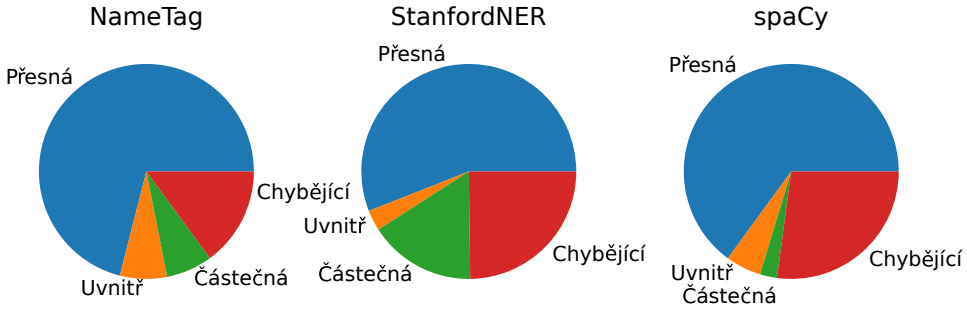
Kromě lepších výsledků „po schůzkách“ můžeme pozorovat lepší výsledky při globálním pohledu na celou testovací datovou kolekci (viz Obrázek 2.10). Pro všechny rozpoznávače pojmenovaných entit vylepšená verze algoritmu způsobila podstatné zvýšení citlivosti.




Obrázek 2.10: Vylepšené globální výsledky

Průměrná přesnost pak poklesla, protože pokud pojmenovaná entita ani v původní verzi algoritmu netvořila citlivou informaci, tak její vyhledávání v dalších kontextech pravděpodobně nenalezlo novou citlivou informaci.

Při porovnání přesnosti shody mezi úseky citlivých informací a pojmenovaných entit z vylepšeného algoritmu došlo u všech rozpoznávačů ke snížení množství nenalezených úseků. NameTag i spaCy zachovávají ve vylepšeném algoritmu vysoké procento přesné shody, Stanford NER ve vylepšeném algoritmu nalézá velkou část nových citlivých informací pouze částečně. Stanford NER už v původní verzi nenalezne tolik citlivých informací, proto při vyhledávání pojmenovaných entit v dalších kontextech, často využívá pouze částečnou informaci (např. z kombinace jména a příjmení původně nalezne pouze jméno). Ve výsledcích má proto také nejmenší podíl shod uvnitř. Podle výsledků se proto domnívám, že Stanford NER dosahuje obecně horších výsledků u víceslovných pojmenovaných entit. Výsledky měření pro všechny rozpoznávače můžeme porovnat na Obrázku 2.11.




Obrázek 2.11: Porovnání druhů shody citlivých údajů s pojmenovanou entitou pro vylepšený algoritmus

I s vylepšenou verzí algoritmu stále dosahujeme v obou druzích měření nejlepších výsledků NameTag. Má nejvyšší citlivost, která je pro nás prioritní. Podrobné výsledky, jak dopadly jednotlivé algoritmy pro všechny porovnávané rozpoznávače ve všech verzích algoritmu můžeme nalézt na konci kapitoly v souhrné Tabulce 2.10.

### 2.6.3 Použití vylepšení v rámci naší aplikace

Jak se ukázalo z výsledků měření, tak vylepšená verze algoritmu zvýšila citlivost nejlepšího rozpoznávače (NameTag) na 85 % vzhledem k datům z testovací datové kolekce. Nevýhodou tohoto vylepšeného přístupu je ale nižší přesnost oproti standardní verzi.

V naší aplikaci můžeme využít ještě jedno vylepšení, které zkombinuje vylepšenou citlivost se snahou o zachování původní přesnosti ze standardního algoritmu. Tento výsledek získáme, pokud budeme dohledávat citlivé informace v dalších kontextech až pokud anotátor označí pojmenovanou entitu (v terminologii automatických pravidel „podezřelé slovo“) jako „soukromé“.

Díky tomuto přístupu ušetříme dohledávání některých pojmenovaných entit, které vůbec neoznačují nějakou citlivou informaci. Tím získáme vylepšenou citlivost jen s částečným snížením přesnosti.

| **Rozpoznávač** | **Citlivost** | | **Přesnost** | **Počet shod podle druhu** | | | |
|---|---|---|---|---|---|---|---|
| | Libovolná shoda | Přesná shoda | | Přesná | Vnořená | Částečná | Chybějící |
| NameTag | 0,573 | 0,495 | 0,452 | 3272 | 412 | 109 | 2823 |
| NameTag+ | 0,850 | 0,710 | 0,273 | 4699 | 468 | 462 | 987 |
| StanfordNer | 0,472 | 0,429 | 0,742 | 2839 | 180 | 103 | 3494 |
| StanfordNer+ | 0,752 | 0,560 | 0,563 | 3704 | 207 | 1065 | 1640 |
| spaCy | 0,489 | 0,419 | 0,363 | 2774 | 347 | 113 | 3382 |
| spaCy+ | 0,729 | 0,650 | 0,347 | 4298 | 354 | 170 | 1794 |

Tabulka 2.10: Shrnutí výsledků pro rozpoznávače pojmenovaných entit

# 3. Vývojová dokumentace

V následující kapitole si popíšeme anotační aplikaci z vývojářského pohledu. Nejprve ukážeme členění aplikace, formát ukládaných dat a poté představíme základní algoritmy a principy, které aplikace používá. Pro označování naší aplikace budeme v názvech balíčků používat zkratku `psan`.

Aplikace používá pro ulehčení vývoje a spouštění nástroj *Make*. Spouštění aplikace pro ladění a produkci nalezneme v Příloze B.

## 3.1 Konvence používané ve zdrojovém kódu

Pro pojmenování tříd, proměnných a balíčků v kódu pro Python používáme standardní konvence, které vychází z PEP 8 [28]:

- Pro odsazování používáme 4 mezery.
- Maximální délka řádků je omezena na 130 znaků.
- Importy jsou na samostatných řádcích. Zároveň jsou importy rozděleny pomocí prázdných řádků do 3 skupin. V první skupině nalezneme importy ze standardní knihovny, ve druhé skupině importy externích knihoven a ve třetí skupině importy jiných částí našeho kódu. V každé skupině jsou pak importy řazeny podle abecedy.
- Pro pojmenování tříd používáme *CamelCase* (např. `MyClass`).
- Pro pojmenování proměnných užíváme malá písmena s podtržítky (např. `line_number`) a pro konstanty velká písmena s podtržítky (např. `MAX_INT`).
- Pro názvy proměnných a komentáře používáme angličtinu.
- Pro práci s výčtovými typy z databáze (`ENUM`) a pro ostatní textové konstanty využíváme pouze dopředu definované konstanty z modulu `model`.

Kontrola základních konvencí je součástí automatických testů. Tuto kontrolu můžeme kdykoliv vynutit pomocí příkazu `make lint` z kořenové složky projektu.

## 3.2 Členění aplikace

Samotná aplikace je členěna na několik logických částí. Každá část je zaměřena na jednu hlavní funkci:

- **Webové rozhraní** je část aplikace, se kterou interaguje uživatel. Backend webového rozhraní používá *Flask*. Pro generování HTML stránek využíváme šablonovací nástroj *Jinja2*. Frontend využívá kombinaci *Boostrapu* a *jQuery* a surového JavaScriptu.
- **Služby na pozadí** se starají o události, které vyžadují netriviální čas běhu. K této úloze se využívá asynchronního přístupu pomocí knihovny *Celery*.

- **Automatická knihovna** pak označuje část aplikace, která obsahuje automatickou anotační logiku. Pro čistotu návrhu a případnou znovupoužitelnost i v jiných projektech je tato část navržena tak, aby mohla případně fungovat i samostatně.

Kromě těchto částí aplikace využívá pro ukládání strukturovaných dat databázi v *PostgreSQL*. Pro správu fronty úloh pro služby na pozadí aplikace používá distribuovanou databázi klíčů *Redis*.

### 3.2.1 Členění podle místa nasazení

Aplikace je ve výchozí konfiguraci spouštěna uvnitř Docker kontejneru. Tento přístup totiž napomáhá lepší přenositelnosti do jiných prostředí. Aplikace ale stejně tak může fungovat i bez kontejnerizace.

Pro své fungování aplikace vyžaduje několik externích služeb jako je například databáze. Součástí práce jsou tři konfigurace pro nasazení aplikace za použití kontejnerizace, které zároveň řeší napojení těchto externích služeb. Tyto konfigurace jsou definovány pro nástroj *Docker Compose*.

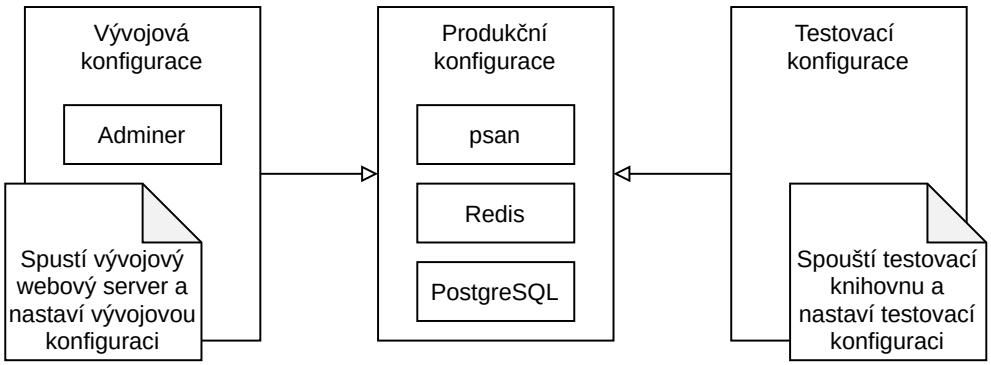
- V **režimu pro vývoj** (debug) se aplikace spouští s vývojovým webovým serverem, který je vestavěný ve Flasku. V tomto režimu web server zobrazuje výjimky ve webovém rozhraní a ve výstupu do konzole. Pro zjednodušení vývoje má webová aplikace plný přístup pro čtení a zápis k celému souborovému systému. Zdrojový kód je připojen do kontejneru přes bind mount a při změně kódu aplikace se změněná část aplikace dokonce sama restartuje. Díky tomuto přístupu můžeme testovat změny, aniž by bylo třeba znovu vytvářet nebo restartovat Docker kontejner.

- V **produkčním režimu** aplikace používá webový server *uWsgi*, který je určený pro produkční nasazení a podporuje běh i ve více vláknech. V případě chyby webový server vrací `HTTP 500` a chybu zapíše pouze do výstupu z konzole. V tomto režimu je webová aplikace spuštěna jako speciální uživatel, který má omezená práva k souborovému systému. Tento uživatel může zapisovat pouze do instanční složky, kam jsou ukládány nahrané dokumenty od uživatele. Díky těmto omezením dochází k ochraně aplikace před třídou útoků, ve kterých se přepisuje zdrojový kód aplikace.

- **Testovací režim** pak slouží ke spuštění automatických testů, které slouží pro otestování základních funkcí aplikace.

Každá konfigurace je definována svým konfiguračním souborem. Produkční konfiguraci nalezneme v kořenu projektu v souboru `docker-compose.yaml`, vývojovou konfiguraci nalezneme v souboru `docker-compose.debug.yaml` a testovací v souboru `docker-compose.test.yaml`. Testovací a vývojová konfigurace dědí konfiguraci z produkční konfigurace (viz Obrázek 3.1).

Jednotlivé konfigurace se liší externími závilostmi, které spouští. Produkční konfigurace spouští hlavní Docker kontejner se samotnou aplikací (tj. webové rozhraní se službami na pozadí a automatickou knihovnou). V dalších kontejnerech je pak spuštěn PostgreSQL a Redis.




Obrázek 3.1: Závislostí mezi režimy spuštění

V testovacím prostředí dochází k používání stejného prostředí pouze s jinou konfigurací pro názvy databáze a logování. Po spuštění kontejneru se zavolá testovací knihovna místo webového serveru.

Ve vývojovém prostředí se spouští vývojový webový server a je přidán další kontejner s programem *Adminer*, který slouží pro spravování databáze přes web.

### 3.2.2 Členění podle balíčků

Jednotlivé části aplikace se dále člení do balíčků. Každý balíček je zaměřen na nějakou hlavní část aplikace:

- **`psan`** označuje balíček tvořící hlavní část kódu. V tomto balíčku jsou v modulech popsány jednotlivé části uživatelského prostředí.
- **`psan.tool`** je balíček tvořící automatickou knihovnu pro pseudonymizaci. Moduly v tomto balíčku definují společná rozhraní a třídy. Obsahují pak i definice typů proměnných pro výměnu dat s ostatními částmi aplikace.
- **`psan.tool.tasks`** obsahuje úlohy tvořící části hlavního anotačního algoritmu.
- **`psan.celery`** obsahuje všechny služby, které běží na pozadí. Služby z tohoto balíčku volají postupně jednotlivé části hlavního anotačního algoritmu.
- **`config`** obsahuje konfigurace Flasku pro vývojový a produkční režim. Dále obsahuje společnou konfiguraci.
- **`tests`** obsahuje Unit testy pro testování základní funkčnosti aplikace.

