



# NÁSTROJ PRO DETEKCI LICENCÍ A PODMÍNEK POUŽITÍ NA WEBU

**Tomáš Peterka**

Bakalářská práce  
Fakulta informačních technologií  
České vysoké učení technické v Praze  
Katedra softwarového inženýrství  
Studijní program: Informatika  
Specializace: Webové inženýrství  
Vedoucí: Ing. Jaroslav Kuchař, Ph.D.  
16. května 2025

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Peterka** Jméno: **Tomáš** Osobní číslo: **515744**  
Fakulta/ústav: **Fakulta informačních technologií**  
Zadávající katedra/ústav: **Katedra softwarového inženýrství**  
Studijní program: **Informatika**  
Specializace: **Webové inženýrství**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Nástroj pro detekci licencí a podmínek použití na webu**

Název bakalářské práce anglicky:

**Tool for detection of licenses and terms of use on web**

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**Ing. Jaroslav Kuchař, Ph.D. katedra softwarového inženýrství FIT**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **30.01.2025**

Termín odevzdání bakalářské práce: **16.05.2025**

podpis vedoucí(ho) ústavu/katedry

podpis děkana(ky)

## III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

\_\_\_\_\_

Datum převzetí zadání

Peterka Tomáš  
\_\_\_\_\_

Podpis studenta

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Peterka** Jméno: **Tomáš** Osobní číslo: **515744**  
Fakulta/ústav: **Fakulta informačních technologií**  
Zadávající katedra/ústav: **Katedra softwarového inženýrství**  
Studijní program: **Informatika**  
Specializace: **Webové inženýrství**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Nástroj pro detekci licencí a podmínek použití na webu**

Název bakalářské práce anglicky:

**Tool for detection of licenses and terms of use on web**

Pokyny pro vypracování:

Na webu existuje mnoho zdrojů, které lze využít pro účely dalšího zpracování. Některé jsou ale vlastníky omezovány definováním podmínek nebo uvedením licencí. Cílem práce je navrhnout a implementovat nástroj pro detekci a zpracování licenčních podmínek a podmínek použití na webových stránkách.

- Seznamte se s problematikou extrakce informací z webových zdrojů, zahrnující techniky zpracování přirozeného jazyka (NLP).
- Prozkoumejte existující zdroje dat a nástroje, které mohou pomoci s identifikací a analýzou licenčních podmínek a podmínek použití.
- Navrhnete, implementujete a otestujete řešení extrakce a reprezentace relevantních informací z webových stránek.
- Vyvinutý nástroj by měl být schopen detekovat uvedenou licenci, zpracovat a vyhodnotit podmínky použití.
- Využijte jazykové modely pro vyhodnocení podmínek použití.
- Vytvořte prototyp webové aplikace využívající implementované řešení pro prezentaci a vizualizaci dílčích výstupů.

Seznam doporučené literatury:

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2025 Tomáš Peterka. Všechna práva vyhrazena.

*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení, je nezbytný souhlas autora.*

Odkaz na tuto práci: Peterka Tomáš. *Nástroj pro detekci licencí a podmínek použití na webu*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2025.

*Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Jaroslavu Kuchařovi, Ph.D., za jeho odborné vedení, cenné rady, trpělivost a podporu během celé doby zpracování této práce. Velké poděkování patří také mé rodině za jejich trvalou podporu, povzbuzení a pochopení, které mi poskytovali po celou dobu studia i při realizaci této práce.*

## Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

Prohlašuji, že jsem v průběhu příprav a psaní závěrečné práce použil nástroje umělé inteligence. Vygenerovaný obsah jsem ověřil. Stvrzuji, že jsem si vědom, že za obsah závěrečné práce plně odpovídám.

V Praze dne 16. května 2025

## Abstrakt

Tato bakalářská práce se zaměřuje na automatizovanou detekci a zpracování licencí a podmínek použití na webových stránkách. Výsledkem je nástroj, který kombinuje web scraping, zpracování přirozeného jazyka a moderní jazykové modely pro identifikaci a analýzu těchto dat. Nástroj je vytvořen jako Python knihovna, jejíž funkce jsou prezentovány na prototypu webové aplikace. Schopnost nástroje detekovat jednotlivé licence je testována jak na reálných datech tak na datech vygenerovaných. Rovněž jsou v práci testovány výstupy jazykových modelů a to metrikami ROGUE, BERTScore a Faithfulness.

**Klíčová slova** licence, podmínky použití, web scraping, NLP, jazykové modely, knihovna, webová aplikace, Python

## Abstract

This bachelor's thesis focuses on the automated detection and processing of licenses and terms of service on websites. The outcome is a tool that combines web scraping, natural language processing, and modern language models to identify and analyze such data. The tool is implemented as a Python library, with its functionalities demonstrated on a prototype web application. The tool's ability to detect individual licenses is tested on both real and synthetic data. Furthermore, the outputs of language models are evaluated using the ROUGE, BERTScore, and Faithfulness metrics.

**Keywords** licenses, terms of service, web scraping, NLP, language models, library, web application, Python

## Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Analýza</b>	<b>2</b>
1.1 Licence na webu . . . . .	2
1.1.1 Creative Commons . . . . .	3
1.2 Podmínky použití na webu . . . . .	4
1.3 Web scraping . . . . .	5
1.4 NLP . . . . .	6
1.4.1 Jazykové modely . . . . .	7
1.5 Existující nástroje a řešení . . . . .	8
1.6 Shrnutí . . . . .	8
<b>2 Návrh</b>	<b>9</b>
2.1 Požadavky na nástroj . . . . .	9
2.1.1 Funkční požadavky . . . . .	9
2.1.2 Nefunkční požadavky . . . . .	10
2.2 Architektura nástroje . . . . .	10
2.2.1 Knihovna . . . . .	10
2.2.1.1 Tok dat . . . . .	12
2.2.2 Prototyp webové aplikace . . . . .	15
2.2.2.1 Backend . . . . .	15
2.2.2.2 API . . . . .	16
2.2.2.3 Frontend . . . . .	17
<b>3 Implementace</b>	<b>19</b>
3.1 Výběr technologií . . . . .	19
3.2 Moduly . . . . .	20
3.2.1 Správa požadavků . . . . .	20
3.2.2 Extrakce dat . . . . .	21
3.2.3 Identifikace licence . . . . .	21
3.2.4 Zpracování ToS . . . . .	22
3.3 Formát výsledků . . . . .	23
3.4 Definice parametrů . . . . .	25
3.5 Prototyp webové aplikace . . . . .	27
3.5.1 Backend . . . . .	27
3.5.2 API . . . . .	29



3.5.3	Frontend . . . . .	31
<b>4</b>	<b>Testování</b>	<b>34</b>
4.1	Testování detekce . . . . .	34
4.2	Hodnocení modelů . . . . .	36
4.2.1	Výběr modelů . . . . .	36
4.2.2	ROUGE . . . . .	37
4.2.3	BERTScore . . . . .	37
4.2.4	Faithfulness . . . . .	37
4.2.5	Výsledky . . . . .	37
	<b>Závěr</b>	<b>39</b>
	<b>Obsah příloh</b>	<b>45</b>

## Seznam obrázků

2.1	Architektura nástroje . . . . .	11
2.2	Vývojový diagram 1. části nástroje . . . . .	13
2.3	Vývojový diagram 2. části nástroje . . . . .	14
2.4	Úvodní rozhraní . . . . .	17
2.5	Výsledkové rozhraní . . . . .	18
2.6	Rozhraní souhrnu . . . . .	18
3.1	Celery diagram . . . . .	30
3.2	Odeslání URL adres a API klíčů . . . . .	32
3.3	Zobrazení výsledků analýzy a možnost sumarizovat ToS . . . . .	32
3.4	Zobrazení shrnutí ToS a možnost položit otázku . . . . .	33
3.5	Zobrazení odpovědi na otázku a možnost položit další otázku . . . . .	33

## Seznam tabulek

2.1	Přehled endpointů navrženého API . . . . .	16
4.1	Přehled výsledků testování detekce licencí . . . . .	35
4.2	Přehled výsledků metrik hodnocení . . . . .	38

## Seznam výpisů kódu

3.1	Metoda <i>prepare_prompt</i> třídy BaseModel . . . . .	23
3.2	JSON formát výsledků . . . . .	24
3.3	Klíčová slova pro odkazy a textový obsah ToS . . . . .	25
3.4	Parametry licencí Creative Commons . . . . .	25
3.5	Regulární výraz pro Creative Commons . . . . .	25

3.6	Regulární výraz pro MIT License . . . . .	26
3.7	Regulární výraz pro Apache License . . . . .	26
3.8	Regulární výraz pro GNU GPL . . . . .	26
3.9	Regulární výraz pro BSD License . . . . .	27

## Seznam zkratek

AGPL	Affero General Public License
AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BSD	Berkeley Software Distribution (License)
CC	Creative Commons
CSS	Cascading Style Sheets
DOM	Document Object Model
GPL	General Public License
GPT	Generative Pre-trained Transformer
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
LGPL	Lesser General Public License
LLM	Large Language Model
MIT	Massachusetts Institute of Technology (License)
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
POS	Part of Speech
REST	Representational State Transfer
ROGUE	Recall-Oriented Understudy for Gisting Evaluation
SPDX	Software Package Data Exchange
ToS	Terms of Service
UA	User Agent
URL	Uniform Resource Locator

# Úvod

V dnešní době digitálních technologií je dostupnost webového obsahu obrovská a neustále roste. S tím však přibývá i množství právních dokumentů, jako jsou licence a podmínky použití, které uživatelé často přehlížejí. Přesto jsou tyto dokumenty klíčové, protože vymezují, jak mohou uživatelé obsah sdílet, upravovat a dále využívat. Neznalost těchto podmínek může vést k neúmyslnému porušení autorských práv, což v některých případech může mít i právní důsledky.

Hlavním problémem je, že právní dokumenty jsou často dlouhé, složité formulované a pro běžné uživatele obtížně srozumitelné. To vede k tomu, že je uživatelé vnímají spíše jako obtíž a ignorují je, aby dosáhli svých cílů, aniž by byli omezováni [1].

Cílem této práce je analyzovat metody extrakce informací z webových zdrojů a navrhnout a implementovat nástroj, který pomocí těchto metod dokáže detekovat licence a zpracovat podmínky použití na webových stránkách. Tento nástroj tak uživatelům poskytne stručné a relevantní informace o licenčních omezeních, čímž usnadní orientaci v textech o podmínkách použití obsahu.

Práce je rozdělena do čtyř hlavních kapitol, které postupně pokrývají analýzu, návrh, implementaci a testování nástroje. V první kapitole jsou analyzovány licence a podmínky použití na webu, web scraping, jazykové modely, zpracování přirozeného jazyka a existující řešení. Druhá kapitola se věnuje návrhu nástroje, včetně požadavků, architektury a prototypu webové aplikace. Třetí část se zaměřuje na implementaci, zahrnující výběr technologií, extrakci a zpracování dat, využití jazykových modelů a vývoj prototypu. A nakonec čtvrtá kapitola, která zahrnuje testování nástroje a hodnocení přesnosti jazykových modelů.

Téma této práce bylo vybráno z důvodu její vysoké aktuálnosti spojené s rozmachem využívání jazykových modelů. Navíc současné principy zpracování přirozeného jazyka a rostoucí význam jazykových modelů nacházejí široké uplatnění i v řadě dalších oblastech.

# Kapitola 1

## Analýza

*Před zahájením návrhu a implementace nástroje je nezbytné provést analýzu klíčových aspektů problému. Tato kapitola se věnuje rozboru licencí a podmínek použití na webových stránkách, technikám web scrapingu pro extrakci informací z webových zdrojů, principům fungování jazykových modelů, zpracování přirozeného jazyka a přehledu existujících nástrojů a řešení. Cílem je vytvořit pevný základ pro další fáze vývoje nástroje.*

### 1.1 Licence na webu

Licence na webu představují rámec, který definuje, jak mohou uživatelé nakládat s obsahem dostupným na internetu. Tyto licence jsou klíčovým prvkem pro ochranu duševního vlastnictví a zároveň umožňují stanovit jasná pravidla pro další využití obsahu. Existuje více než 400 různých licencí, které se liší podle toho, jaké podmínky stanovují. Každá taková licence může být využívána pro různé druhy obsahu, jako jsou texty, obrázky, videa nebo software. Každá licence má navíc své vlastní zkratky či symboly, které ji identifikují. [2]

Licence na webu lze rozdělit do několika základních kategorií podle toho, jaké podmínky stanovují pro využívání obsahu:

**Proprietární licence:** Tyto licence jsou nejrestriktivnější a znemožňují kopírování, úpravy nebo distribuci obsahu. Chrání tak vlastníka před neoprávněným použitím.

**Public domain:** Obsah označený jako public domain je volně dostupný k použití a úpravám bez jakýchkoli omezení.

**Unlicensed:** Pokud není explicitně uvedena licence, obsah podléhá autorskému právu a jeho použití může být omezeno, i když podmínky nejsou jasně specifikovány.

**Open source licence:** Umožňují používání, úpravy a sdílení obsahu za určitých podmínek. Dělí se na následující kategorie:

**Permisivní licence:** (např. Apache, MIT, BSD) Jedná se o licence, které zajišťují velmi volné použití softwaru. Umožňují jeho začlenění do komerčních i soukromých projektů, přičemž se liší například v požadavcích na zachování licenčních upozornění nebo autorských práv.

**Copyleft licence:** (např. GPL, AGPL) Tyto restriktivní licence vyžadují, aby upravený nebo distribuovaný software byl šířen pod stejnou licencí.

**GNU/LGPL:** Umožňuje vývojářům propojit svůj software s otevřenými knihovnami a licencovat výsledný kód podle vlastního uvážení, včetně proprietárních licencí. Podmínkou však je, že samotná knihovna musí zůstat pod licencí LGPL [3]

Zatímco většina open source licencí nachází uplatnění především v oblasti softwaru, pro obsah na webových stránkách, jako jsou texty, obrázky či multimedia, existují otevřené licence typu Creative Commons. Ty se staly standardem mezi otevřenými licencemi díky své flexibilitě a přizpůsobivosti různým typům tvůrčího obsahu. V současnosti je evidována více než jedna miliarda využití [4], což svědčí o jejich širokém přijetí.

### 1.1.1 Creative Commons

Creative Commons (CC) je mezinárodní nezisková organizace, která podporuje rozšiřování a udržování společného prostoru sdílených znalostí a kultury. Creative Commons licence poskytují standardizovaný způsob, jak mohou jednotliví tvůrci i velké korporace povolit veřejnosti využívat jejich autorská díla v souladu s autorským právem. Z pohledu uživatele, který dílo znovu používá, existence Creative Commons licence na chráněném díle objasňuje, co s tímto dílem může dělat. [5]

Existuje šest základních typů CC licencí, které se liší v podmínkách použití:

**CC BY:** Nejotevřenější licence, která umožňuje redistribuci, vytváření odvozenin a dokonce komerční využití díla za podmínky, že je uveden původní autor a jsou označeny případné změny.

**CC BY-SA:** Podobná jako CC BY, ale vyžaduje, aby odvozená díla byla sdílena pod stejnou licencí.

**CC BY-NC:** Povoluje sdílení a úpravy díla, ale pouze pro nekomerční účely.

**CC BY-ND:** Umožňuje redistribuci díla, ale zakazuje jeho úpravy.

**CC BY-NC-SA:** Kombinuje podmínky nekomerčního využití a sdílení pod stejnou licencí.

**CC BY-NC-ND:** Nejrestriktivnější CC licence, která povoluje pouze sdílení původního díla pro nekomerční účely bez možnosti úprav. [5]

Speciálním případem je licence CC0, která umožňuje autorům uvolnit své dílo do veřejného vlastnictví a zříci se všech autorských práv. Tato licence tak spadá do kategorie „Public domain“ a je určena pro autory, kteří chtějí, aby jejich dílo bylo volně dostupné bez jakýchkoli podmínek.

Reprezentace takových licencí by pak měla být jednoznačná a snadno identifikovatelná, aby uživatelé mohli rychle zjistit, jak mohou daný obsah využívat. Doporučeným způsobem, jak licenci zobrazit, je uvést stručný text s informací o licenci a odkazem na vybranou CC licenci, například: © 2019. *This work is openly licensed via CC BY 4.0.* Pro pokročilejší zobrazení lze využít HTML kód přímo od CC. [5]

Umístění označení licence na webových stránkách není nijak standardizované a může se lišit na základě konkrétního obsahu. Pokud se jedná o konkrétní média jako obrázky nebo videa, bývá licence často zobrazena přímo u obsahu. Ale pokud se jedná o celou webovou stránku, bývá licence zobrazena v patičce stránky nebo je zmíněna v rámci textu podmínek použití. Tato pozorování byla získána během analýzy několika desítek webových stránek, obsahujících různé typy CC licencí.

## 1.2 Podmínky použití na webu

Dalším, již dříve zmíněným, způsobem jak omezit využití obsahu na webových stránkách, jsou podmínky použití. Podmínky použití (ToS) stanovují pravidla, která musí jednotlivci nebo organizace dodržovat, aby mohli platformu využívat. Tyto podmínky jsou obvykle právně závazné, pokud neporušují místní zákony. Poskytovatel může tyto podmínky čas od času upravit a je jeho povinností informovat uživatele o jakýchkoli změnách. [6]

Na rozdíl od licencí, které se primárně zaměřují na autorská práva a způsoby využití obsahu, ToS pokrývají širší spektrum aspektů souvisejících s používáním webových stránek. Zavedení ToS přináší provozovatelům webů řadu důležitých výhod. Vymezují pravidla, která musí uživatelé dodržovat při využívání služeb, čímž pomáhají omezit zneužívání webu. Zároveň slouží k omezení právní odpovědnosti provozovatele a snižují tak potenciální rizika. Navíc mohou objasnit, komu patří obsah na webu, a stanovují tak vlastnická práva.

Reprezentace těchto podmínek není striktně standardizovaná a může se na různých webových stránkách lišit. Avšak je v zájmu autorů, aby byly ToS snadno rozpoznatelné, což vytváří často používané textové vzory jak podmínky zobrazit.



Analyzováno bylo více než sto webových stránek (v anglickém jazyce). Tento vzorek je tvořen jednak stránkami, které byly nalezeny při rešerši k tématu práce, a jednak výběrem z nejnavštěvovanějších webů dle [7]. Na základě této analýzy byly identifikovány následující časté textové vzory odkazující na ToS nebo jejich části:

- *Terms of Service*
- *Terms and Conditions*
- *Terms of Use*
- *License*
- *Legal*

Aby uživatelé mohli snadno zjistit, jak mohou obsah využívat, je důležité, aby byly ToS snadno dostupné. Podobně jako u licencí, nelze určit jejich jednoznačné umístění na webových stránkách. Nicméně při analýze stránek bylo zjištěno, že patička webových stránek je nejčastějším místem, kde jsou ToS umístěny.

Patička totiž obvykle zahrnuje informace týkající se své sekce, jako jsou odkazy na související dokumenty, údaje o autorských právech či autora patičky. Pokud patička obsahuje rozsáhlé části obsahu, tyto části obvykle představují přílohy, rejstříky, podrobné licenční smlouvy nebo jiný podobný obsah [8]. Z tohoto důvodu není překvapující, že vzory textů ToS i označení licencí byly v rámci analýzy nejčastěji identifikovány právě v patičce.

### 1.3 Web scraping

Aby bylo možné získat informace o licencích a podmínkách použití na webových stránkách, je nutné provést extrakci dat. Aby tato extrakce nemusela být prováděna ručně, což by bylo velmi pracné a náročné, využívá se automatizovaná technika známá jako web scraping.

Web scraping je proces, který získává nestrukturovaná data, jako je HTML obsah webových stránek, a převádí je do strukturovaných formátů, což usnadňuje jejich analýzu či ukládání. Tato technika je zvláště důležitá v situacích, kdy by ruční sběr dat byl nepraktický kvůli rozsahu nebo četnosti aktualizací, například při sledování tržních trendů v reálném čase nebo shromažďování výzkumných datových sad. Tím, že umožňuje přístup k jinak pracně získatelným informacím, web scraping slouží jako výkonný nástroj napříč průmyslovými odvětvími i akademickou sférou. [9]

Metody a nástroje používané při web scrapingu se přizpůsobují různým technickým potřebám a složitostem webových stránek. Mezi využívané techniky patří přístupy, jako je porovnávání textových vzorů pomocí regulárních

výrazů, ale i pokročilejší metody, například parsování DOM, které cílí na konkrétní prvky webové stránky. Používají se také automatizační nástroje simulující lidské prohlížení, které umožňují zpracování dynamického obsahu generovaného JavaScriptem. [9] Tyto možnosti tak umožňují přizpůsobit web scraping dle potřeb použití, ať už jde o jednoduchou extrakci dat, nebo složité webové struktury.

Navzdory své užitečnosti čelí web scraping mnoha výzvám, které jeho použití komplikují. Právní a etické otázky představují hlavní překážky, protože web scraping může být v rozporu s podmínkami používání webových stránek nebo zákony o duševním vlastnictví, což vytváří nejasné prostředí pro jeho využití. Kromě toho technické bariéry, jako jsou CAPTCHA, blokování IP adres nebo neustále se měnící rozložení stránek, vyžadují neustálé inovace, aby scraping zůstal efektivní. Kvalita získaných dat také představuje problém, protože nekonzistence nebo neúplné informace často vyžadují další zpracování. [9]

Při web scrapingu je tak důležité dodržovat etické standardy. Mnoho webových stránek zakazuje procházení svého obsahu vyhledávacími roboty (využívající web scraping) pomocí souboru *robots.txt*. Tento soubor je obvykle umístěn v kořenovém adresáři webové stránky (například <https://www.example.com/robots.txt>) a určuje, zda mohou vyhledávací roboti přistupovat k celému webu nebo jen k vybraným zdrojům [10]. Ačkoli tento soubor nezastaví nikoho, kdo chce web scraping provádět, je etickým standardem jej respektovat.

## 1.4 NLP

Zpracování přirozeného jazyka (NLP) je disciplína na pomezí umělé inteligence a počítačové lingvistiky, jejímž cílem je umožnit počítačům porozumět, interpretovat a generovat lidský jazyk. NLP lze rozdělit do dvou hlavních oblastí:

1. Generování přirozeného jazyka (NLG)
2. Porozumění přirozenému jazyku (NLU)

Mezi procesy spojené s chápáním a tvorbou textového obsahu, které tyto kategorie zahrnují, patří mimo jiné:

- **Rozpoznávání pojmenovaných entit**  
Identifikace a klasifikace entit v textu.  
*Příklady entit:* osoby, organizace, místa, data.
- **Strojový překlad**  
Automatický převod textu z jednoho jazyka do druhého.
- **Odpovídání na otázky**  
Automatické vyhledávání a poskytování odpovědí na otázky položené v přirozeném jazyce.

- **Analýza sentimentu**

Určení emocí nebo postoje vyjádřeného v textu a jeho klasifikace jako: pozitivní, negativní, neutrální.

- **Generování textu**

Tvorba textu podobného lidskému na základě zadaných podmínek nebo podnětů.

- **Označování slovních druhů (POS tagging)**

Přiřazení každému slovu ve větě konkrétního slovního druhu. [11]

NLP se neustále vyvíjí díky pokrokům v hlubokém učení, dostupnosti rozsáhlých datových sad a rostoucí poptávce po inteligentních systémech pro zpracování jazyka. Klíčovou roli v tomto vývoji hrají moderní jazykové modely, které významně posouvají možnosti NLP.

### 1.4.1 Jazykové modely

Jazykové modely existují již od 80. let 20. století a slouží ke statistickému modelování vlastností přirozeného jazyka na základě analýzy textů, například frekvencí slov, jejich pravděpodobností a kontextu. Díky nedávnému nárůstu dostupnosti textových sbírek a zlepšení výpočetních zdrojů byly představeny velké jazykové modely (LLM), které přinesly revoluci do pole NLP. LLM vycházejí ze stejných principů jako tradiční jazykové modely, ale díky obrovským textovým databázím, často zahrnujícím téměř celý web, je možné trénovat mnohem větší a výkonnější modely. [12]

V rámci NLP přinesly zásadní změnu zejména modely hlubokého učení, konkrétně architektury typu transformer, jako jsou BERT (Bidirectional Encoder Representations from Transformers) nebo GPT (Generative Pre-trained Transformer). Tyto modely výrazně posílily schopnosti systémů porozumět a generovat jazyk podobný lidskému. Díky nim se zlepšily výsledky v úlohách jako strojový překlad, vývoj chatbotů, analýza sentimentu a dalších jazykových aplikací. Transformátorové modely využívají trénink na rozsáhlých datových sadách, což jim umožňuje zachytit nuance jazyka a efektivně zpracovávat složité textové struktury. [13]

V dnešní době existuje celá řada modelů a to jak open source, tak proprietárních. Navíc čím dál častěji vznikají nové modely, které dosahují stále lepších výsledků. To však ztěžuje jednoznačné určení dlouhodobě nejlepšího modelu pro konkrétní účel. Dne 6. 4. 2025 je na prvních příčkách model *Gemini-2.5-Pro-Exp*. [14, 15]

K detekci licencí lze tak využít základní metody NLP, jako například rozpoznávání pojmenovaných entit, které dokáže identifikovat klíčové informace v textu a přiřadit jim konkrétní kategorie. Avšak pro zpracování složitějších textů, jako jsou ToS, se jako vhodnější jeví moderní jazykové modely, které

díky svému tréninku na rozsáhlých datových sadách a schopnosti zachytit kontext a nuance jazyka, umožňují hlubší analýzu textu.