### 3.2.3 Adresářová struktura aplikace

Projekt je rozdělen do několika adresářů podle běžných konvencí z Flasku. Kromě adresářů, které odpovídají jednotlivým balíčkům, najdeme v projektu další složky a soubory. V kořenu projektu nalezneme spouštěcí skripty, konfiguraci pro Make, sestavovací skripty pro Docker, konfiguraci pro Docker Compose a konfigurační soubor pro webový server uWsgi. Dále je projekt členěn do několika dalších složek:

- **db** obsahuje skripty pro vytvoření tabulek a definice typů procedur a triggerů pro databázi.
- **instance** je výchozí umístění instanční složky (v případě neexistence jí vytváří automaticky Make). Obsah této složky je specifický pro danou instanci aplikace (pro dané nasazení aplikace). Ve složce můžeme najít instanční konfiguraci a ve výchozím nastavení i uživatelská data (nahrané dokumenty).
- **ner_eval** obsahuje skripty pro zpracování testovací datové kolekce a skripty pro generování grafů pro porovnávání rozpoznávačů pojmenovaných entit.
- **psan/static** obsahuje statické soubory přístupné z webu (jako kaskádové styly a obrázky).
- **psan/templates** obsahuje šablony, ze kterých se generují `HTML` stránky pomocí Jinja2.

Při vytváření Docker image se nepřenáší celá adresářová struktura obsahující všechny vývojové soubory, ale jen potřebné části. Databázové skripty a skripty na porovnávání rozpoznávačů pojmenovaných entit a podobné soubory výsledný image neobsahuje.

V rámci vývojové konfigurace se dynamicky připojuje celá adresářová struktura, aby vzniklo propojení mezi kontejnerem a hostitelským souborovým systémem (používá se tzv. bind mount).

## 3.3 Reprezentace dat

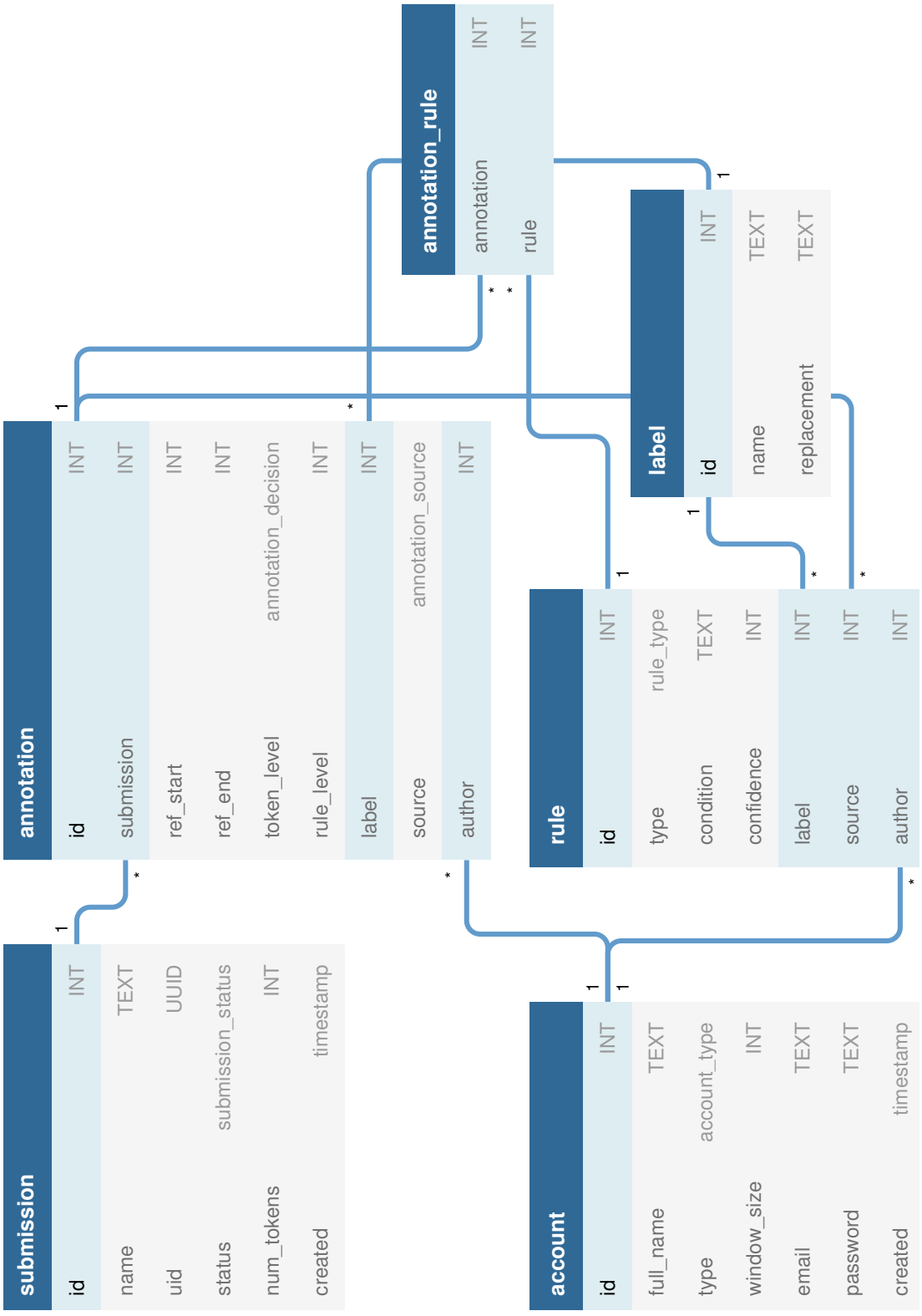
Aplikace ukládá data pomocí dvou způsobů. Textová data jsou ukládána do textových souborů v souborovém systému a strukturovaná data jsou ukládána do databáze.

Každý dokument určený pro pseudonymizaci je uložen v textové formě do interní složky a je o něm vytvořen patřičný záznam v databázi. Dokument je poté předzpracován (proběhne tokenizace a rozpoznání pojmenovaných entit). Výsledky tohoto předzpracování jsou uloženy do souboru ve formátu XML. Každý token je očíslován vzestupně pro možnost pozdější indexace z ostatních datových struktur.

V průběhu anotačního procesu se anotace ukládají do databáze. Každá anotace popisuje rozhodnutí o nějakém úseku tokenů (textu). Na anotace se váží pseudonymizační názvy a automatická pravidla. Automatická pravidla určují podmínky pro automatické vytvoření nějaké anotace. Podrobný průběh anotačního algoritmu popíšeme v sekci 3.4.

### 3.3.1 Popis databáze

Pro ukládání strukturovaných dat využíváme databázi v PostgreSQL. Aplikace využívá několik tabulek, které jsou vzájemně propojeny pomocí cizích klíčů (viz Obrázek 3.2).




Obrázek 3.2: Schéma databáze

- **account** popisuje údaje o uživatelských účtech v aplikaci. Uživatelé se dělí na dva druhy — na standardní uživatele a na administrátory. Administrátor má neomezený přístup ke všem souborům a funkcím v aplikaci. Může do aplikace provádět importy a exporty dat. Standardní uživatel oproti tomu má možnost pouze provádět anotace v rámci aktuálně přiděleného anotačního okna.

- **submission** obsahuje údaje o nahraných dokumentech. O každém dokumentu je uchováván jeho název a fáze dokumentu z hlediska anotačního procesu.

- **label** obsahuje informace o pseudonymizačních označeních. Každé označení má svůj název (pro identifikaci mezi ostatími) a text používaný při nahrazování.

- **annotation** obsahuje informace o anotacích v textu. Anotace mohou pocházet od anotátora, nebo od automatických pravidel. Každá anotace je určena dokumentem, ve kterém se nachází a pozicí v tomto dokumentu. Anotace obsahuje informace o rozhodnutích na úrovni tokenu a rozhodnutích na úrovni pravidla. Kromě rozhodnutí pak anotace mohou obsahovat pseudonymizační označení na úrovni tokenu.

- **rule** obsahují automatická pravidla. Pravidla jsou několika typů. Typ pravidla určuje interpretaci podmínky pravidla (`condition`). Rozhodnutí pravidla se udává pomocí míry spolehlivosti (`confidence`), která definuje váhu pravidla, pokud více pravidel lze aplikovat na jeden úsek textu.[1] Při změně spolehlivosti se automaticky přepočítává hodnota rozhodnutí podle pravidla v navázaných textových anotacích.

  Některá pravidla mohou být navázaná na určitou anotaci (např. pravidla popisující „podezřelá slova“ vzniklá podle rozhodnutí na úrovni tokenu). Některá pravidla také mohou být spojena s nějakým pseudonymizačním označením, které má být při jejich aplikaci použito.

- **annotation_rule** obsahuje vazby mezi textovými anotacemi a automatickými pravidly. Při změně této vazby se automaticky přepočítává hodnota rozhodnutí podle pravidla v navázaných anotacích.

V databázi využíváme pro sloupečky s omezeným množstvím hodnot výčtové datové typy (`ENUM`). Pro reprezentaci těchto hodnot v Pythonu využíváme `Enum` objekty definované v modulech `model`. Tím je aplikace připravena na případnou pozdější změnu těchto typů. Pro zachování integrity dat se využívají i další mechanizmy z databází, jako jsou unikátní sloupečky a omezující podmínky (constraints).

### 3.3.2 Ukládání do souborů

Pro ukládání textových dat jsou využívány soubory v souborovém systému. Každý dokument má přiřazeno unikátní identifikační číslo (UID). Toto číslo při-

[1] Tuto problematiku popíšeme podrobněji v sekci 3.4.

řazuje každému dokumentu svou unikátní složku. V této složce je uložen původní vstupní soubor a otagovaný soubor ve formátu XML.

Otagovaný soubor přidává do původního dokumentu XML značky, které označují tokeny, věty a úseky pojmenovaných entit. Tokeny se číslují od 0 od začátku souboru. Označení věty a pojmenované entity nesou informaci o rozsahu tokenů, který obklopují. Pojmenovaná entita pak zároveň nese i informaci o svém typu.

Některé znaky se mohou vyskytovat mimo token, pokud je tokenizátor za token neoznačí. Sem patří například znak nového řádku a mezery. Přesnou specifikaci tohoto XML souboru ve formátu DTD nalezneme v následujícím bloku:

```
<?xml encoding="UTF-8"?>

<!ELEMENT submission (sentence)+>

<!ELEMENT sentence (ne|token)+>
<!ATTLIST sentence
  start CDATA #REQUIRED
  end CDATA #REQUIRED>

<!ELEMENT ne (ne|token)+>
<!ATTLIST ne
  start CDATA #REQUIRED
  end CDATA #REQUIRED
  type CDATA #REQUIRED>

<!ELEMENT token (#PCDATA)>
<!ATTLIST token
  id CDATA #REQUIRED>
```

Výsledný pseudonymizovaný dokument se vytváří z anotačních dat a XML souboru přímo za běhu, takže není uložen v souborovém systému.

## 3.4 Anotační algoritmus

Při pseudonymizaci textových dokumentů procházejí dokumenty několika fázemi, ve kterých je dokument postupně zpracováván. V prvních fázích dochází k automatickému zpracování, kdy probíhá analýza dokumentu pomocí lingvistických nástrojů. Po dokončení analýzy následuje tzv. *anotační proces*, ve kterém provádí rozhodnutí anotátor. Postup zpracování dokumentu všemi těmito fázemi budeme označovat *anotační algoritmus*.

Cílem anotačního algoritmu je nalézt citlivé údaje a přiřadit jim označení pro pseudonymizaci. Výsledky hledání a označovaní se ukládají do anotací. Anotace je určena úsekem textu v dokumentu. Anotace můžeme rozdělit podle druhu vzniku do dvou skupin:

- Anotace od uživatele (označované `USER`), které vznikly nějakým rozhodnutím anotátora na úrovni tokenu. Rozhodnutí na úrovni tokenu mají největší váhu, protože anotátor k nim využil kontext.

- Automatické anotace (označované `RULE`), které vznikly aplikací nějakého automatického pravidla. Automatická anotace se mění na tu od uživatele, pokud tuto anotaci uživatel libovolně upraví.