## 1.5 Existující nástroje a řešení

V oblasti detekce a analýzy licenčních podmínek existuje několik nástrojů a řešení. Mezi ně například patří FOSSA [16], FOSSID [17], Black Duck [18] či FOSSology [19]. Avšak všechny tyto nástroje se zaměřují především na detekci open source licencí v kódu. V rámci této analýzy tedy nebyl nalezen volně využívaný nástroj, který by řešil detekci licencí a zpracování podmínek použití na webových stránkách.

Avšak byla nalezena bakalářská práce [20], která se zabývá tématem *License-aware web crawling*. V práci je popsán vývoj nástroje, který využívá web scraping k extrakci licenčních informací z webových stránek. Na základě těchto informací je za pomoci NLP technik detekována CC licence a je zobrazena extrahovaná část ToS textu. Práce byla zhotovena v rámci projektu LAW4OSAI (License-Aware Web Crawling for Open Search AI), který má za cíl propojit právní expertizu s technologickými pokroky ve vyhledávání na webu. Prostřednictvím licencovaného procházení webového obsahu chce projekt podpořit transparentnější a právně souladný přístup k vyhledávání [21]. Ačkoli řešení této práce nikterak nevyužívá jazykové modely a dále nezpracovává získaný ToS text, tak stále může posloužit jako inspirace pro návrh a implementaci nástroje v rámci této práce.

## 1.6 Shrnutí

Na základě analýzy dílčích témat bylo identifikováno několik poznatků, které budou dále využity při návrhu a implementaci nástroje. Bylo zjištěno, že licence a ToS na webových stránkách jsou klíčovými dokumenty, které definují pravidla pro využití obsahu. Nejčastěji využívané licence na webových stránkách jsou Creative Commons, které umožňují autorům jednoduše definovat podmínky pro využití obsahu. Zároveň bylo zjištěno, že reprezentace těchto dokumentů není standardizovaná a může se lišit na základě konkrétního obsahu. Nicméně byly identifikovány časté textové vzory, které se používají pro zobrazení licencí a ToS na webových stránkách. Také bylo zjištěno, že společné umístění těchto dokumentů se nachází zpravidla v patičce webových stránek. Pro extrakci informací z webových stránek byl analyzován web scraping, který umožňuje získat strukturovaná data z nestrukturovaných nebo polostrukturovaných dat. Následně pro zpracování získaných dat byly prozkoumány jazykové modely, které umožňují analýzu textu a porozumění jeho kontextu. Nakonec byly identifikovány existující nástroje a řešení, ze kterých jako nejprínosnější byla vybrána bakalářská práce [20], která se zabývá tématem *License-aware web crawling*.

*Kapitola popisuje návrh nástroje pro analýzu CC licencí a ToS na webových stránkách. Definuje funkční a nefunkční požadavky, navrhuje architekturu s procesy extrakce a zpracování dat a představuje prototyp webové aplikace s API a uživatelským rozhraním.*

## 2.1 Požadavky na nástroj

Požadavky na nástroj představují základ jeho návrhu a následné implementace, jelikož definují očekávané chování a vlastnosti systému. Požadavky v této práci vycházejí z potřeby automatizovat analýzu CC licencí a ToS na webových stránkách, což je úkol, který vyžaduje efektivní zpracování dat a jejich srozumitelnou prezentaci uživateli. V rámci návrhu jsou tak požadavky rozděleny do dvou kategorií: funkční a nefunkční. Funkční požadavky specifikují konkrétní funkcionality, které nástroj musí nabízet, aby splnil svůj účel, jako je detekce licencí nebo analýza textů pomocí jazykových modelů. Naopak nefunkční požadavky se zaměřují na kvalitativní aspekty nástroje, včetně jeho výkonu, spolehlivosti a uživatelské přívětivosti.

### 2.1.1 Funkční požadavky

**Detekce CC licence:** Detekce použitých Creative Commons licencí na webových stránkách.

**Klasifikace licence:** Rozpoznání konkrétního typu CC licence.

**Detekce ToS odkazů:** Identifikace klíčových textových vzorů vedoucích k ToS textu.

**Zpracování ToS textu:** Analýza extrahovaného ToS textu pomocí jazykových modelů.

**Vizualizace výsledků:** Vizualizace detekovaných licencí a zpracovaných podmínek použití ve webovém rozhraní.

**API:** Možnost přístupu k funkcím nástroje pomocí API, které by mělo být navrženo s REST API principy.

### 2.1.2 Nefunkční požadavky

**Výkon:** Nástroj by měl za standardních podmínek detekovat ToS text a CC licenci na webové stránce do 5 sekund.

**Spolehlivost:** Detekce licencí a ToS odkazů by měla dosahovat přesnosti alespoň 90%.

**Přívětivost:** Nástroj by měl být intuitivní a snadno použitelný.

## 2.2 Architektura nástroje

Nástroj pro detekci licencí a podmínek použití na webu je navržen jako knihovna, která poskytuje funkcionality pro extrakci a analýzu dat z webových stránek. Tato knihovna je nezávislou komponentou, která může být využita v různých kontextech, jako jsou skripty automatizovaných procesů nebo webové aplikace. Hlavním cílem tohoto návrhu je zajistit flexibilitu a znovupoužitelnost, aby nástroj mohl být snadno integrován do různých prostředí bez nutnosti dalších nadstavb. Obecný návrh architektury nástroje je znázorněn na obrázku 2.1.

### 2.2.1 Knihovna

Proces návrhu knihovny je rozdělen do dvou hlavních částí, aby bylo možné efektivně využít jazykové modely při zpracování textů ToS. První část se zaměřuje na základní analýzu a extrakci dat z webových stránek s cílem získat co nejvíce relevantních informací. Druhá část zahrnuje využití jazykových modelů, které provádějí hlubší analýzu extrahovaných textů, jako je sumarizace nebo odpovídání na dotaz. Toto rozdělení umožňuje uživateli získat data z webových stránek ještě před tím, než jsou zpracována jazykovými modely. Díky tomu je možné nástroj využít i bez použití jazykových modelů, což může být užitečné v případech, kdy uživatelé nechtějí nebo nemohou využívat API jazykových modelů.

Navržená knihovna je založena na modulární architektuře, která zajišťuje rozdělení funkcionalit do samostatných komponent. Každá komponenta je zodpovědná za konkrétní část zpracování dat, což umožňuje snadnou rozšiřitelnost. Jak lze vidět na obrázku 2.1, knihovna se skládá z následujících komponent:

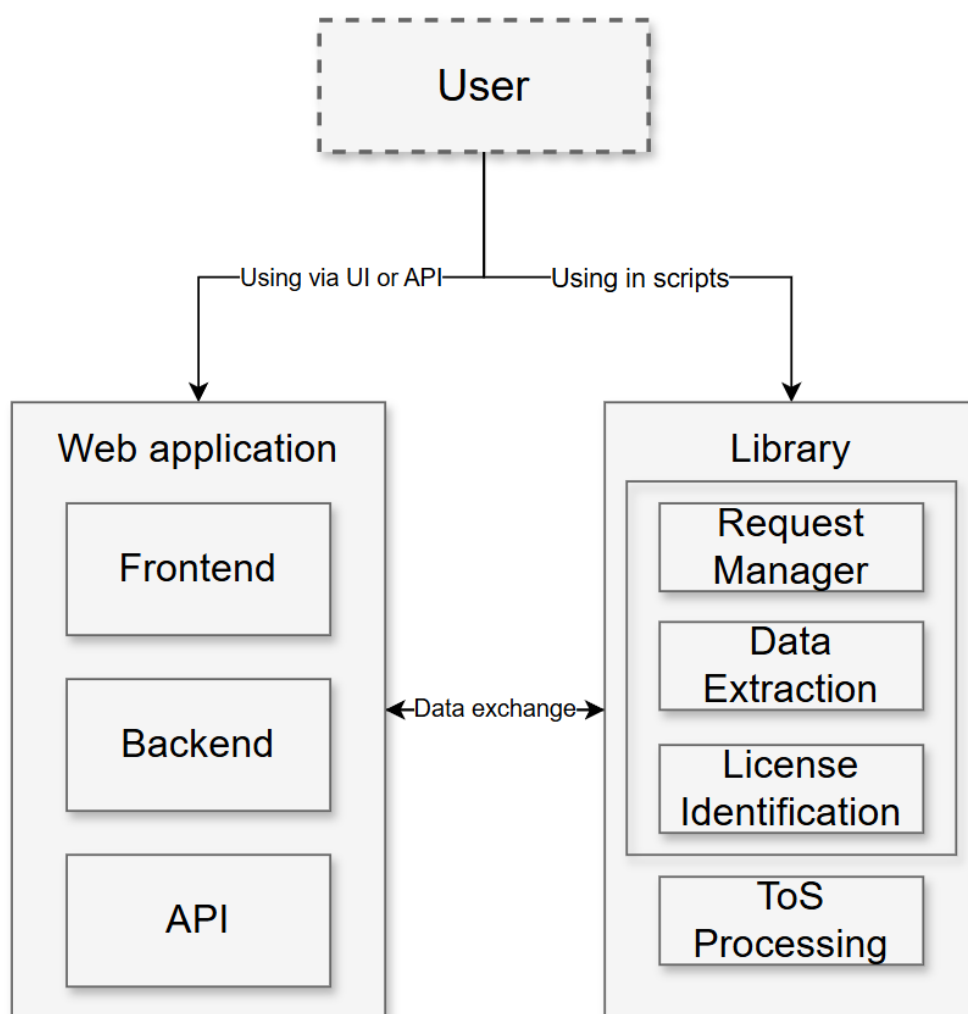
**Modul pro správu požadavků:** Zajišťuje komunikaci s webovými servery, včetně načítání stránek a ověřování přístupu podle pravidel definovaných

v souboru robots.txt. Slouží jako vstupní bod pro získávání dat z webových zdrojů.

**Modul pro extrakci dat:** Zodpovědný za parsování HTML obsahu stránek a získání relevantních informací.

**Modul identifikace licence:** Provádí analýzu textu a odkazů za účelem identifikace konkrétních typů licencí, na základě předdefinovaných vzorů a klíčových slov.