### 3.4.1 Automatická pravidla

Automatické anotace vznikají aplikací automatických pravidel. Anotační pravidla jsou definována typem pravidla a podmínkou pro aplikaci. Anotační algoritmus je navržen tak, aby byl snadno rozšiřitelný o další pravidla. Aktuálně jsou v aplikaci využívána následující pravidla:

- Pravidla podle typu slova slouží pro vyhledávání posloupnosti tokenů, které odpovídají posloupnosti typů slov. V tomto případě podmínka pro aplikaci pravidla obsahuje seznam tokenů.
- Pravidla podle typu pojmenované entity slouží k vyhledávání posloupností tokenů, které tvoří nějakou pojmenovanou entitu určitého typu. Podmínka pro aplikaci pravidla obsahuje vyhledávaný typ pojmenované entity.

Aplikace automatických pravidel probíhá postupně po jednotlivých souborech. Při aplikaci pravidel se využívá otagovaný XML soubor, který obsahuje přidané lingvistické informace. Po přečtení nového tokenu nebo pojmenované entity rozhodneme s využitím databáze pravidel, jestli nás následující úsek může zajímat — tj. jestli v tomto místě dokumentu může začínat nějaké známé pravidlo. Pokud by toto místo mohlo být začátkem místa aplikace pravidla, tak požádáme vnitřní API (viz sekce 3.5.3) o zaznamenání následujícího úseku po určité místo. Vnitřní API po přečtení úseku vyvolá událost, ve které obdržíme požadovaný úsek a podle získaných informací a údajů z databáze pravidel rozhodneme, jestli můžeme nad tímto úsekem aplikovat pravidlo. Pokud můžeme aplikovat pravidlo, tak vytvoříme automatickou anotaci pro tento úsek.

Kromě typu obsahují automatická pravidla míru spolehlivosti, která reprezentuje, jakou váhu dané pravidlo má. Kladnou míru spolehlivosti mají pravidla, která text označují „veřejný“. Zápornou míru mají pravidla, co označují text jako „soukromý“. Tato volba umožňuje pravidlům hlasovat o výsledném rozhodnutí, pokud více pravidel anotuje jeden úsek textu.[2]

Pokud je výsledné rozhodnutí v určeném intervalu kolem 0 (ve výchozí konfiguraci v intervalu od $-1$ do 1), tak výsledná anotace „podle pravidla“ označuje úsek textu za „podezřelý“. O takovém úseku textu by tedy měl rozhodnout anotátor. Tato situace typicky může nastat, pokud podle jednoho pravidla je úsek „veřejný“ a podle druhého pravidla je úsek „soukromý“.

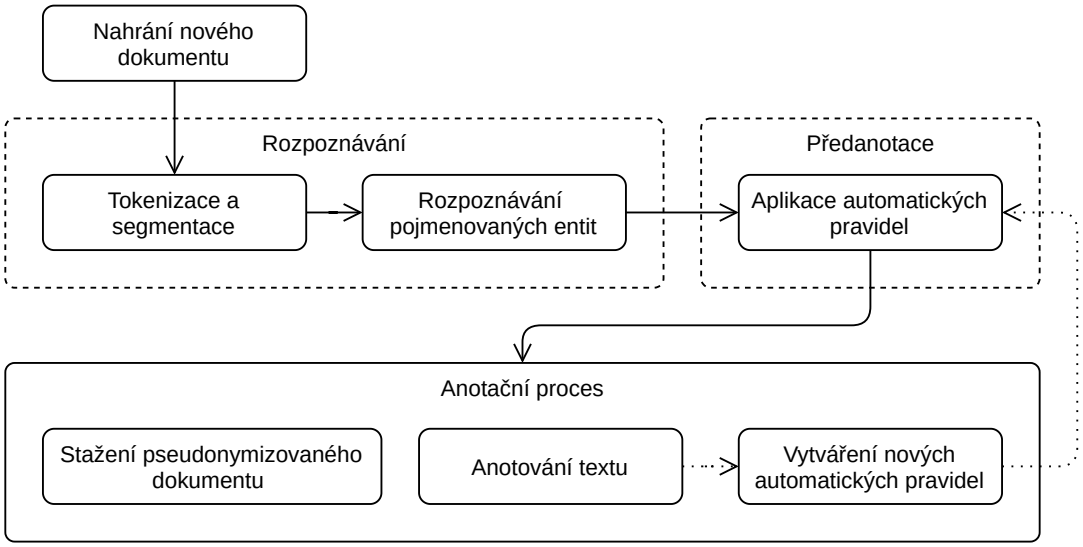
Pokud chceme vytvořit pravidlo pro hledání „podezřelých“ slov, tak jim z tohoto důvodu typicky nastavujeme míru spolehlivosti 0. Tím máme zajištěno, že toto pravidlo neovlivní výsledky existujícím anotacím a anotátor bude muset u nových anotací provést nějaké rozhodnutí.

### 3.4.2 Fáze dokumentu

Každý dokument je vždy v určité fázi anotačního algoritmu bez ohledu na ostatní dokumenty. Rozhodnutí anotátora v jiných dokumentech ale přesto mohou

[2]Rozhodnutí na úrovni tokenu (od anotátora) má vždy nejvyšší váhu.

vyvolávat události i na ostatních dokumentech (viz Obrázek 3.3). Jako příklad můžeme uvést aplikaci nového automatického pravidla vytvořeného při anotaci jednoho dokumentu, které se posléze aplikuje na ostatní dokumenty.




Obrázek 3.3: Fáze anotačního algoritmu

Fáze anotačního algoritmu odpovídají fázím dokumentu. Fáze dokumentu odpovídají konstantám z typu `submission_status` z databáze. Výčtová reprezentace pro Python se nachází v modulu `psan.model`.

- Nový dokument (`NEW`) označuje výchozí fázi dokumentu. Označuje chvíli, kdy byl dokument nahrán do aplikace, ale ještě nebyla dokončena tokenizace a rozpoznávání pojmenovaných entit. V této fázi se vytváří otagovaný XML soubor pro daný dokument.

- Rozpoznaný dokument (`RECOGNIZED`) je fáze, která nastává po dokončení rozpoznávání. V této fázi se analyzuje otagovaný XML soubor, aby se v databázi vytvořila automatická pravidla pro pojmenované entity. Do databáze se přidávají automatické anotace pro „podezřelé“ úseky textu. Součástí této fáze je také aplikace automatických pravidel.

- Po dokončení aplikace automatických pravidel je dokument předanotován (`PRE_ANNOTATED`). V této fázi je dokument přístupný anotátorovi v rámci anotačního procesu.

- Dokument je dokončený (`DONE`), pokud jsou všechny „podezřelé“ úseky textu rozhodnuté (buď automatickým pravidlem nebo anotátorem). Dokumenty v této fázi už anotátor nedostává dále k dispozici v anotačním procesu. Administrátor má k dokumentu přístup vždy.

Pokud dokument vstoupí do předanotační fáze, tak už je možné stáhnout jeho pseudonymizovanou verzi podle aktuálních rozhodnutí. Zbývající nerozhodnuté („podezřelé“) úseky jsou v tomto případně brány jako „veřejné“.

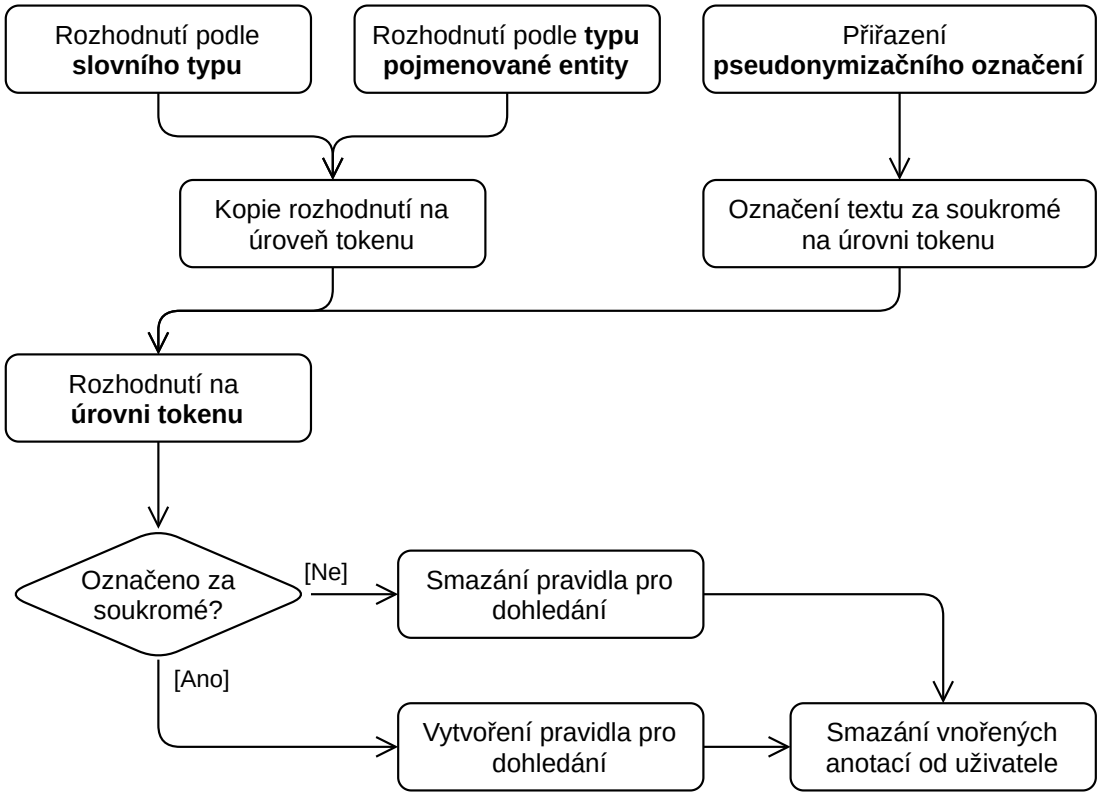
Anotační algoritmus byl navržen tak, aby bylo možné používat libovolné lingvistické nástroje. Ve fázi rozpoznávání je například možné používat libovolný

tokenizátor a rozpoznávač pojmenovaných entit, pokud pomocí adaptéru zvládnou tyto nástroje generovat XML dokument podle specifikace z části 3.3.2.

### 3.4.3 Anotační proces

Po dokončení automatického zpracovávání dochází k anotování dokumentu anotátorem v rámci anotačního procesu. V této části dostává anotátor k dispozici anotační okno, ve kterém by měl rozhodovat o „podezřelých“ slovech a dohledávat neoznačené citlivé informace.

Při anotovaní má anotátor k dispozici 3 druhy rozhodnutí. Pro každý druh rozhodnutí má anotátor na výběr buď označit slova jako „veřejná“, nebo jako „soukromá“. Podle druhu rozhodnutí se následně provedou některé další akce. Tento proces je znázorněn na Obrázku 3.4




Obrázek 3.4: Schéma anotačního procesu

Pokud se v průběhu anotačního procesu označí slova jako „soukromá“, dojde k vytvoření tzv. *dohledacího pravidla*. Toto pravidlo je speciální druh automatického pravidla podle slovního typu, které je navíc svázané s anotací, pro kterou vzniklo. Cílem tohoto pravidla je dohledat stejné slovní typy v jiných částech textu. Navázání na anotaci umožňuje pravidlo odstranit, pokud anotátor později rozhodnutí změní.

Tímto postupem využíváme vylepšeného algoritmu, který jsme navrhli při evaluaci rozpoznávačů pojmenovaných entit. V našem případě jsme ho mírně upravili, protože se slova dohledají, až pokud je anotátor skutečně označí někde za „soukromá“. Tímto postupem omezujeme množství rozhodnutí nad pojmenovanými entitami, které netvoří nějakou citlivou informaci.

Pokud je provedeno rozhodnutí na úrovni tokenu (tj. rozhodnutí s využitím kontextu), tak se nejprve nastaví patřičné rozhodnutí v databázi. Pokud je úsek označený jako „soukromý“, tak dojde zároveň k vytvoření pravidla pro dohledání. Pokud došlo ke změně označení na „veřejné“, tak dojde ke smazání pravidla pro dohledání. Smazáním tohoto pravidla dojde automaticky i k odebrání všech automatických anotací, které toto pravidlo pro dohledání vytvořilo.

Rozhodnutí na úrovni typu (tj. rozhodnutí, které nezávisí na kontextu) nastaví automatické pravidlo podle typu slova pro dané typy slov. Pokud dojde k vytvoření nového pravidla, tak dojde k naplánovaní úlohy pro automatickou aplikaci pravidel. Rozhodnutí na úrovni typu je zároveň dále zpracováno jako rozhodnutí na úrovni tokenu.

Rozhodnutí podle typu pojmenované entity nastaví označení pro dané automatické pravidlo. Všechny anotace pravidla podle pojmenovaných entit se vytvářejí už v rámci automatického zpracování, takže není třeba je dodatečně dohledat. Rozhodnutí podle typu pojmenované entity se také dále zpracovává jako rozhodnutí na úrovni tokenu.

Pokud je nějaký úsek označený jako „soukromý“, nastává potřeba mu přiřadit pseudonymizační označení. Toto označení slouží pro propojení stejných informací v dokumentu, aby získaly stejnou náhradu ve výsledném pseudonymizovaném textu. Pokud označení není svázáno s nějakým pravidlem, musí uživatel vybrat správné označení, případně musí přidat označení nové.

Všechna rozhodnutí provádí anotátor v anotačním okně. Anotační okno označuje úsek textu, který se zobrazí anotátorovi při provádění anotací. Velikost okna je nastavitelná pro každého uživatele. Tímto omezením uživatel nezíská přístup k celému dokumentu.

Při vytváření anotačního okna se vybere náhodně dokument, ze kterého se vybere první „podezřelý“ úsek textu (případně první úsek textu bez pseudonymizačního označení). Kolem tohoto úseku se vytvoří anotační okno. Protože rozhodování anotátora mohou vyžadovat kontext, tak anotační okno je vždy zmenšeno, aby začátek a konec okna odpovídal začátku respektive konci nějaké věty.

## 3.5 Funkční popis

Anotační proces provádí anotátor v uživatelském rozhraní. Uživatelské rozhraní volá další části aplikace jako služby na pozadí a automatickou knihovnu. V následující sekci si popíšeme fungování těchto částí aplikace a jejich vzájemné vazby.

### 3.5.1 Webové rozhraní

Web pro svůj běh využívá Flask, který poskytuje rozhraní pro webový server v Pythonu. Flask umožňuje generovat web pomocí šablon za využití nastroje Jinja2. Jinja2 generuje dokumenty pomocí šablonovacího systému, který podporuje dědičnost, vlastní makra a podmíněné úseky šablony. V naší aplikaci máme takto definovanou šablonu `base.html`, která obsahuje společné části webu. Ostatní šablony pak pouze generují svůj vlastní unikátní obsah.

Ve Flasku využíváme dále knihovnu *Bootstrap-Flask*, která do něj přidává podporu pro Bootstrap 4. Bootstrap je nástroj na vytváření responsivních webů

za použití různých předpřipravených komponent a kaskádových stylů. Knihovna nám převážně umožňuje využívat snadněji Bootstrap při generování HTML stránek ze šablon.

Pro usnadnění generování některých formulářů v šablonách využíváme knihovnu *WTForms*. V této knihovně je každý formulář definován pomocí objektu. Knihovna se pak stará o generování formuláře na HTML stránce přes Jinja2 a také pomáhá provádět validaci odeslaných dat na straně serveru (případně i na straně klienta pomocí validace formulářů v HTML5). Příjemným bonusem této knihovny je vestavěná podpora ochrany proti CSRF (Cross site request forgery).[3] Objektové definice formulářů nalezneme v modulu `psan.model`.

Při generování stránek přes Jinja2 je možné předávat do šablon proměnné z Pythonu. V některých případech bylo ale užitečnější komunikovat se serverem z JavaScriptu. Pro tyto účely se v naší aplikaci využívá AJAX. Jako formát výměny strukturovaných dat se zvolil JSON. Pro odesílání HTTP POST požadavků přes AJAX aplikace také využívá ochranu proti CSRF útokům, tentokrát se využívá `X-CSRFToken` HTTP hlavička, která obsahuje CSRF token.[4]

Příkladem místa, kde bylo třeba využívat AJAX, je anotační rozhraní. Nové načítání celé stránky po provedení akce působilo původně velmi rušivě. V tomto místě dokonce bylo nutné využívat dat uložených do mezipaměti prohlížeče — pomocí HTTP ETag hlavičky, která serveru nabídne možnost využít poslední odpověď z mezipaměti prohlížeče, místo aby server musel znovu generovat stejnou odpověď. Obě dvě tyto vylepšení výrazně snižují odezvy uživatelského prostředí při anotačním procesu.

Ukládání dat o přihlášeném uživateli probíhá pouze na straně klienta, pomocí session cookie. Session cookie přidává k ukládaným datům digitální podpis. „Podvodný uživatel“ si sice může své session cookie změnit, ale nedokáže k němu přiřadit správný digitální podpis. Pro zacházení se session cookie používáme standardní API z Flasku `flask.session`.

Pro některé činnosti potřebujeme mít možnost poskytnout uživateli dočasné oprávnění pro nějakou činnost. Pro tento účel využíváme bezstavové tokeny, který obsahují potřebné informace spolu s časem a digitálním podpisem. Pro účely generování a ověřování platnosti těchto tokenů využíváme knihovnu *itsdangerous*.

Pro práci s databází jsme si zavedli vlastní API v modulu `psan.db`, které otevírá databázi s využitím údajů z konfigurace aplikace. Toto API umožňuje znovupoužití předchozího připojení do databáze a stará se také o automatické uzavření spojení.

Každá logická část uživatelského rozhraní je vyčleněna do vlastního modulu. V každém modulu je definován Flask Blueprint. Blueprint umožňuje routovat cesty v URL za použití společného prefixu (v podobě nějaké složky). V naší aplikaci má každý modul svůj Blueprint, a proto má i každý modul svou vlastní složku v rámci URL. Uživatelské rozhraní můžeme podle modulů rozdělit na následující části:

- `psan.account` slouží pro správu účtu, změnu hesla a smazání účtu.

[3] Jedná se o typ útoku, kdy se podvodná stránka snaží provést nějakou nevyžádanou akci na jiné stránce, kde je uživatel přihlášen.

[4] Token se nastaví v JavaScriptu pomocí předání proměnné do šablony. Podvodná stránka ho nemá jak získat.

- `psan.annotate` hlavní část aplikace, která obsluhuje anotační proces. Obsahuje logiku pro vytváření anotačních oken a logiku pro zpracování anotačních rozhodnutí od uživatele.
- `psan.auth` obsahuje přihlašovací a odhlašovací logiku. Poskytuje anotaci, která ověřuje, že ke stránce přistupuje pouze uživatel s patřičným oprávněním. Dále obsahuje rozhraní pro obnovení hesla a registraci.
- `psan.generate` slouží pro generování pseudonymizovaného souboru.
- `psan.label` slouží pro správu pseudonymizačních označení, jejich úpravu, mazání, přidávání a export.
- `psan.rule` poskytuje rozhraní pro správu automatických pravidel. V rámci úpravy je možné pravidla mazat, importovat a exportovat.
- `psan.submission` umožňuje spravovat nahrané dokumenty, sledovat jejich status a stahovat vstupní data.

Webový server můžeme spustit v produkčním a vývojovém režimu. Podle zvoleného režimu pak Flask zvolí odpovídají konfiguraci z balíčku `config`. Konfiguraci aplikace podrobně popíšeme v Příloze A. Při vytváření výsledné konfigurace aplikace dodržuje následující pořadí:

- Společná konfigurace, která není závislá na režimu běhu
- Produkční nebo testovací konfigurace, přepisuje hodnoty ve společné konfiguraci.
- Instanční konfigurace, která je definovaná ve složce `instance`. Tato konfigurace přepisuje hodnoty v obecnějších konfiguracích.

### 3.5.2 Služby na pozadí

Pro spouštění delších úloh aplikace využívá frontu úloh na pozadí. Pro tento účel využíváme knihovnu Celery. Ta interně pro správu fronty využívá Redis.

Celé Celery se stará o přidávání úloh do fronty a spouštění úloh z fronty. Logicky můžeme Celery rozdělit na dvě části. V první části se Celery chová jako standardní knihovna pro Python, která se stará o plánování úloh. Druhá část, někdy označovaná *Celery worker server*, slouží pro spouštění úloh na pozadí. Tato část běží vždy ve vlastním procesu. Tato architektura umožňuje snadno spouštět náročnější úlohy na více strojích.

V naší aplikaci je pro jednoduchost Celery worker součástí hlavního Docker image. Pro správu životního cyklu obou dvou aplikací uvnitř Docker kontejneru využíváme vlastní skript, který kontroluje status obou procesů. Skript můžeme nalézt v souboru `run.sh` v kořeni projektu.

Úlohy jsou definovány v balíčku `psan.celery`. Každá úloha souvisí s jednou fází automatického zpracování dokumentu:

- `psan.celery.pre_process` označuje úlohu, která se provádí po vložení nového souboru. V rámci této úlohy dojde na tokenizaci textu a rozpoznávání pojmenovaných entit a automatickou aplikaci pravidel.

- `psan.celery.re_annotate` označuje úlohu, která provádí aplikaci automatických pravidel. Tato úloha je automaticky provedena v průběhu anotačního procesu, pokud dojde k přidání nového pravidla.

Tyto úlohy využívají pro své fungování algoritmy z automatické knihovny.

### 3.5.3 Automatická knihovna

Služby na pozadí a webové rozhraní používají pro jednotlivé operace automatickou knihovnu. Ta se nachází v balíčku `psan.tool`. Tato knihovna poskytuje rozhraní pro jazykové nástroje a implementuje fáze a operace anotačního algoritmu.

Cílem vzniku knihovny je umožnit využívání anotační algoritmu i s jinými aplikacemi, tedy nezávisle na zvoleném uživatelském rozhraní. Další motivací bylo umožnit používat automatické fáze anotačního algoritmu bez nutnosti využívání uživatelského rozhraní.

Aby byla knihovna přenositelná, tak obsahuje definice typů potřebných v anotačním algoritmu (v rámci modulu `psan.tool.model`). Pro implementaci anotačních fází pak poskytuje povinná rozhraní pro lingvistické nástroje. V modulu `psan.tool.ner` nalezneme rozhraní pro rozpoznávač pojmenovaných entit a pak i adaptér pro NameTag.

V modulu `psan.tool.parser` nalezneme pomocnou třídu pro parsování otagovaných XML souborů (podle specifikace ze sekce 3.3.2). Ta využívá pro parsování XML souboru SAX parser, který je upraven tak, aby poskytoval API generující události nastávající při zpracování XML souboru podle naší specifikace.

Událost je vyvolána, pokud narazíme na nový token nebo na novou pojmenovanou entitu. Zároveň můžeme registrovat v API žádost o zaznamenání následujícího úseku pro pozdější zpracování. Úsek je definovaný svou délkou a v rámci zaznamenávání jsou uloženy všechny lingvistické informace, které úsek obsahuje. Pokud knihovna načte celý požadovaný úsek, tak je vyvolána další událost. Knihovna samozřejmě podporuje různé vnořené a překrývající se úseky.

Této parsovací knihovny využívají automatické fáze anotačního procesu implementované v balíčku `psan.tool.task`. V tomto balíčku nalezneme modul `pre_annotate` pro spuštění tokenizace a rozpoznávání pojmenovaných entit a modul `re_annotate`, který slouží pro automatickou aplikaci pravidel (viz sekce 3.4).

Pro usnadnění práce s anotacemi a pravidly poskytuje knihovna API rozhraní v modulu `psan.tool.controller`, které pomáhá při vytváření pravidel a anotací v databázi. Úkolem tohoto modulu je sjednotit logiku využívanou v uživatelském rozhraní s logikou užívanou v automatických fázích anotačního procesu.

Důležitou součástí tohoto rozhraní je metoda `get_decisions`, která generuje výsledné anotace složené z rozhodnutí „podle pravidla“ a „podle tokenu“ spolu s pseudonymizačními označeními. Tento výstup se používá v uživatelském rozhraní při anotačním procesu nebo při generování pseudonymizovaného dokumentu.

### 3.5.4 Testy

Součástí aplikace je několik testů v balíčku `tests`. Ty ověřují funkčnost veřejných částí webu, databáze a základně testují všechny části webu dostupné po

přihlášení. Tento přístup umožňuje ověřit funkčnost aplikace a spouštěcích skriptů automaticky po všech změnách kódu.

Kromě těchto testů obsahuje projekt integrační testy pro GitHub. V nich se kromě testů spouští dále nástroje pro statickou analýzu kódu jako je Bandit a flake8. Spuštění těchto nástrojů na lokálním stroji popisuje Příloha B.

# 4. Uživatelská dokumentace

V této kapitole popíšeme aplikaci z pohledu uživatele. Popíšeme si její jednotlivé funkce a ukážeme si, jak provádět anotační proces.