**Modul zpracování ToS:** Oddělený modul, který zpracovává vyfiltrovaný ToS text pomocí jazykových modelů.



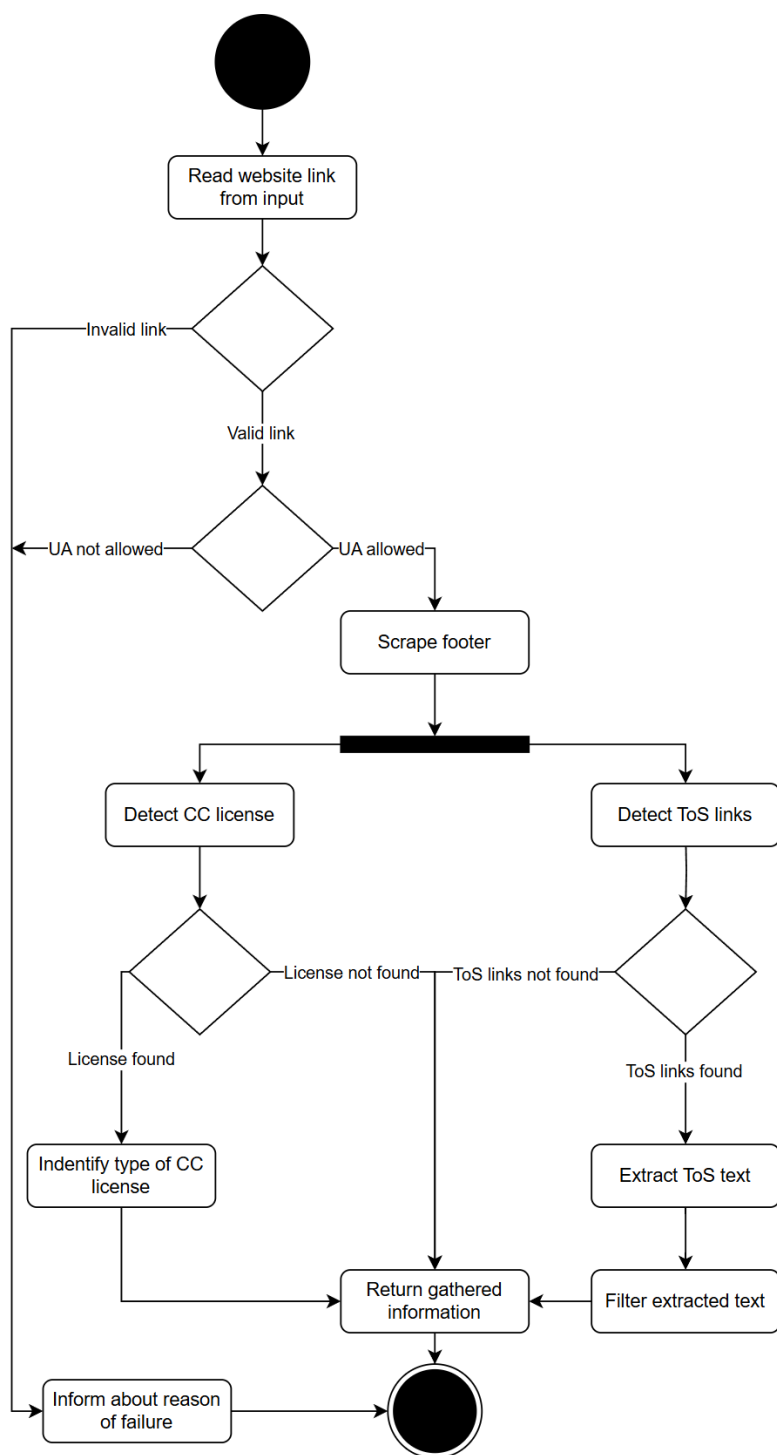
■ Obrázek 2.1 Architektura nástroje

### 2.2.1.1 Tok dat

V rámci návrhu je důležité definovat, jakým způsobem budou jednotlivé komponenty spolupracovat a jakým způsobem budou data předávána mezi nimi. Pro tento účel byly navrženy vývojové diagramy. Tyto diagramy představují nástroj pro vizualizaci a popis procesů navrženého nástroje. Umožňují tak přehledně znázornit tok dat a jednotlivé kroky zpracování, což usnadňuje pochopení návrhu nástroje.

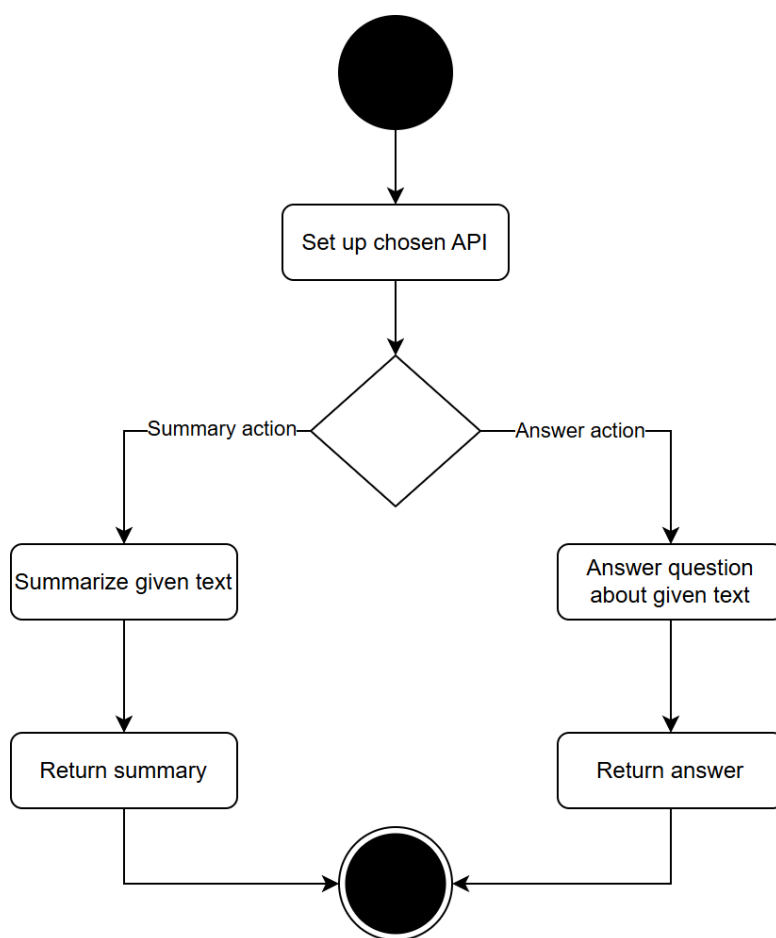
První vývojový diagram 2.2 znázorňuje proces základní analýzy a extrakce dat z webových stránek, což tvoří první hlavní část navrženého nástroje. Proces začíná zadáním URL adresy webové stránky, která má být analyzována. Následně systém ověří, zda je zadaný odkaz platný, a pokud není, proces se ukončí s informací o neplatnosti odkazu. V případě platného odkazu nástroj zkontroluje soubor robots.txt, aby zjistil, zda je web scraping obsahu povolen pro daného UA (User Agent). Pokud povolen není, proces se ukončí s informací o nepovoleném přístupu. V opačném případě nástroj provede web scraping stránky a pokračuje dvěma paralelními větvemi. V první větvi hledá Creative Commons licenci a pokud ji nalezne, identifikuje její typ, a tuto informaci vrátí jako součást výsledků. Pokud licenci nenalezne, vrátí odpovídající informaci. Ve druhé větvi nástroj vyhledává odkazy na ToS text. Pokud odkazy nejsou nalezeny, proces vrátí informaci o jejich nenalezení. V opačném případě nástroj z nalezených odkazů ToS text extrahuje a tento text filtruje, aby odstranil irelevantní části. Nakonec jsou vráceny všechny dosud získané informace, které mohou zahrnovat typ licence, extrahovaný ToS text nebo důvod selhání, pokud některý krok neproběhl úspěšně.





■ Obrázek 2.2 Vývojový diagram 1. části nástroje

Druhý vývojový diagram 2.3 popisuje proces analýzy extrahovaných textů pomocí jazykových modelů, což představuje druhou hlavní část nástroje. Proces začíná nastavením rozhraní API pro komunikaci s jazykovým modelem, který bude použit pro analýzu. Následně uživatel volí jednu ze dvou hlavních akcí. První možností je sumarizace textu, kdy jazykový model zpracuje vstupní text a vrátí jeho stručné shrnutí. Druhou možností je odpověď na dotaz, kdy uživatel zadá otázku týkající se vstupního textu a jazykový model vrátí vygenerovanou odpověď. Výsledkem procesu je tedy buď shrnutí textu, nebo odpověď na zadanou otázku, v závislosti na zvolené akci.



■ Obrázek 2.3 Vývojový diagram 2. části nástroje

## 2.2.2 Prototyp webové aplikace

V průběhu tvorby zadání této bakalářské práce bylo s vedoucím práce rozhodováno o různých způsobech, jak nástroj používat. Mimo využití nástroje, jakožto samostatné knihovny, bylo rozhodnuto o vytvoření prototypu webové aplikace, která by umožnila uživatelům snadno přistupovat k funkcím nástroje nejen prostřednictvím uživatelského rozhraní, ale také pomocí API. Cílem tak není navrhnout plnohodnotnou webovou aplikaci obsahující plnohodnotnou databázi, uživatelské účty a další běžné části webových aplikací, ale spíše jednoduchý prototyp, který demonstruje funkčnost navrženého nástroje.

### 2.2.2.1 Backend

Backend webové aplikace je navržen jako centrální komponenta prototypu, která zprostředkovává zpracování požadavků od uživatele a koordinuje analýzu dat souvisejících s ToS a licencemi webových stránek. Jeho hlavním cílem je zajistit propojení mezi uživatelským rozhraním a navrženou knihovnou. Součástí backendu je také API, které umožňuje externí přístup k funkcím nástroje. Architektura backendu je navržena jako modulární systém:

**Router:** Komponenta odpovědná za směrování požadavků od uživatele a jejich předání příslušným službám. Definuje cesty pro interakci s uživatelem, například zadání URL adres nebo žádosti o pokročilou analýzu jazykovými modely.

**Services:** Služby, které zajišťují koordinaci mezi routerem, API a analytickým nástrojem. Tyto služby zpracovávají požadavky, předávají data ke zpracování a vracejí výsledky ve formátu vhodném pro obě rozhraní.

**API:** Programové rozhraní, které umožňuje externí přístup k funkcím nástroje. API je navrženo tak, aby zrcadlilo hlavní funkce systému a poskytovalo strukturované odpovědi pro analýzu, sumarizaci a odpovídání na otázky, čímž doplňuje interakci přes frontend.

Tok dat v backendu začíná přijetím požadavku z uživatelského rozhraní či API prostřednictvím routeru. Router předá požadavek službám, které komunikují s knihovnou pro provedení analýzy webových stránek. Výsledky jsou poté zpracovány a vráceny ve strukturované podobě a to buď frontendu pro zobrazení, nebo klientovi přes API. Tento návrh zajišťuje oddělení prezentační a analytické vrstvy a díky integraci API rozšiřuje možnosti využití systému.

### 2.2.2.2 API

Navržené API má za cíl poskytnout alternativní rozhraní pro přístup k funkcím nástroje prostřednictvím HTTP požadavků, což umožňuje jeho využití bez nutnosti interakce přes uživatelské rozhraní. API je určeno pro externí aplikace, skripty nebo automatizované procesy. Je navrženo jako REST API, protože tento typ rozhraní je jednoduchý, široce používaný a vhodný pro komunikaci s různými typy klientů. Při komunikaci jsou vstupní i výstupní data ve formátu JSON, což zajišťuje snadné zpracování dat na straně klienta.

Všechny operace jsou navrženy jako asynchronní, aby bylo možné efektivně zvládat časově náročné úkoly, jako je web scraping a generování textů pomocí jazykových modelů. Každý endpoint používá HTTP metodu POST pro spuštění úlohy. Po inicializaci úkolu vrací identifikátor úlohy a odkaz na další endpoint, který využívá metodu GET. Tento doplňkový endpoint slouží k monitorování stavu úlohy a poskytuje přístup k výsledkům. Všechny dílčí endpointy jsou popsány v tabulce 2.1.

Pro analýzu webových stránek slouží endpoint `/scans`, který zpracovává JSON požadavek se seznamem URL adres. Výsledky, včetně informací o nalezených licencích a extrahovaných textech ToS, jsou dostupné přes doplňkový endpoint `/scans/<task_id>`.

Sumarizaci extrahovaných textů ToS zajišťuje endpoint `/summaries`. Tento endpoint přijímá JSON požadavek s textem k sumarizaci a specifikací API rozhraní včetně příslušného API klíče. Shrnutý text lze získat prostřednictvím `/summaries/<task_id>`.

K zodpovídání otázek týkajících se sumarizovaného textu je navržen endpoint `/answers`. Přijímá JSON požadavek obsahující otázku, sumarizovaný text a specifikaci API rozhraní s API klíčem. Odpověď je poskytována přes `/answers/<task_id>`.