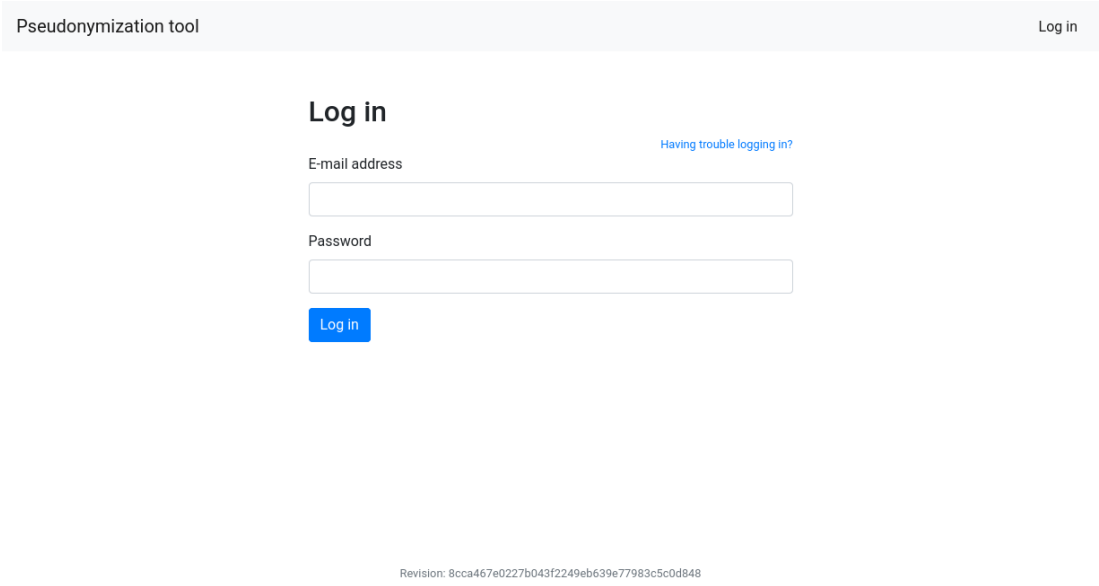
Uživatel v rámci anotačního procesu provádí rozhodnutí o vybraných úsecích textu. O každém úseku rozhodujeme, jestli je tento úsek „veřejný“ nebo „soukromý“. Veřejný úsek může zůstat ve výsledném pseudonymizovaném dokumentu. Soukromý úsek je oproti tomu předmětem procesu pseudonymizace. Tento úsek označuje místo, ve kterém se nacházejí nějaké citlivé informace.

V textu jsou zároveň zvýrazněny tzv. podezřelé úseky textu, které označují místa, která s vysokou pravděpodobností budou obsahovat nějaký citlivý údaj. Úkolem uživatele (anotátora) je o těchto místech rozhodovat.

Anotační proces se provádí v tzv. anotačním okně. Anotační okno označuje souvislý úsek textu, který je dán uživateli k anotování. Anotační okno má pak určitou velikost. Velikost okna je pro každého uživatele nastavena různě, okno vždy začíná a končí na začátku respektive konci věty.

Ve výchozím stavu je aplikace dostupná na portu 5000 místního stroje. Pro přístup můžeme použít URL `http://localhost:5000/`.[1] Celý web je responsivní, takže je aplikaci možné využívat v případě potřeby i na mobilních zařízeních. Web byl primárně optimalizován pro Firefox 89, ale bude fungovat i na prohlížečích založených na jádru WebKit.

Po otevření úvodní stránky je v horní listě zobrazeno menu. Obsah menu se po přihlášení změní, aby byl umožněn přistup k částem webu, které vyžadují přihlášení. Množství položek v menu odpovídá oprávnění daného uživatele. Na úvodní stránce máme k dispozici pouze odkaz na přihlášení (viz Obrázek 4.1). Na přihlašovací stránce má uživatel kromě přihlášení možnost také obnovit své heslo po zadání registračního e-mailu.




Obrázek 4.1: Přihlášení do aplikace

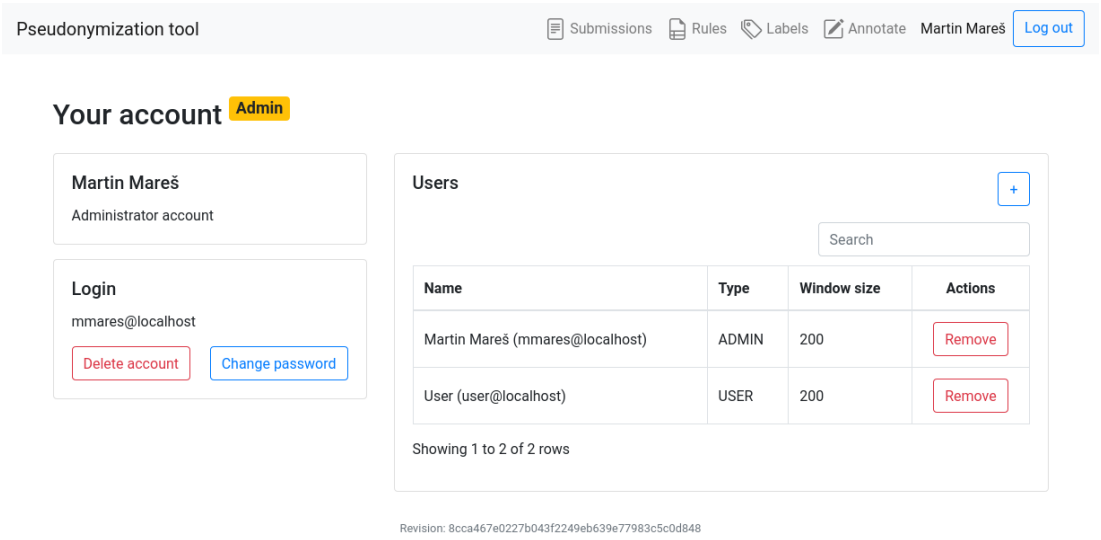
[1] Samotná adresa se může lišit podle konfigurace při nasazení. Pro produkční nasazení je doporučeno používat HTTPS.

## 4.1 Správa uživatelů

Aplikace rozděluje uživatele do dvou kategorii. Kategorie omezuje funkce aplikace, které uživatel může využít. Zároveň upravuje i chování některých částí aplikace, jako je třeba omezení velikosti anotačního okna.

- **Administrátor** má neomezený přístup ke všem funkcím aplikace, může přidávat a mazat uživatele. Může spravovat dokumenty, pravidla a pseudonymizační označení. Má právo provádět exporty a importy, a při anotovaní není omezen anotačním oknem.
- **Standardní uživatel** má oproti tomu přístup pouze k určenému anotačnímu oknu. Kromě anotování může uživatel ještě v průběhu anotačního procesu přidávat nové pseudonymizační označení (nemá ale přístup ke slovům, která se používají pro nahrazení tohoto označení ve výsledném textu).

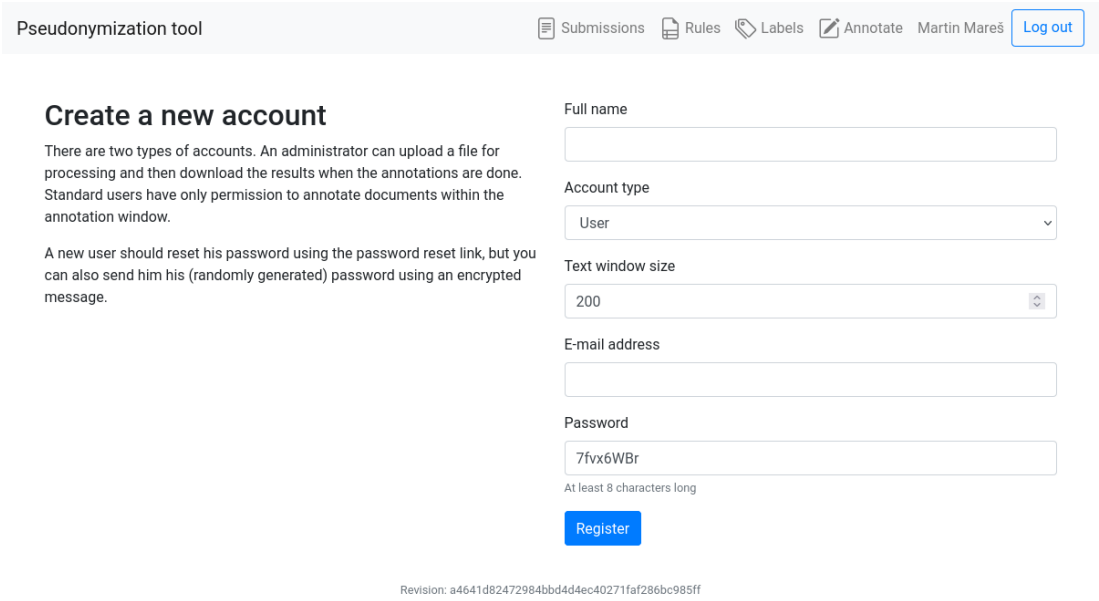
Po přihlášení máme k dispozici stránku pro správu účtu (viz Obrázek 4.2). Všichni uživatelé mají možnost změnit na této stránce své heslo nebo smazat celý svůj účet (při smazání účtu nedojde ke smazání anotací a pravidel, které uživatel vytvořil).




Obrázek 4.2: Správa účtu ve verzi pro administrátora

Administrátor má na této stránce možnost odebírat a přidávat nové uživatele. Zároveň má k dispozici přehled existujících uživatelů. Při registraci nového uživatele je třeba nastavit jméno, typ účtu, maximální velikost anotačního okna (počítáno po tokenech) a přihlašovací údaje (viz Obrázek 4.3).

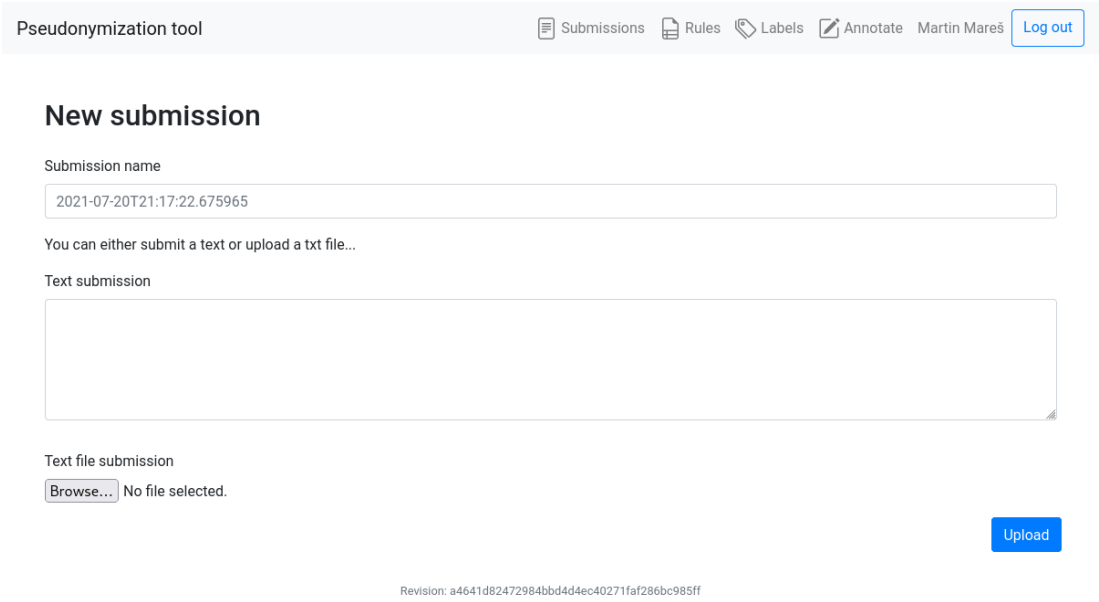
Kromě administrátora může nového uživatele vytvořit i nepřihlášený uživatel pomocí registračního tokenu. Ten je generován do logu aplikace po spuštění. Platnost tohoto tokenu je několik minut, hlavním účelem je usnadnit vytvoření prvních uživatelů do prázdné databáze. Vlastnosti tohoto tokenu (a jeho existenci) lze ovlivnit v konfiguraci.




Obrázek 4.3: Registrace nového uživatele

## 4.2 Správa dokumentů

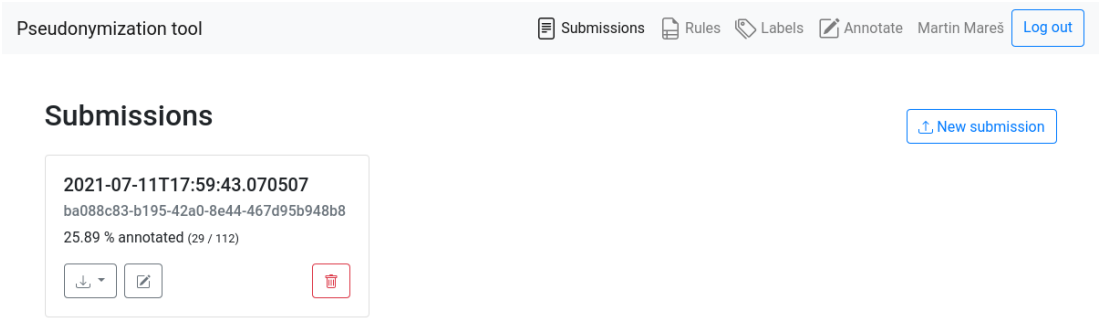
Administrátor může po přihlášení do aplikace nahrávat soubory k pseudonymizaci. Soubory je možné nahrávat z textového souboru nebo je lze vkládat přes textové pole. Název souboru se generuje automaticky z aktuálního času nebo se přebírá z názvu souboru (viz Obrázek 4.4).




Obrázek 4.4: Nahrání nového souboru

Po nahrání souboru je dokument automaticky zpracováván a jsou na něj aplikována automatická pravidla. V přehledu souborů je pak zobrazen postup při anotačním procesu, tj. kolik podezřelých slov bylo rozhodnuto (viz Obrázek 4.5).

U každého souboru je možné stáhnout vstupní data nebo soubor kdykoliv



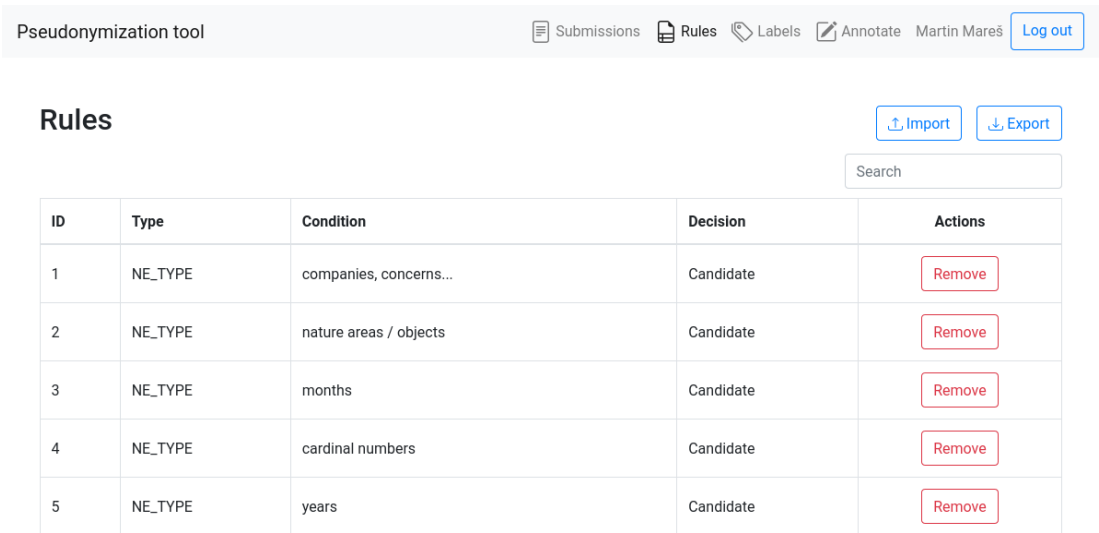

Obrázek 4.5: Správa souborů k pseudonymizaci

smazat. Při smazání souboru dojde také ke smazání všech navázaných anotací, automatická pravidla v aplikaci zůstanou.

Po dokončení automatického zpracování je dokument dostupný pro anotování v rámci náhodných anotačních oken pro všechny uživatele. Administrátor může tento dokument případně zvolit pro anotování přímo. Od dokončení automatického zpracování lze stáhnout výsledný pseudonymizovaný dokument. Pokud při stažení nejsou ještě rozhodnuty všechny podezřelé úseky, jsou tyto úseky chápány jako „veřejné“. Chybějící pseudonymizační označení jsou pak nahrazena výchozím textem `[xxx]`.

## 4.3 Správa pravidel

Administrátor může dále spravovat automatická pravidla (viz Obrázek 4.6). Automatická pravidla vytvářejí automaticky anotace pro úseky textu pomocí určených pravidel. Tato pravidla jsou vytvářena při některých rozhodnutích anotátora v průběhu anotačního procesu. Pravidla v naší aplikaci můžeme rozdělit do dvou typů. Prvním typem jsou pravidla, která vyhledávají úseky textu podle typu pojmenované entity (`NE TYPE`). Druhou možností je vyhledávat úseky textu podle typu slova, tj. podle úseků textu s určeným obsahem (`WORD TYPE`).

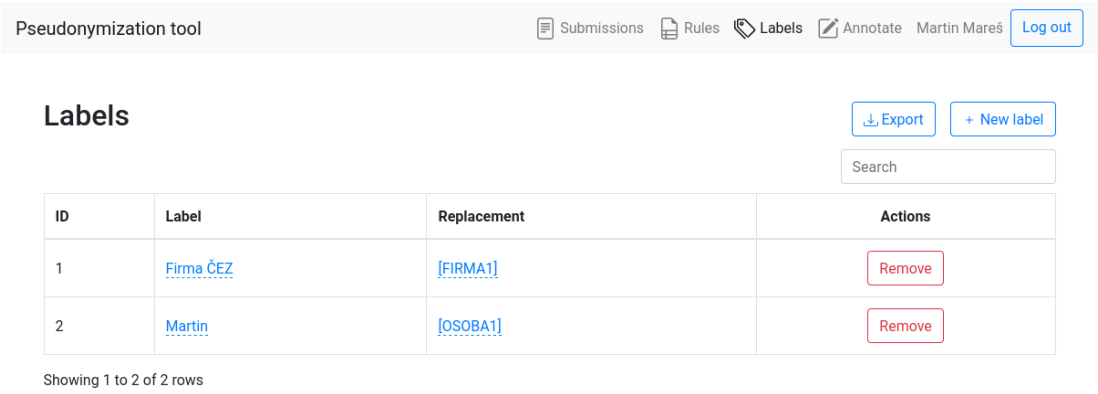



Obrázek 4.6: Správa automatických pravidel

Automatická pravidla je možné exportovat a importovat do formátu CSV. Součástí exportu je i poslední autor anotačního pravidla.

## 4.4 Správa pseudonymizačních označení

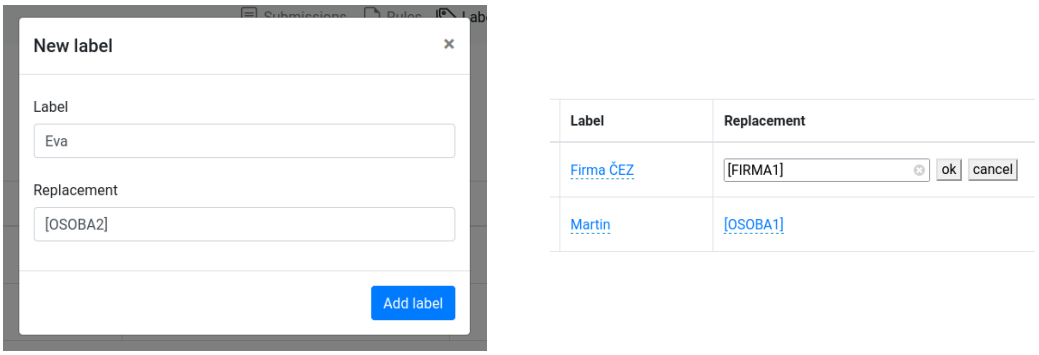
Kromě pravidel může administrátor spravovat i pseudonymizační označení (viz Obrázek 4.7). Tato označení slouží pro spojování citlivých informací, které označují jednu entitu. Jedna entita by si měla zachovat i v pseudonymizovaném textu pořád stejné označení. Bez stejného označení pak přicházíme o některé informace z textu.




Obrázek 4.7: Správa pseudonymizačního označení

Uživatelské rozhraní umožňuje dále přidávat nová označení (viz Obrázek 4.8) a exportovat označení do formátu CSV. Exportovaná označení tvoří tzv. pseudonymizační klíč.

Administrátor může dále upravovat názvy označení a měnit texty používané pro nahrazování „soukromých“ úseků při generování výsledného pseudonymizovaného dokumentu. Kromě úpravy je možné označení i smazat. Při smazání označení dojde pouze k odstranění označení u příslušných anotací. Při provádění anotačního procesu je později nutné opět přiřadit chybějící pseudonymizační označení k dotčeným anotacím.

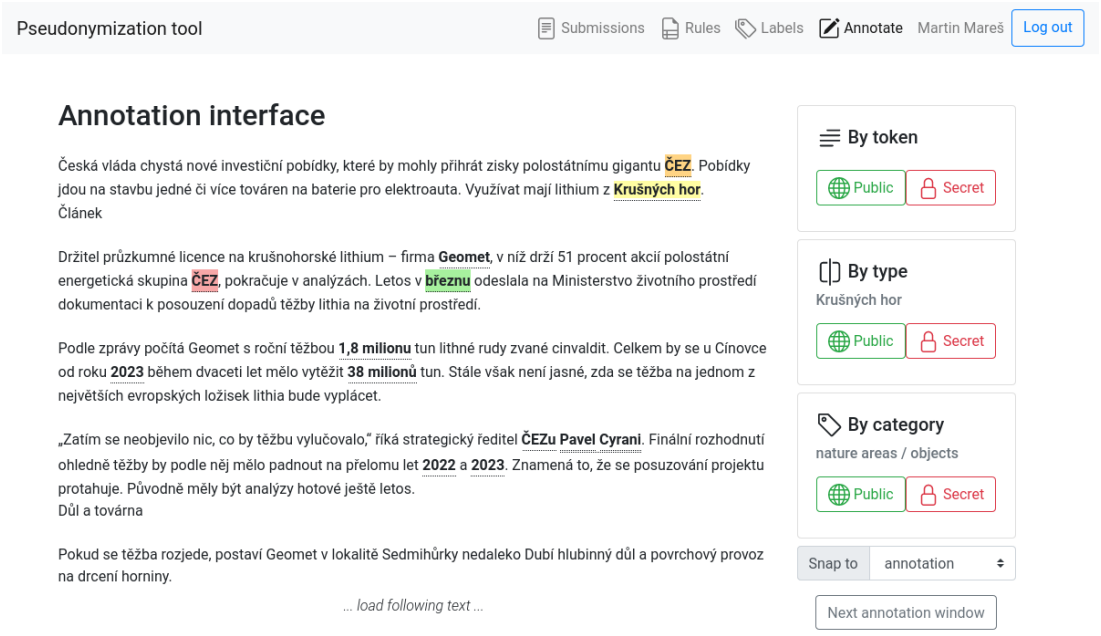



Obrázek 4.8: Přidávání a úprava pseudonymizačních názvů

## 4.5 Anotační proces

Rozhodování o úsecích textu se provádí uvnitř anotačního okna. Naše aplikace reprezentuje toto okno, jako jednu webovou stránku souvislého textu (viz Obrázek 4.9). Standardní uživatel může provádět anotace pouze v takto náhodně přidělených oknech. Dokud uživatel nerozhodne alespoň o jednom podezřelém úseku, nemá možnost přejít do jiného anotačního okna.

Administrátor není omezen anotačním oknem. V rozhraní pro správu dokumentů si může vybrat libovolný dokument, u kterého si tím vynutí anotování. Při anotování si také může anotační okno průběžně zvětšovat.




Obrázek 4.9: Anotační okno aplikace

### 4.5.1 Anotační okno

Anotační okno obsahuje prostý text, který je formátován pouze pomocí znaků nových řádku. Naše aplikace díky tomu může do textu přidávat další informace pomocí přidaného formátování.

Základní přidanou informací jsou podtržené úseky textu. Každý podtržený úsek představuje nějakou anotaci. Tato anotace mohla vzniknout buď ručně nebo pomocí nějakého automatického pravidla. V některých případech pak dochází k překrývání anotací přes sebe. V naší aplikaci jsou tyto případy znázorněny pomocí vícenásobného podtržení. Ukázkovým příkladem takové situace mohou být vnořené pojmenované entity. Například libovolné jméno a příjmení tvoří dohromady zároveň jedno pojmenování osoby. Tyto vnořené anotace vznikají pouze díky automatickým pravidlům, protože z pohledu ruční anotace s okolním kontextem nedává smysl rozhodovat o stejném textu vícekrát.

Kromě podtržení používáme v aplikaci zvýraznění úseků pomocí barev. Zvýraznění pomocí barev je vždy navázáno na nějakou anotaci, proto je nalezneme s podtržením. Podtržení nám může pomoci identifikovat spojité úseky anotace.