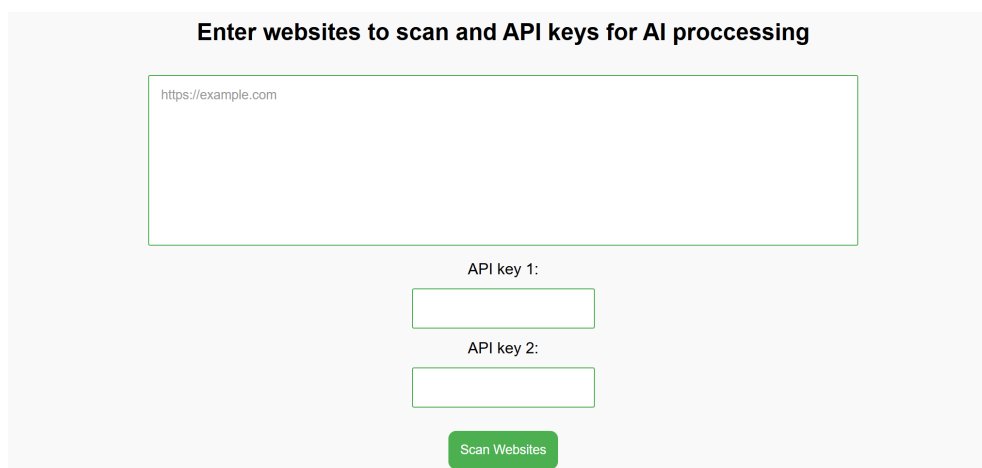
Endpoint	Metoda	Popis
<code>/scans</code>	POST	Analýza webových stránek
<code>/scans/&lt;task_id&gt;</code>	GET	Kontrola stavu analýzy a její vyzvednutí
<code>/summaries</code>	POST	Sumarizace textu
<code>/summaries/&lt;task_id&gt;</code>	GET	Kontrola stavu sumarizace a její vyzvednutí
<code>/answers</code>	POST	Odpověď na otázku
<code>/answers/&lt;task_id&gt;</code>	GET	Kontrola stavu odpovědi a její vyzvednutí

**Tabulka 2.1** Přehled endpointů navrženého API

### 2.2.2.3 Frontend

Frontend je navržen jako uživatelské rozhraní prototypu, jehož účelem je umožnit interakci s backend částí a prezentovat dílčí výsledky nástroje. Hlavním cílem frontendu je poskytnout přehledné prostředí, ve kterém může uživatel zadávat vstupní data, spouštět analýzu a prohlížet zpracované výsledky. Návrh je doplněn ukázkami, které znázorňují, jak by jednotlivé části rozhraní a jejich zamýšlené funkce mohly vypadat.

**Úvodní rozhraní:** Úvodní stránka slouží jako vstupní bod aplikace, kde uživatel nalezne formulář pro zadání URL adres a API klíčů. Jak ukazuje Obrázek 2.4, rozhraní umožňuje zadat více URL adres najednou a volitelně přidat API klíče pro zpracování pomocí jazykových modelů.



The screenshot shows a web form titled "Enter websites to scan and API keys for AI processing". It features a large text input area containing the placeholder text "https://example.com". Below this, there are two input fields for "API key 1:" and "API key 2:". At the bottom of the form is a green button labeled "Scan Websites".

■ **Obrázek 2.4** Úvodní rozhraní

**Výsledkové rozhraní:** Tato část aplikace slouží k analýze obsahu jednotlivých webových stránek. U každé URL adresy jsou zobrazeny odkazy na ToS, nalezené licence a extrahovaný text. Jak je patrné z návrhu na Obrázku 2.5, uživatel zde může vybrat API službu a požádat o sumarizaci obsahu pomocí tlačítka „Generate Summary“.

<https://www.example.com/>

Links to ToS information:  
<https://www.example.com/terms>

Licenses mentioned in ToS: license1, license2  
▼ Extracted ToS text

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean at feugiat nisi. Morbi et felis fringilla, eleifend lectus in, porta metus. Cras fermentum rhoncus sem vel porta. Donec sagittis felis orci, vitae semper quam consectetur vel. Ut vitae congue dolor. Mauris non nisi lectus. Etiam pulvinar consectetur orci ac commodo.

☒ API 1  
☐ API 2

Generate Summary

■ Obrázek 2.5 Výsledkové rozhraní

**Rozhraní souhrnu:** Po vygenerování sumarizace je uživateli zobrazen shrnutý text a formulář pro kladení doplňujících otázek. Na Obrázku 2.6 je zachycen návrh této sekce, kde lze zadat otázku, zvolit API službu pro zpracování a následně získat odpověď přímo na stejné stránce.

Summary

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean at feugiat nisi. Morbi et felis fringilla, eleifend lectus in, porta metus. Cras fermentum rhoncus sem vel porta. Donec sagittis felis orci, vitae semper quam consectetur vel. Ut vitae congue dolor. Mauris non nisi lectus. Etiam pulvinar consectetur orci ac commodo.

Here you can ask a question about the summary

☒ API 1  
☐ API 2

Submit question

■ Obrázek 2.6 Rozhraní souhrnu

## Implementace

*Tato kapitola popisuje praktickou realizaci navrhovaného nástroje. Zaměřuje se na výběr technologií, strukturu modulů, implementaci klíčových funkcionalit a vývoj prototypu webové aplikace. Představeny jsou také detaily formátu výstupů a volby parametrů pro detekci licencí a zpracování ToS.*

### 3.1 Výběr technologií

Pro implementaci nástroje se využívá programovací jazyk Python, který tvoří základ nejen navrhované knihovny, ale také prototypu webové aplikace. Níže je přehled hlavních používaných technologií:

**Python** Zvolen pro svou univerzálnost a rozsáhlý ekosystém knihoven pro web scraping a zpracování přirozeného jazyka. Navíc je široce používán v oblasti strojového učení a umělé inteligence, což z něj činí ideální volbu pro práci s jazykovými modely.

**Requests** Knihovna pro odesílání HTTP požadavků a získávání obsahu webových stránek [22]. V rámci nástroje se používá pro načítání HTML obsahu stránek včetně dokumentu robots.txt a zajištění správného nastavení UA pro web scraping.

**BeautifulSoup** Knihovna pro parsování HTML a extrakci informací z webových stránek [23]. Umožňuje efektivní prohledávání DOM objektů a identifikaci prvků, jako jsou odkazy na ToS nebo CC licence.

**Huggingface\_hub** Knihovna pro přístup k jazykovým modelům z platformy Hugging Face [24]. Využívá se pro svůj široký výběr open source modelů a snadnou integraci v rámci Python prostředí.

**Google-genai** Knihovna pro interakci s API rozhraním generativní umělé inteligence společnosti Google [25]. Oproti Hugging Face tak reprezentuje proprietární alternativu pro zpracování textu a generování odpovědí.

**Flask** Lehký webový framework pro vývoj backendu [26]. Byl zvolen pro svou jednoduchost a schopnost rychle vytvořit funkční rozhraní potřebné pro prototyp webové aplikace.

**Celery** Nástroj pro správu asynchronních úloh na pozadí [27]. Využívá se ke zpracování časově náročných operací, jako je sumarizace ToS textů v rámci API dotazů.

**Redis-py** Python klient pro databázi Redis [28]. V implementaci zajišťuje efektivní správu fronty úloh pro Celery a ukládání dočasných výsledků.

## 3.2 Moduly

Implementace je rozdělena do čtyř hlavních modulů: Správa požadavků, Extrakce dat, Identifikace licence a Zpracování ToS. První tři moduly jsou integrovány do centrální třídy *LicenseDetector*, která je vstupním bodem celé knihovny. Tato třída inicializuje příslušné komponenty a řídí jejich spolupráci při zpracování webového obsahu prostřednictvím metody *scan\_websites*, jež přijímá seznam URL adres a vrací výsledky analýzy.

Modul Zpracování ToS je navržen jako samostatná komponenta využívající jazykové modely pro analýzu podmínek použití. Není přímo propojen s *LicenseDetector*, což umožňuje jeho nezávislé použití v různých kontextech.

Následující podkapitoly podrobně popisují implementaci jednotlivých modulů, včetně jejich klíčových funkcí a technických detailů.

### 3.2.1 Správa požadavků

Modul pro správu požadavků zajišťuje komunikaci s webovými servery a zároveň dbá na dodržování etických zásad při přístupu k webovým stránkám. Je implementován prostřednictvím třídy *RequestManager*, která pro správu HTTP požadavků využívá knihovnu *requests*. Při inicializaci třídy *RequestManager* dochází k vytvoření relace, která umožňuje efektivní opakované odesílání požadavků a správu stavových informací napříč jednotlivými dotazy. Klíčové funkce modulu zahrnují:

**Ověření přístupu podle *robots.txt*:** Metoda *is\_allowed\_by\_robots* kontroluje, zda je pro danou URL a UA povolen web scraping. Pokud soubor *robots.txt* není dostupný, modul předpokládá, že přístup je povolen.

**Načítání souboru *robots.txt*:** Metoda *fetch\_robots* získává obsah souboru *robots.txt* z dané URL adresy s využitím knihovny *Protego* [29].

**Načítání webových stránek:** Metoda *fetch\_page* načítá obsah webové stránky ze zadané URL adresy. Využívá HTTP GET požadavek a zpracovává případné chyby, jako je neplatná odpověď serveru nebo překročení časového limitu spojení.



**Správa User-Agent:** Metody *set\_user\_agent* a *get\_user\_agent* umožňují správu hlavičky User-Agent, což je užitečné při testování vlastního UA.

Tento modul tak tvoří základ pro další komponenty nástroje, které závisí na získání a ověření webového obsahu.

### 3.2.2 Extrakce dat

Modul pro extrakci dat je zodpovědný za zpracování HTML obsahu webových stránek a získávání relevantních informací, jako jsou odkazy na licence a ToS. Je implementován třídou *DataExtractor*, která využívá knihovnu BeautifulSoup pro parsování HTML struktury. Třída je navržena tak, aby fungovala ve spolupráci s modulem pro správu požadavků, který je předán při inicializaci. Klíčové funkce modulu jsou:

**Parsování HTML obsahu:** Metoda *parse\_html* převádí surový HTML obsah na objekt *BeautifulSoup*, který umožňuje prohledávání a manipulaci s DOM strukturou stránky.

**Extrakce odkazů z patičky:** Metoda *extract\_footer\_links* prohledává elementy *<footer>* na stránce a identifikuje licenční a ToS odkazy. Odkazy jsou identifikovány na základě klíčových slov a regulárních výrazů.

**Extrakce relevantního textu:** Metoda *extract\_relevant\_text* prochází textový obsah stránek obsažený v elementech *<p>*, *<li>* a *<span>*, které běžně nesou rozvinutý či kontextový text na stránce, na rozdíl od nadpisových elementů. Rozděluje text na věty a vybírá ty, které obsahují klíčová slova. Výsledný text je následně spojen do jednoho řetězce pro další zpracování.

Zmíněná klíčová slova a regulární výrazy jsou definovány v souboru parametrů *parameters.py*, který je popsán v podkapitole 3.4. Výběr těchto vzorů probíhal v průběhu vývoje a testování nástroje na různých webových stránkách. Hlavním cílem bylo vybalancovat přesnost a citlivost detekce, aby bylo možné efektivně identifikovat relevantní odkazy a texty.

### 3.2.3 Identifikace licence

Modul pro identifikaci licence je zodpovědný za detekci a klasifikaci licencí obsažených v extrahovaném textu nebo odkazech. Je implementován třídou *LicenseIdentifier*, která vrací identifikované licence ve standardizovaném formátu. Mezi klíčové funkce modulu patří:

**Detekce licencí Creative Commons:** Metoda *determine\_cc\_license* analyzuje zadanou URL adresu a text pro identifikaci CC licencí. Nejprve

kontroluje přítomnost licence CC0. Pokud není CC0 nalezena, hledá specifické CC licence v URL na základě kombinací verzí a typů licencí (např. „licenses/by/4.0“ pro CC-BY-4.0). Pokud URL neobsahuje licenci, používá regulární výrazy k extrakci licence z textu a mapuje textové názvy na standardní formát (např. „Attribution“ na „BY“). Nakonec vrátí identifikovanou licenci jako řetězec (např. „CC-BY-4.0“) nebo „Unknown“, pokud licence není nalezena.

**Extrakce obecných licencí:** Metoda *extract\_licenses* prohledává zadaný text a hledá shody s předdefinovanými vzory licencí. Každý vzor odpovídá konkrétnímu typu licence (např. „MIT“, „Apache“). Výsledkem je seznam unikátních názvů licencí nalezených v textu.

Oproti původnímu návrhu modulu je implementace rozšířena o detekci jiných typů licencí, než jsou licence Creative Commons. Motivace pro toto rozšíření vzešla během testování nástroje na různých webových stránkách, kde v rámci textu ToS byly zmíněny i jiné typy licencí. Díky tomu je nástroj schopen uživateli poskytnout souhrn zmíněných licencí ještě před zpracováním ToS jazykovými modely.