- **Bílou (průhlednou)** barvou označujeme „podezřelé“ úseky textu. Buď o tomto úseku nebylo rozhodnuto, nebo automatická pravidla jsou vůči sobě v konfliktu (jedno pravidlo označuje úsek jako „veřejný“ a druhé jako „soukromý“).
- **Zelenou barvou** označujeme „veřejné“ úseky textu. To jsou úseky, které mohou zůstat ve výsledném pseudonymizovaném dokumentu nezměněné.
- **Červenou barvou** označujeme „soukromé“ úseky textu. Tyto úseky nesmí být v pseudonymizovaném dokumentu. Tyto úseky zároveň mají přiřazené pseudonymizační označení.
- **Oranžovou barvou** označujeme „soukromé“ úseky bez pseudonymizačního označení — tyto úseky boudou ve výsledném pseudonymizovaném dokumentu nahrazeny pouze `[xxx]`.
- **Žlutou barvou** je zvýrazněn aktuálně vybraný úsek textu.

Cílem anotačního procesu je rozhodnout o všech „podezřelých“ úsecích (tj. označit je za „soukromé“ s pseudonymizačním označením nebo „veřejné“).

Pro vybírání úseků textu k anotování můžeme používat počítačovou myš. Ukazatel myši zvýrazňuje slovo (nebo úseky textu tvořící anotaci) žlutě. Pro označení delšího úseku vybereme nejprve začátek a poté zvolíme konec při současném držení klávesy Shift. Alternativně můžeme vybírat úseky textu pomocí klávesových zkratek, které byly inspirovány filozofií editoru Vim. Všechny zkratky jsou popsány na konci kapitoly v Tabulce 4.1.

Často potřebujeme označovat nějakou již existující anotaci, ale nastávají i případy, kdy nám existující anotace nevyhovuje zvoleným úsekem textu.

Aplikace proto nabízí dva režimy vybírání slov. V režimu po anotacích se prioritně vybírají celé anotace, oproti tomu v režimu po tokenech se vybírají vždy pouze jednotlivé tokeny.

### 4.5.2 Rozhodování

Z pohledu anotačního procesu můžeme učinit o každém vybraném úseku textu jedno ze dvou rozhodnutí. Můžeme úsek textu označit za „veřejný“, nebo „soukromý“.

Z pohledu rozhodování nedává smysl označovat úsek jako „podezřelý“, protože pokud neumíme ani s kontextem učinit rozhodnutí, tak buď musíme získat další externí zdroj informací, nebo musíme úsek označit stejně jako „soukromý“.

Jednotlivá anotační rozhodnutí mají z pohledu naší aplikace různou sílu podle toho, jak ovlivňují ostatní části textu. Protože anotátor při každém rozhodování využívá aktuálního kontextu slova, tak každé rozhodnutí vždy vytváří anotaci pro dané místo v dokumentu. Rozhodnutí, která jsou platná pro daný kontext nazýváme jako rozhodnutí *podle tokenu*.

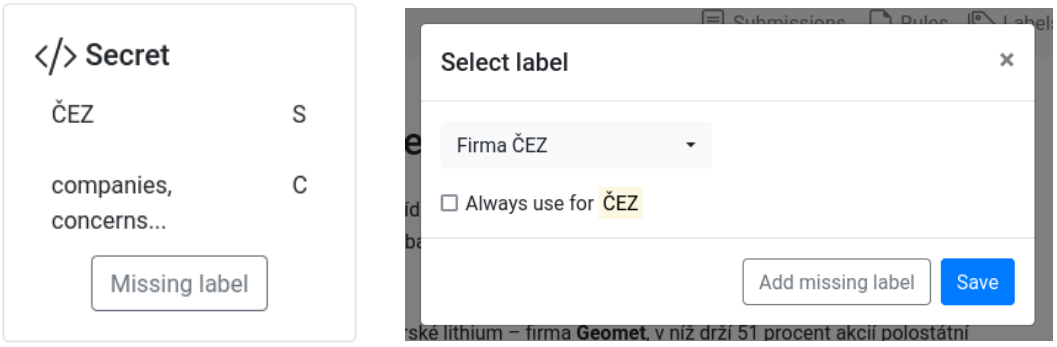
V některých situacích můžeme učinit silnější rozhodnutí, pokud víme, že všechny výskyty nějakého úseku textu, které splňují stejné kritérium, mají mít stejné rozhodnutí. Příkladem takového kritéria jsou typy slov, které bez ohledu na kontext vždy tvoří citlivou informaci. Těmto rozhodnutím budeme říkat *podle typu*.

Dalším rozhodnutí jsou rozhodnutí *podle typu pojmenované entity*, kde rozhodujeme o všech úsecích textu, kterým rozpoznávač pojmenovaných entit přiřadil stejný typ.

V aplikaci děláme rozhodnutí, která někdy ovlivňují ostatní části textu, proto i v našem anotačním okně můžeme nalézt anotace, které jsou ovlivněny více rozhodnutími z různých částí textu. Z tohoto důvodu aplikace poskytuje pro rozhodnuté anotace detail o důvodech daného rozhodnutí (viz Obrázek 4.10). V detailu rozhodnutí můžeme nalézt výsledné rozhodnutí a indicie, které k rozhodnutí vedly. Levá část tabulky popisuje důvod a pravá část tabulky popisuje rozhodnutí (`S` označuje soukromé, `P` veřejné a `C` označuje pravidlo pro vytváření podezřelých úseků).

Součástí detailu je informace o zvoleném pseudonymizačním označení (případně i možnost toto označení změnit). Pokud označíme nějaký úsek textu jako „soukromý“, automaticky dostaneme na výběr z existujících označení, abychom mohli pro daný úsek nějaké označení vybrat (viz Obrázek 4.10).

Pokud vhodné označení nenajdeme, tak můžeme nové označení přidat. Aplikace dále umožňuje svázat toto pseudonymizační označení s nějakým pravidlem pro automatické vytváření anotací. Díky tomu můžeme automaticky získávat úseky citlivých informací včetně jejich pseudonymizačních označení.




Obrázek 4.10: Detail anotace a nastavení pseudonymizačního označení

Příklad na Obrázku 4.10 říká, že řetězec „ČEZ“ uvedený v textu je vyznačen jako „soukromý“, protože tomuto úseku odpovídá pravidlo „podle typu“ „ČEZ“. Daný úsek by byl i bez tohoto pravidla „podezřelý“, a to proto, že jej rozpoznávač pojmenovaných entit poznává jako označení firmy. Pseudonymizační označení anotátor použil „Firma ČEZ“, čímž se k této interpretaci také přihlásil a tím ji vysvětlil.

Pokud označíme nějaký úsek textu jako „soukromý“, tak aplikace zároveň vytvoří pravidlo pro dohledání. To slouží pro nalezení výskytů daného označeného textu v ostatních kontextech. Výskyty v ostatních kontextech jsou poté zobrazeny v anotačním okně jako podezřelé. Pokud později změníme rozhodnutí na „veřejné“, tak dojde i k odebrání pravidla pro dohledání (a následně k odebrání takto vzniklých anotací z okolního textu).

### 4.5.3 Klávesové zkratky

Při provádění anotačního procesu podporujeme provádět většinu operací bez použití myši. Cílem této funkce je maximálně ušetřit anotátorovi čas při rozho-

dování. Klávesové zkratky jsou popsány v Tabulce 4.1.

| Klávesa | Význam |
|---|---|
| `h` | Posune interval o jeden token doleva |
| `H` | Rozšíří interval o jeden token doleva |
| `J` | Zmenší interval o jeden token zleva |
| `K` | Zmenší interval o jeden token zprava |
| `l` | Posune interval o jeden token doprava |
| `L` | Rozšíří interval o jeden token doprava |
| `e` | Otevře okno pro úpravu pseudonymizačního označení |
| `s` | Označí interval jako „soukromý“ |
| `S` | Označí typy slova z intervalu za „soukromé“ |
| `p` | Označí interval jako „veřejný“ |
| `S` | Označí typy slova z intervalu za „veřejné“ |
| `w` | Přesune anotační okno na další úsek |

Tabulka 4.1: Klávesové zkratky dostupné v anotačním procesu

### 4.5.4 Anotace od různých anotátorů

Aplikace podporuje současnou práci několika anotátorů. V rámci jedné instance aplikace jsou automatická pravidla a pseudonymizační označení sdílena mezi všemi uživateli a dokumenty. Rozhodnutí na úrovni tokenu může každý anotátor změnit, ale u každého rozhodnutí aplikace zaznamenává posledního autora. Pokud anotátor přidá pravidlo, tak je pravidlo automaticky aplikováno na celý dokument, který anotuje. Na ostatní dokumenty jsou pravidla aplikována maximálně jednou za 120 sekund, aby se snížila celková zátěž této úlohy (tuto konstantu lze v případě potřeby upravovat).

Uživatelské okno obsahuje náhodně vybraný (nedokončený) dokument, okno je vytvářeno podle nastavení uživatele na obě strany od prvního nerozhodnutého úseku. Tím má anotátor možnost kontrolovat rozhodnutí z první půlky okna, která mohou pocházet od jiného anotátora. Pokud bychom chtěli anotátory oddělit, tak by bylo třeba nechat je pracovat ve dvou různých instancích aplikace a výsledné soubory následně porovnat. Tato funkcionalita by se v případě potřeby dala do aplikace přidat.

# Závěr

V naší práci jsme navrhli a implementovali aplikaci, která usnadňuje proces pseudonymizace. Aplikace používá pro vyhledávání citlivých informací rozpoznávač pojmenovaných entit, který označuje úseky textu, ve kterých se osobní údaje mohou nacházet. O těchto úsecích aplikace dokáže rozhodovat podle automatických pravidel, a především o nich rozhoduje anotátor s využitím kontextu. Každému citlivému úseku v aplikaci přiřazujeme pseudonymizační označení. Označení může přidělit anotátor nebo ho přiřazuje aplikace pomocí automatických pravidel. Účelem pseudonymizačního označení je identifikovat stejné citlivé informace. Úseky se stejným pseudonymizačním označením poté nahrazujeme stejným nahrazujícím textem (pseudonymizačním nahrazením) při generování výsledného dokumentu. Celý pseudonymizační proces jsme navrhli v souladu s legislativními požadavky.

Z legislativního pohledu se ukázalo, že při pseudonymizaci je třeba se zaměřit na osobní údaje, které umožňují identifikaci fyzické osoby. Zároveň je nutné pseudonymizovat i citlivé osobní údaje, které popisují zdravotní stav případně genetické nebo biologické rysy fyzické osoby.

Z širšího hlediska patří mezi citlivé informace i další sdělení, jako je například nepřímá kritika nebo různé osobní poznámky. Automatické vyhledávání takových informací se ukázalo jako obtížné, proto ve finální aplikaci je podpora pouze manuálního vyhledávání takových úseků. Kromě hledání omezených úseků textu, lze stále dovodit citlivé informace i s použitím širšího kontextu. Proto jsme v naší práci popsali některé principy, které by bylo vhodné používat při vytváření pseudonymizačních nahrazení.

V práci jsme rozebrali různé možnosti, jak vyhledávat citlivé informace pomocí různých přístupů. Jako nejlepší metoda se ukázaly rozpoznávače pojmenovaných entit. Při používání rozpoznávačů jsme popsali některé jejich nedokonalosti při vyhledávání pojmenovaných entit. Podle těchto poznatků jsme aplikaci upravili. Aplikaci jsme navrhli, aby fungovala i s vnořenými pojmenovanými entitami, které se vzájemně překrývají. Principy, které tato funkce vyžadovala pro reprezentaci anotací, se poté velmi osvědčily i pro ukládání anotací od anotátora.

Z důvodu úspory času anotátora a pro podporu automatického zpracování textu jsme zavedli automatická pravidla, která podle určitých kritérií automaticky rozhodují o úsecích textu, případně i automaticky přiřazují pseudonymizační označení.

Při vytváření pseudonymizačních nahrazení jsme se setkali s vazbami mezi příbuznými slovy v původním textu (např. „Madrid“ a „Madriďan“), které by správně měly sdílet i odpovídající pseudonymizační nahrazení. Nedostatečná podpora detekce těchto vazeb v jazykových nástrojích ale znesnadňuje automatické generování pseudonymizačních nahrazení. Z tohoto důvodu v naší aplikaci podporujeme ruční generování pseudonymizačních nahrazení.

Aplikaci jsme vytvořili jako webovou službu, která běží uvnitř Docker kontejneru. Toto řešení umožňuje používat aplikaci vzdáleně. Je zároveň přívětivé z legislativního pohledu, protože ulehčuje spuštění aplikace u správce osobních údajů. Zdrojová datová kolekce tak může snáze zůstat na hardwaru správce osobních údajů. Aplikaci jsme zároveň navrhli se zvýšenými požadavky na bezpečnost.

Kvalitu kódu jsme průběžně ověřovali pomocí různých statických analyzátorů kódu.