Stejně jako u modulu pro extrakci dat, i zde jsou klíčová slova a regulární výrazy definovány v souboru parametrů. Tento přístup umožňuje jednoduše pravidla aktualizovat a přizpůsobit bez nutnosti měnit samotný kód modulu.

### 3.2.4 Zpracování ToS

Modul pro zpracování podmínek použití je dle návrhu implementován jako samostatný modul, který je rozdělen do několika částí. Hlavní funkcionalitou modulu je zpracování textového obsahu ToS získaného z předchozích modulů a jeho transformace do stručného shrnutí nebo odpovědi na specifické otázky pomocí jazykových modelů. Klíčové komponenty modulu zahrnují:

**Správa modelů:** *ModelManager* je factory třída, která slouží jako rozhraní pro výběr a inicializaci jazykových modelů. Umožňuje dynamicky získat instanci modelu na základě jeho názvu. Tato flexibilní konstrukce zajišťuje, že systém lze jednoduše rozšířit o případné další modely.

**Abstraktní třída modelů:** *BaseModel* je abstraktní třída, která definuje společné rozhraní pro všechny jazykové modely. Obsahuje metodu pro přípravu dotazů (promptů), která formátuje vstupní data do podoby vhodné pro model, a metodu pro validaci vstupních dat. Validace zajišťuje, že vstupní data obsahují povinné položky, jako je obsah ToS nebo název webu, čímž se předchází chybám při zpracování.

**Implementace modelu *Mistral*:** Třída *Mistral* integruje model *Mixtral-8x7B-Instruct-v0.1* prostřednictvím knihovny *huggingface\_hub*. Obsahuje metody *summarize* a *answer\_question*, které generují shrnutí a odpovědi na

základě zadaného promptu. Třída zároveň zajišťuje ošetření chyb, jako je neplatný API klíč nebo nedostupnost modelu. Výstupem je text, který může být dále zpracován nebo uložen.

**Implementace modelu *Gemini*:** Třída *Gemini* využívá model *gemini-2.5-pro-exp-03-25* prostřednictvím knihovny *google.generativeai*. Stejně jako třída *Mistral* implementuje metody *summarize* a *answer\_question*, které komunikují s modelem přes API a zpracovávají chyby, jako je neplatný API klíč.

Při implementaci tohoto modulu bylo experimentováno s různými jazykovými modely, a to jak open source, tak proprietárními. Ze skupiny proprietárních modelů byl vybrán *gemini-2.5-pro-exp-03-25* pro svou přístupnost prostřednictvím bezplatného API a svým bezkonkurenčním výsledkům. Z open source modelů dostupných na bezplatné verzi Hugging Face API (modely do 10GB) byl vybrán *Mixtral-8x7B-Instruct-v0.1*. Výběr tohoto modelu nebyl tak jednoznačný a probíhal v průběhu detailnějšího testování, popsaného v příslušné podkapitole 4.2.1.

Co se týče zvolených promptů pro generování shrnutí a odpovědí na otázky, byly navrženy s důrazem na přesnost a stručnost. Pro sumarizaci je prompt strukturován tak, aby model obdržel název webu a kompletní text ToS s instrukcí vytvořit shrnutí. Pro odpovědi na otázky prompt obsahuje souhrn ToS a konkrétní otázku v rámci kontextu shrnutí. To zajišťuje, že model reaguje na situaci, kdy uživatel klade dotazy mimo tematický rámec shrnutí.

■ **Výpis kódu 3.1** Metoda *prepare\_prompt* třídy *BaseModel*

```
1 def _prepare_prompt(self, data, question=None):
2     if question:
3         return (
4             f"Summary of the Terms of Service
              text:\n{data['summary']}\n\n"
5             f"Answer the following question about the ToS
              summary:\n{question}"
6         )
7     return f"Summarize the following Terms of Service text
              from {data['website']}: \n{data['content']}"
```

### 3.3 Formát výsledků

Výstupní data první části nástroje jsou generována již dříve zmíněnou třídou *LicenseDetector*, která shromažďuje a strukturovaně vrací informace o licenčních údajích získaných z analyzovaných webových stránek. Hlavní metodou pro generování výsledků je *scan\_websites*, která zpracovává seznam URL adres a vrací seznam slovníků, kde každý slovník představuje výsledky pro jednu

webovou stránku. Formát výsledků 3.2 je navržen tak, aby byl konzistentní a snadno zpracovatelný pro další použití:

■ **Výpis kódu 3.2** JSON formát výsledků

```
1  [  
2    {  
3      "website": "string",  
4      "invalidUrl": "boolean",  
5      "blockedByRobotsTxt": "boolean",  
6      "licenseLink": "string | null",  
7      "licenseType": "string | null",  
8      "relevantLinks": [],  
9      "licenseMentions": [],  
10     "content": "string | null"  
11   },  
12   ...  
13 ]
```

**website:** URL analyzované stránky.

**invalidUrl:** Logická hodnota označující, zda byla vstupní URL neplatná.

**blockedByRobotsTxt:** Logická hodnota signalizující, zda byl přístup zablokován pravidly *robots.txt*.

**licenseLink:** URL odkazu na CC licenci.

**licenseType:** Typ CC licence.

**relevantLinks:** Seznam relevantních odkazů (ToS a podobné dokumenty) získaných z patičky stránky.

**licenseMentions:** Seznam názvů licencí nalezených v textovém obsahu relevantních odkazů.

**content:** Extrahovaný textový obsah z relevantních odkazů, který může být dále zpracován jazykovými modely.

Výstupy druhé části nástroje jsou generovány dvěma hlavními metodami třídy *BaseModel*, které jsou implementovány v jednotlivých třídách jazykových modelů. Metoda *summarize* vrací slovník, který rozšiřuje vstupní data o klíč *summary* se stručným shrnutím textu ToS. Aby mohla tato metoda fungovat správně, musí vstupní data obsahovat klíče *website* (název webu) a *content* (text podmínek použití získaný v první části nástroje). Naproti tomu metoda *answer\_question* generuje textovou odpověď na konkrétní dotaz, který se týká již vytvořeného shrnutí. Pro její správné fungování je vyžadováno, aby vstup obsahoval klíč *summary* spolu s položkou *question*, která formuluje otázku k danému shrnutí.

### 3.4 Definice parametrů

Soubor *parameters.py* slouží jako jednotné místo pro definici klíčových slov a regulárních výrazů využívaných při detekci i extrakci licencí, odkazů a textů ToS. Následující texty popisují jednotlivé části tohoto souboru:

#### ■ Výpis kódu 3.3 Klíčová slova pro odkazy a textový obsah ToS

```
1 RELEVANT_LINKS_KEYWORDS = ["terms", "legal", "license",
2                             "licensing", "use", "policy", "copyright"]
3 RELEVANT_TEXT_KEYWORDS = [
4     "intellectual property", "right", "authorised",
5     "commercial use", "non-commercial use",
6     "fair use", "license", "copyright", "creative commons",
7     "cc by", "gpl", "apache", "mit", "bsd"]
```

Tyto konstanty definují klíčová slova pro modul *DataExtractor*. První obsahuje slova pro identifikaci odkazů v patičkách webových stránek vedoucích k ToS. Druhá zahrnuje slova pro extrakci textového obsahu souvisejícího s licenčními informacemi.

#### ■ Výpis kódu 3.4 Parametry licencí Creative Commons

```
1 CC_VERSIONS = ['1.0', '2.0', '2.5', '3.0', '4.0']
2 CC_TYPES = ['BY', 'BY-SA', 'BY-ND', 'BY-NC', 'BY-NC-SA',
3             'BY-NC-ND']
4 CC_TYPE_MAP = {
5     'attribution': 'BY', 'by': 'BY', 'noncommercial': 'NC',
6     'non-commercial': 'NC', 'nc': 'NC',
7     'sharealike': 'SA', 'share-alike': 'SA', 'sa': 'SA',
8     'noderivatives': 'ND', 'no-derivatives': 'ND', 'nd':
9     'ND'}
```

Tyto konstanty definují parametry pro zpracování CC licencí. První specifikuje podporované verze licencí, druhá uvádí dostupné typy licencí, a třetí mapuje textové popisy (např. „attribution“ na „BY“) pro standardizaci výstupů.

#### ■ Výpis kódu 3.5 Regulární výraz pro Creative Commons

```
1 CC_PATTERN = re.compile(
2     r'\b(?:CC[\s-]+(?:?:BY|Attribution|SA|
3     r'Share[\s-]?Alike|NC|Non[\s-]?Commercial|
4     r'ND|No[\s-]?Derivatives)[\s-]?)+
5     r'(?:\d\.\d)?|
6     r'Creative[\s-]+Commons[\s-]*(?:
7     r'?:Attribution|Share[\s-]?Alike|
8     r'Non[\s-]?Commercial|No[\s-]?Derivatives|
9     r'BY|SA|NC|ND)[\s-]*)*
10    r'(?:\d\.\d)?|CC0)\b',
11    re.IGNORECASE)
```

Výraz pro detekci CC licencí rozpoznává tři varianty:

1. „CC“ následované jedním nebo více atributy (např. „BY“, „SA“) a volitelným číslem verze (např. „4.0“).
2. „Creative Commons“ s volitelnými atributy a číslem verze.
3. Samostatné „CC0“.

Tento a následující regulární výrazy slouží k identifikaci licencí prostřednictvím modulu *LicenseIdentifier*. Základ těchto výrazů vychází ze seznamu licencí [2].

#### ■ Výpis kódu 3.6 Regulární výraz pro MIT License

```
1 MIT_PATTERN = re.compile(  
2     r'\bMIT[-\s]+(?:\w+\s+){0,3}(?:License|Variant) ',  
3     re.IGNORECASE)
```

Tento výraz pro detekci MIT licence rozpoznává text začínající „MIT“ následovaný mezerou nebo pomlčkou, případně až třemi dalšími slovy, a končící slovem „License“ nebo „Variant“.

#### ■ Výpis kódu 3.7 Regulární výraz pro Apache License

```
1 APACHE_PATTERN = re.compile(  
2     r'\bApache[-\s]+(?:License[-\s])* '  
3     r'(?:(?:v?\d+(?:\.\d+)?|v?\d+(?:\.\d+)?)\b',  
4     re.IGNORECASE)
```

Tento výraz detekující Apache licenci rozpoznává text začínající „Apache“ následovaný mezerou nebo pomlčkou, po kterém může následovat buď „License“ s volitelným číslem verze (např. „2.0“), nebo přímo číslo verze (např. „v2.0“).

#### ■ Výpis kódu 3.8 Regulární výraz pro GNU GPL

```
1 GPL_PATTERN = re.compile(  
2     r'\b(?:GNU[\s-]+(?:A|L)?GPL| '  
3     r'(?:(?:A|L)?GPL[\s-]*(?:v?\d+(?:\.\d+)?| '  
4     r'General\s+Public\s+License)\b',  
5     re.IGNORECASE)
```

Tento výraz pro detekci GNU GPL rozpoznává tři varianty:

1. „GNU“ následované „GPL“ s volitelným prefixem „A“ nebo „L“.
2. „GPL“ s volitelným prefixem „A“ nebo „L“ a číslem verze (např. „v3“ nebo „2.0“).
3. Plný název „General Public License“.

**■ Výpis kódu 3.9** Regulární výraz pro BSD License

```
1 BSD_PATTERN = re.compile(  
2     r'\bBSD[\s-]+(?:\d+[\s-] Clause|License)\b',  
3     re.IGNORECASE)
```

Tento výraz detekující BSD licenci rozpoznává text začínající „BSD“ následovaný mezerou nebo pomlčkou, po kterém následuje buď číslo a „Clause“ (např. „2-Clause“) nebo slovo „License“.

## 3.5 Prototyp webové aplikace

Prototyp je implementován jako jednoduché, avšak funkční rozhraní, které umožňuje uživatelům zadávat URL adresy, zobrazovat výsledky analýzy licenčních informací, generovat shrnutí a odpovídat na otázky týkající se ToS. Implementace je dle návrhu rozdělena na backend, který zpracovává požadavky a komunikuje s moduly nástroje, a frontend, který zajišťuje interaktivní uživatelské rozhraní.

### 3.5.1 Backend

Backend prototypu je implementován pomocí dříve zmiňovaného frameworku Flask. Hlavní komponenty jsou pak definovány v několika třídách, které společně zajišťují routování, logiku zpracování a komunikaci s databází. Klíčové části backendu zahrnují:

**Routování požadavků:** Třída *Router* definuje následující cesty pro zpracování HTTP požadavků:

- `/`  
Zobrazuje domovskou stránku (*index.html*) s formulářem pro zadání URL adres a API klíčů.
- `/scans/<scan_id>`  
Načítá výsledky analýzy z databáze podle *scan\_id* a zobrazuje je v šabloně *scans.html*. Tato cesta je cílem přesměrování z POST požadavku `/scans`, který inicializuje analýzu stránek metodou *scan\_websites* třídy *Services*.
- `/scans/<scan_id>/summaries/<summary_id>`  
Načítá shrnutí ToS z databáze podle *summary\_id* a zobrazuje ho v šabloně *summary.html*. Tato cesta je cílem přesměrování z POST požadavku `/scans/<scan_id>/summaries`, který generuje shrnutí metodou *summarize\_results* třídy *Services* na základě vybraného API (huggingface nebo googleai).

- `/scans/<scan_id>/summaries/<summary_id>/answers/<answer_id>`  
Načítá shrnutí a odpověď na otázku z databáze podle *summary\_id* a *answer\_id* a zobrazuje je opět v šabloně *summary.html*. Tato cesta je cílem přesměrování z POST požadavku `/scans/<scan_id>/summaries/<summary_id>/answers`, který generuje odpověď metodou *answer\_question* třídy *Services*.

**Logika zpracování:** Třída *Services* zajišťuje integraci s moduly nástroje a koordinaci operací. Klíčové funkce zahrnují:

- *scan\_websites*: Volá metodu *scan\_websites* třídy *LicenseDetector* pro analýzu stránek, ukládá výsledky do databáze prostřednictvím třídy *Database* a vrací *scan\_id*.
- *summarize\_results*: Parsuje výsledky detekce, volá metodu *summarize* vybraného jazykového modelu ukládá shrnutí do databáze a vrací *summary\_id*.
- *answer\_question*: Načítá shrnutí z databáze, volá metodu *answer\_question* vybraného modelu, ukládá odpověď do databáze a vrací *answer\_id*.
- Validace vstupů: Zajišťuje kontrolu URL adres, API klíčů a formátu dat. V případě neplatných vstupů vyhazuje výjimky, které jsou zpracovány na úrovni routování.

**Správa dat:** Třída *Database* zpracovává všechny operace s databází, včetně ukládání a načítání výsledků analýzy, shrnutí a odpovědí. Data jsou ukládána pouze dočasně a po určité době jsou automaticky odstraněna.

Ačkoli původní návrh prototypu nepředpokládal použití databáze, velký objem textových dat v podobě extrahovaného obsahu stránek a shrnutí ToS motivoval k zavedení databázového systému pro efektivní správu a uchovávání těchto dat. Z tohoto důvodu byla zvolena jednoduchá databáze Redis, která je rychlá a efektivní pro ukládání dočasných dat. Redis je rovněž využíván v rámci asynchronního zpracování požadavků prostřednictvím Celery v rozhraní API, jak je popsáno v podkapitole 3.5.2 níže, což umožňuje konzistentní správu dat napříč celou aplikací.

Pro zlepšení uživatelského zážitku byla při routování implementována technika Post/Redirect/Get, která zabráňuje opakovanému odesílání formulářů při obnovení stránky a zajišťuje soulad s moderními webovými standardy.

Třída *Services*, přestože je zjednodušena díky oddělení databázové logiky do třídy *Database*, stále koordinuje více operací, jako je validace vstupů, parsování dat a komunikace s modely. Pro budoucí rozšíření by bylo vhodné zvážit její rozdělení do dílčích služeb na samostatné třídy, aby byla zajištěna větší modularita a snadnější údržba.

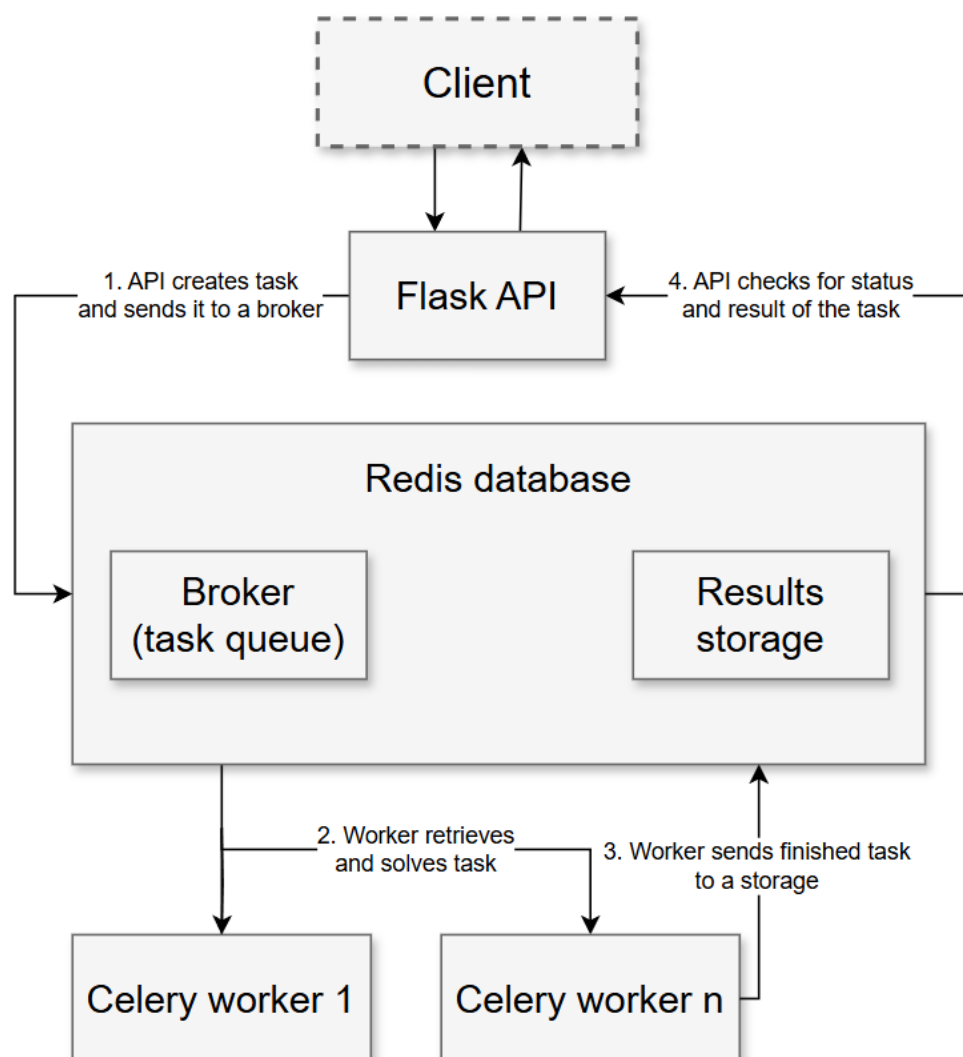


### 3.5.2 API

**Routování požadavků:** Třída *APIRouter* definuje endpointy 2.1 pro úlohy skenování webových stránek, generování shrnutí ToS a zodpovídání otázek. Požadavky jsou přijímány ve formátu JSON a odpovědi vráceny s odpovídajícími HTTP status kódy.

**Asynchronní zpracování:** Úlohy jsou zpracovávány asynchronně prostřednictvím systému Celery. Třída *APIServices* následně validuje vstupní hodnoty a tyto úlohy zpracovává.

Celery je systém určený pro plánování a řízení výpočetně náročných úloh, který umožňuje jejich oddělení od hlavního aplikačního toku. V této implementaci hraje klíčovou roli Redis databáze, která slouží jako broker i jako úložiště výsledků. Celý tento mechanismus je znázorněn na následujícím diagramu 3.1.



■ **Obrázek 3.1** Celery diagram

Broker (zprostředkovatel) je komponenta, která zajišťuje komunikaci mezi API a Celery workery. Když API přijme požadavek na zpracování úlohy, nepředává ji workerovi přímo, ale odesílá ji do Redis brokera, který funguje jako fronta zpráv. Workeri pak z této fronty odebírají úlohy, zpracovávají je a výsledky opět ukládají do Redis úložiště, odkud si je může aplikace nebo klient později vyžádat.

Tímto způsobem je možné úlohy zpracovávat asynchronně a paralelně, což znamená, že uživatel není nucen čekat na jejich dokončení. Po odeslání úlohy klient obdrží HTTP kód 202 (Accepted) spolu s identifikátorem úlohy, pomocí kterého může sledovat její stav a v případě dokončení, si výsledek úlohy vyzvednout. Samopopisné stavy úloh, se kterými se uživatel může setkat, jsou: PENDING, SUCCESS, FAILURE, RETRY a REVOKED.

### 3.5.3 Frontend

Frontend prototypu webové aplikace je implementován pomocí šablonovacího systému Jinja [30], který zajišťuje dynamické generování obsahu na základě dat předaných z backendu. Bez využití samostatného frontendového frameworku, je rozhraní navrženo jako jednoduché, ale funkční, a v rámci prototypu dostačující. Klíčové šablony jsou definovány v souborech *index.html*, *scans.html* a *summary.html*, které tvoří základ uživatelského rozhraní. Mezi funkcionality frontendu patří:

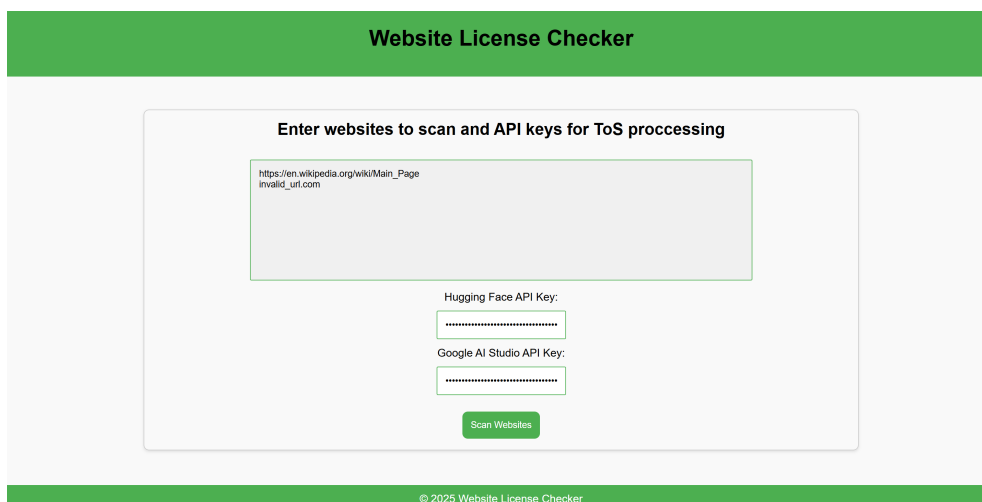
**Zadávání vstupních dat:** Šablona *index.html* obsahuje formulář pro zadání seznamu URL adres webových stránek a API klíčů pro jazykové modely. Uživatel vkládá URL adresy do textového pole a API klíče do vstupních polí typu *password*.

**Zobrazení výsledků analýzy:** Šablona *scans.html* prezentuje výsledky analýzy licencí pro každou zadanou URL adresu v přehledných kartách. Každá karta zobrazuje informace, jako je adresa webu, stav (např. blokování souborem *robots.txt*, neplatná URL), nalezené odkazy na licence, typ licence, relevantní odkazy na ToS a extrahovaný text podmínek použití. Uživatel může rozbalit podrobnosti o ToS a inicializovat generování shrnutí výběrem preferovaného API a odesláním požadavku.

**Zobrazení shrnutí a odpovědí:** Šablona *summary.html* zobrazuje vygenerované shrnutí ToS spolu s použitým API a umožňuje uživateli zadat otázku k shrnutí prostřednictvím formuláře. Vygenerovaná odpověď je následně zobrazena pod samotným dotazem.