Postupy pro vyhledávání citlivých informací jsme následně otestovali na datové kolekci z projektu ELITR. V rámci toho jsme porovnali i různé rozpoznávače pojmenovaných entit. Ukázalo se, že pro porovnání rozpoznávačů jsou problematické různé přístupy k tokenizaci. Z tohoto důvodu jsme si zavedli vlastní metriky, pomocí kterých jsme rozpoznávače porovnávali. Při vyhledávání citlivých informací pro anglické texty dopadl nejlépe rozpoznávač NameTag. Zajímavou roli hrály různé nedokonalosti datové kolekce, které jsme detekovali pomocí analýzy rozptylu výsledků jednotlivých souborů z datové kolekce. Díky těmto výsledkům se podařilo datovou kolekci i částečně pročistit.

Ve výsledcích se také ukázalo, že převážná většina správných nálezů pojmenovaných entit odpovídá přesně úsekům citlivých informací. Většina nálezů pak celý úsek alespoň obsahovala.

Přístup, ve kterém se používali pouze rozpoznávače pojmenovaných entit, negeneroval dostatečné výsledky, proto jsme navrhli vylepšení, ve kterém dohledáváme rozpoznané entity i v jiných kontextech, kde je rozpoznávač původně nenalezl. Celkově jsme s tímto vylepšením dosáhli u NameTagu detekce 85 % citlivých informací. Tento přístup se poté osvědčil i při zpracování anotací od anotátora, protože zavedl možnost dohledávání citlivých informací v dalších kontextech, kde je anotátor mohl přehlédnout.

Do budoucna můžeme do aplikace přidat automatické generování pseudonymizačních nahrazení, které by dále ušetřilo potřebný anotační čas. Vhodným postupem pro generování se nabízí využití typů pojmenovaných entit nebo využití různých seznamů slov vhodných pro nahrazení (např. seznam jmen z kalendáře). Aplikaci by bylo také vhodné v budoucnu rozšířit o další automatická pravidla, která by vyhledávala úseky textu podle editační vzdálenosti od definovaných textových typů. Tím by došlo zároveň k vylepšení detekce úseků s překlepy.

# Seznam použitých zdrojů


[1] Apple Inc. A Day in the Life of Your Data. https://www.apple.com/privacy/docs/A_Day_in_the_Life_of_Your_Data.pdf, 2021. [Online; verze 2021-04-29].

[2] Evropský parlament a Rada (EU). Nařízení Evropského parlamentu a Rady (EU) 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů a o zrušení směrnice 95/46/ES (obecné nařízení o ochraně osobních údajů). https://eur-lex.europa.eu/legal-content/cs/TXT/?uri=CELEX%3A32016R0679, 2016.

[3] Stát Kalifornie. California Consumer Privacy Act of 2018. https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5, 2018.

[4] Dwork Cynthia. Differential Privacy: A Survey of Results. *Theory and Applications of Models of Computation : 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings*, page 1, 2008.

[5] Anna Nedoluzhko and Ondřej Bojar. Towards Automatic Minuting of Meetings. In *Proceedings of the 19th Conference ITAT 2019: Slovenskočeský NLP workshop (SloNLP 2019)*, pages 112–119, Košice, Slovakia, 2019. CreateSpace Independent Publishing Platform.

[6] European Live Translator. First Shared Task on Automatic Minuting. https://elitr.github.io/automatic-minuting/index.html, 2021. [Online; verze 2021-06-18].

[7] Rekonstrukce státu. Dozvíme se, jak rozhodují soudy? Poslanci mohou prosadit zveřejňování rozsudků zákonem. https://www.rekonstrukcestatu.cz/archiv-novinek/dozvime-se-jak-rozhoduji-soudy-poslanci-mohou-prosadit-zverejnovani-rozsudku-zakonem, 2021. [Online; verze 2021-05-12].

[8] Tore Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift 15*, pages 429—-222, 1977.

[9] Philippe Golle. Revisiting the Uniqueness of Simple Demographics in the US Population. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, pages 77—80. Association for Computing Machinery, 2006.

[10] Petr Fanta, Duc Tam Hoang, Adam Huječek, Václav Pernička, Jakub Vlček. TextAn project. https://github.com/PreXident/TextAn, 2015. [Online; verze 2021-05-22].

[11] David Ifeoluwa Adelani, Ali Davody, Thomas Kleinbauer, and Dietrich Klakow. Privacy guarantees for de-identifying text transformations. In *INTERSPEECH 2020*, Shanghai, China, October 2020.

[12] Akademický senát UK. Etický kodex. https://cuni.cz/UK-9490.html, 2018. [Verze schválená Akademickým senátem UK dne 14. 12. 2018].

[13] Andy Greenberg. Apple's 'Differential Privacy' is about collecting your data — but not your data. https://www.wired.com/2016/06/apples-differential-privacy-collecting-data/, 2016. [Online; verze 2021-05-21].

[14] Jana Straková, Milan Straka, and Jan Hajič. Neural architectures for nested ner through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Stroudsburg, PA, USA, 2019. Association for Computational Linguistics.

[15] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[16] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 363–370, 2005.

[17] Jonáš Vidra, Zdeněk Žabokrtský, Lukáš Kyjánek, Magda Ševčíková, and Šárka Dohnalová. DeriNet 2.0, 2019. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[18] Jiří Kosina, SUSE. Pokročilé operační systémy: Security, Exploits. https://www.youtube.com/watch?v=Ce2paquEm70, 2019. [Přednáška na MFF UK z 16. 5. 2019, záznam online].

[19] Docker Inc. Run multiple services in a container. https://docs.docker.com/config/containers/multi-service_container/, 2021. [Online; verze 2021-05-19].

[20] Edward Loper and Steven Bird. NLTK: The Natural Language Toolkit. *CoRR*, cs.CL/0205028, 2002.

[21] Stack Exchange Inc. Stack Overflow Trends. https://insights.stackoverflow.com/trends?tags=r%2Cpython%2Cjavascript%2Cjava%2Cc%2B%2B%2Cc%23, 2021. [Online; verze 2021-06-30].

[22] The PostgreSQL Global Development Group. Appendix D. SQL Conformance. https://www.postgresql.org/docs/9.5/features.html, 2021. [Online; verze 2021-05-23].

[23] Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 617–630, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[24] Magda Ševčíková, Zdeněk Žabokrtský, and Oldřich Krůza. Named entities in czech: Annotating data and developing NE tagger. In Václav Matoušek

and Pavel Mautner, editors, *Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue*, volume 4629 of *Lecture Notes in Computer Science*, pages 188–195, Berlin / Heidelberg, 2007. Springer.

[25] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[26] Milan Straka. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[27] Robert McGill, John W Tukey, and Wayne A Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.

[28] Guido van Rossum, Barry Warsaw and Nick Coghlan. Style Guide for Python Code. `https://www.python.org/dev/peps/pep-0008/`, 2013. [Online; verze 2021-06-10].

# Přílohy

Součástí této práce je elektronická příloha zabalená ve formátu ZIP. V příloze je vložen výsledný zdrojový kód pro spuštění aplikace. Dále obsahuje skripty pro evaluaci.

- **`/src`** obsahuje zdrojový kód aplikace
- **`/src/ner_eval`** obsahuje evaluační skripty pro porovnání rozpoznávačů pojmenovaných entit z testovací datové kolekce
- **`/data`** obsahuje vygenerované souhrnné výsledky pro testovací datovou kolekci (viz popis Tabulky 2.5)

# A. Konfigurace před prvním spuštěním

V této části popíšeme minimální konfiguraci, kterou aplikace potřebuje pro svůj běh. Ve všech případech aplikace vyžaduje nastavit alespoň bezpečnostní klíče. Klíče slouží pro komunikaci s databází a vytváření digitálních podpisů a šifrování hesel. Proto by se tyto klíče nikdy neměly dostat do nepovolaných rukou (nebo např. do verzovacího systému).

Pro nastavení bezpečnostních klíčů je třeba do souboru `.env` v kořenu projektu vložit následující řádky:

```
DB_USER=user
DB_PASSWORD=tajne_heslo
APP_SECRET_KEY=tajny_klic
```

Náhodné bezpečnostní klíče můžeme například vygenerovat po zavolání následujícího příkazu z Pythonu:

```
$ python3 -c 'import os; print(os.urandom(16))'
```

Kromě konfigurace bezpečnostních klíčů aplikace ke svému fungování potřebuje jazykový model pro NameTag. Ten můžeme získat ze stránek projektu na `https://ufal.mff.cuni.cz/nametag/2/models`. Cestu k modelu je pak také třeba nastavit v `.env` souboru v kořenu projektu:

```
NER_MODEL=./instance/czech-cnec2.0-140304.ner
```

V případě produkčního prostředí potřebujeme ještě také nastavit adresy pro odesílání e-mailů z aplikace. Do instanční konfigurace je třeba vložit řádky z následujícího boxu. Instanční konfiguraci nalezneme ve výchozím stavu v souboru `./instance/config.py`.

```
TOKEN_FROM_EMAIL = "Automat <automat@localhost>" # Odesílatel
TOKEN_SMTP_HOST = "localhost" # SMTP server
```

Pro produkční nasazení je doporučeno zapnout podporu HTTPS. Pro zapnutí HTTPS je třeba získat důvěryhodný certifikát. Pro získání důvěryhodného certifikátu lze využít například službu jako *Let's encrypt*.

Po získání certifikátu je třeba upravit konfiguraci v `uwsgi.ini` v kořenu projektu. Přesná konfigurace záleží na používaném certifikátu, obecné postupy jsou popsány v oficiální dokumentaci na `https://uwsgi-docs.readthedocs.io/en/latest/HTTPS.html`.

Alternativně můžeme HTTPS zprovoznit použitím reverse proxy ve spojitosti s uWsgi. V tomto případě můžeme použít například NGINX.

# B. Instrukce pro spuštění aplikace

V této části popíšeme minimální požadavky pro spuštění aplikace, následně také ukážeme, jak aplikaci spouštět v různých režimech.

Pro spuštění aplikace v libovolném režimu konfigurace musí být aplikace nejprve nastavena pro danou instanci (viz Příloha A).

Pro spouštění různých konfigurací se využívá nástroj Make. Konfigurace pro Make byla vytvořena pro operační systém Linux, ale bude fungovat i na Windows při použití Windows Subsystem for Linux 2 (WSL 2). Pro spouštění kontejneru se používá Docker (v doporučené minimální verzi 20.10) a Docker Compose ve verzi alespoň 1.28.

Pro plné vývojové prostředí potřebujeme Python ve verzi 3.7. Aplikace sama o sobě podporuje i novější verze Pythonu, ale knihovna pro rozpoznávač pojmenovaných entit NameTag novější verze Pythonu bohužel zatím nepodporuje (viz `https://github.com/ufal/nametag/issues/12`).[1]

Zdrojový kód aplikace můžeme získat z elektronické přílohy práce nebo z verzovacího systému na adrese `https://github.com/ELITR/pseudonymizer`.

## B.1 Vytváření Docker image

Pro spuštění aplikace v Dockeru musíme nejprve vytvořit Docker image. To provedeme pomocí následujícího příkazu v kořenu projektu (doporučovaná varianta):

```
make docker-build
```

Alternativně v případě problémů s verzí Pythonu můžeme image vytvořit přímo pomocí Docker Compose:

```
docker-compose build
```

## B.2 Práce s aplikací

Pomocí nástroje Make můžeme aplikaci spouštět v různých konfiguracích a spouštět také různé testy. Make se sám stará o udržování závislostí na knihovnách a vytváření virtuálního prostředí pro Python (venv).

- **`make docker-debug`** spustí aplikaci ve vývojové konfiguraci uvnitř Docker kontejneru (s aktuálním kódem).
- **`make docker`** spustí aplikaci v produkční konfiguraci uvnitř Docker kontejneru (s kódem z posledního sestavení Docker image).

[1] V současné době můžeme toto omezení do určité míry obejít vývojem přímo v Docker kontejneru.

- **`make docker-test`** spustí unit testy pro ověření základní funkčnosti uvnitř Docker kontejneru (s aktuálním kódem).
- **`make lint`** spustí statickou kontrolu správného formátování kódu pomocí nástroje flake8.
- **`make bandit`** spustí statickou kontrolu bezpečnostních chyb pomocí nástroje Bandit.
- **`make clean`** pročistí projekt od dočasných souborů a odstraní vytvořené Docker image.

## B.3 Spuštění pro produkci

V čistě produkčním prostředí můžeme aplikaci spouštět po provedení konfigurace (viz Příloha A) pouze zavoláním následujícího příkazu v kořenu projektu:

```
docker-compose up
```

Tímto postupem se vytvoří produkční Docker image spolu se všemi závilostmi. Aplikace se poté spustí v produkčním webovém serveru uWsgi.