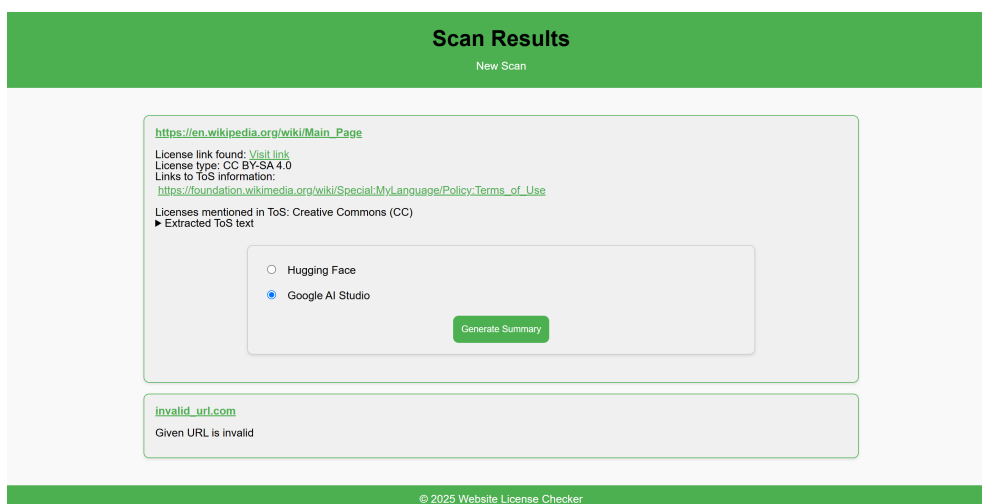
Rozhraní je stylováno pomocí CSS souborů, které zajišťují vizuální konzistenci napříč celou aplikací. Všechny šablony vycházejí ze společné základní struktury definované v souboru *layout.html*, což usnadňuje zachování jednotného vzhledu napříč stránkami. Interaktivita aplikace je záměrně omezena pouze na formuláře a odkazy, bez použití klientského JavaScriptu. Veškerá logika je tak zpracovávána na straně serveru, což přispívá k jednoduchosti implementace. Tento přístup sice omezuje možnosti pokročilejších uživatelských interakcí, nicméně v kontextu prototypu byl vyhodnocen jako plně dostačující.

Vzhled uživatelského rozhraní vychází z předem připravených návrhových obrázků, které sloužily jako podklad pro implementaci jednotlivých šablon. Následující sekvence obrázků ilustruje typický průchod aplikací:



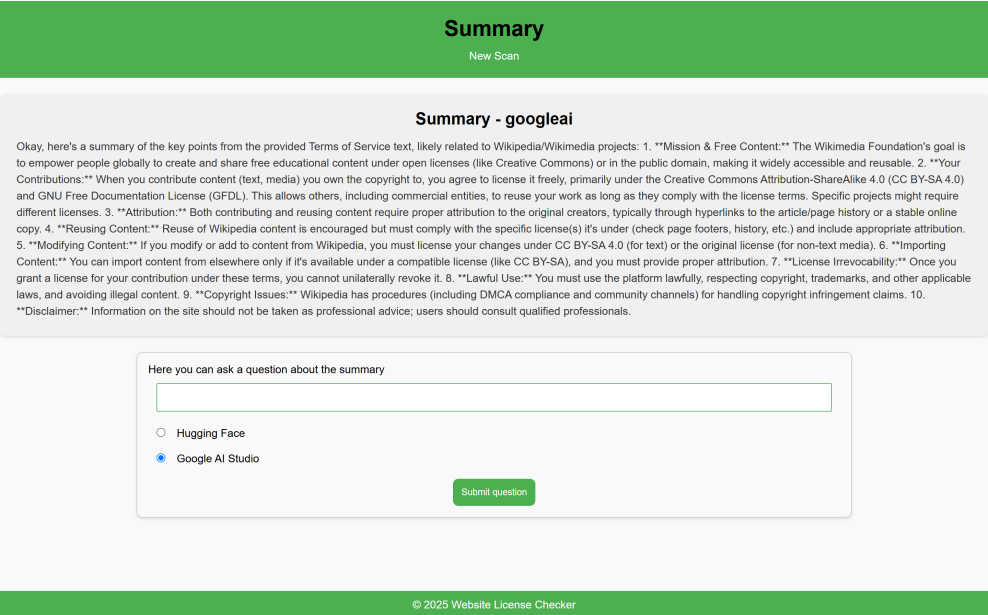
The screenshot shows the 'Website License Checker' interface. It has a green header bar with the title. Below it, a light gray box contains the heading 'Enter websites to scan and API keys for ToS processing'. Inside this box, there is a text input field containing the URL 'https://en.wikipedia.org/wiki/Main\_Page' and an error message 'Invalid\_url.com' below it. Below the input field are two text input fields for API keys: 'Hugging Face API Key:' and 'Google AI Studio API Key:'. At the bottom of the box is a green button labeled 'Scan Websites'. A green footer bar at the bottom of the page contains the copyright notice '© 2025 Website License Checker'.

**Obrázek 3.2** Odeslání URL adres a API klíčů

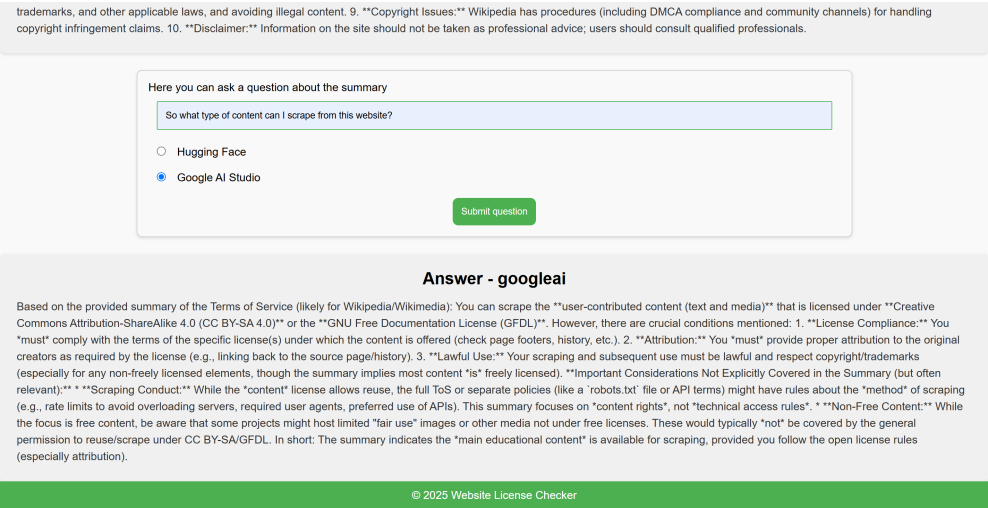


The screenshot shows the 'Scan Results' interface. It has a green header bar with the title. Below it, a light gray box contains the heading 'New Scan'. Inside this box, there is a text input field containing the URL 'https://en.wikipedia.org/wiki/Main\_Page'. Below the input field, there is a section for license information: 'License link found: Visit link', 'License type: CC BY-SA 4.0', and 'Links to ToS information: https://foundation.wikimedia.org/wiki/Special:MyLanguage/Policy/Terms\_of\_Use'. Below this, there is a section for 'Licenses mentioned in ToS: Creative Commons (CC)' and a link to 'Extracted ToS text'. Below the license information is a radio button group with two options: 'Hugging Face' and 'Google AI Studio', with 'Google AI Studio' selected. At the bottom of the box is a green button labeled 'Generate Summary'. Below the box is another text input field containing the URL 'Invalid\_url.com' and an error message 'Given URL is invalid'. A green footer bar at the bottom of the page contains the copyright notice '© 2025 Website License Checker'.

**Obrázek 3.3** Zobrazení výsledků analýzy a možnost summarizovat ToS



■ **Obrázek 3.4** Zobrazení shrnutí ToS a možnost položit otázku



■ **Obrázek 3.5** Zobrazení odpovědi na otázku a možnost položit další otázku

## Testování

*Během vývoje byly jednotlivé části knihovny i webové aplikace průběžně manuálně testovány, a to jak prostřednictvím uživatelského rozhraní, tak voláním navrženého API. Po dokončení implementace však byla vytvořena i sada automatizovaných testů pomocí nástroje pytest. Tyto testy ověřují klíčové funkce první části knihovny, jako je kontrola formátu výstupu, zpracování nevalidních vstupů, funkčnost nastavení vlastního UA a respektování pravidel robots.txt. Avšak hlavní důraz je kladen na testování přesnosti detekce licencí na různých webových stránkách.*

*Druhá část testování se zaměřuje na hodnocení vyzkoušených open source jazykových modelů. K porovnání byly využity standardní metriky jako ROUGE, BERTScore a Faithfulness.*

### 4.1 Testování detekce

Během vývoje byly manuálně testovány vybrané webové stránky obsahující licenční informace, jako například *Wikipedia* [31], *Wikiwand* [32] a *MDN Web Docs* [33], aby byla ověřena schopnost knihovny rozpoznat licence v reálných podmínkách.

Pro automatické testy byly během analýzy a studia tématu sbírány další podobné relevantní stránky. Avšak z důvodu omezeného počtu nalezených stránek bylo přistoupeno k cílenému hledání. Nejprve byl získán seznam 1000 nejnavštěvovanějších webových stránek [7], z nichž bylo náhodně vybráno 100 vhodných pro manuální analýzu. Vhodnými stránkami se rozumí takové, které obsahují klasickou DOM strukturou. Knihovna totiž nevyužívá headless prohlížeč pro vykonání JavaScriptu.

Celkem bylo identifikováno 12 stránek s explicitními odkazy na CC licence v patičce a 21 dalších stránek obsahujících zmínky o sledovaných licencích v obsahu ToS. Avšak tyto nalezené stránky obsahovaly převážně zmínky pouze o CC licencích, což znemožnilo hlubší testování detekce dalších typů licencí. Bylo tak rozhodnuto o rozšíření testovací sady a to vygenerováním licenč-

ních textů obsahující požadované zmínky licencí pomocí jazykového modelu *Gemini-2.5-Pro*. Bylo vytvořeno 50 krátkých textů [34], po 10 pro každý typ licence (MIT, Apache, CC, GPL, BSD), které simulovaly různé formáty licenčních informací. Dále bylo vygenerováno 50 textů [35] obsahujících věty, které svým obsahem mohou vyvolat detekci licence, ačkoli o licenci nehovoří, aby byla otestována odolnost nástroje vůči falešně pozitivním detekcím.

Poslední část testování detekce se zaměřila na identifikaci konkrétního typu CC licence. Jako testovací vzorek byly vybrány stránky, na kterých licence byla dříve korektně detekována, a to jak v patičce, tak v obsahu ToS. Výsledky testování jsou shrnuty v tabulce 4.1:

Kategorie	Počet vstupů	Přesnost
Detekce CC v patičce	12	91 %
Detekce licencí v ToS	21	95 %
Vygenerované texty	100	99 %
Identifikace typu CC licence	31	77 %

■ **Tabulka 4.1** Přehled výsledků testování detekce licencí

Výsledky ukazují více než 90% úspěšnost detekce, čímž nástroj splňuje nefunkční požadavek na přesnost 2.1.2. Je ale důležité zmínit, že testování probíhalo na relativně malém vzorku stránek, z něhož část byla využívána při samotném vývoji nástroje. Navíc byly vybírány pouze takové stránky, které obsahují standardní DOM strukturu.

Ačkoli z testování nevzešel velký počet negativních výsledků, stále bylo možné identifikovat několik omezení detekce. V případech, kdy patička stránky obsahovala více licencí, nástroj detekoval pouze jednu licenci. Další problém nastal u vygenerovaných textů obsahujících samostatný identifikátor, jako například „MIT“. Nástroj v těchto případech licenci nedetekoval, protože vyžaduje více kontextu, aby se vyhnul nechtěné detekci zkratk, které mohou mít jiný význam. Podobně v ToS textech obsahujících výčty licencí, například „...MIT, Apache or public domain licenses...“, nástroj licence MIT ani Apache nerozpoznal. Tato opatrnost v detekci je záměrná, aby se minimalizovaly falešně pozitivní výsledky, což potvrzují testy 50 vygenerovaných textů, které detekci nevyvolaly.

Identifikace konkrétního typu CC licence vykazala nejnižší přesnost 77 %, přičemž chyby nastaly opět při výskytu více licencí na stránce nebo při méně standardní reprezentaci. Výskyt více druhů licencí se zejména projevil v textech ToS, kde specifikoval licenční informace ke konkrétnímu druhu obsahu. V těchto případech nástroj sice určil správný typ licence, ale pouze první detekované, což v testech bylo považováno za chybu.

Při testování byla kromě přesnosti detekce měřena také průměrná doba potřebná k analýze jedné stránky. Naměřené hodnoty byly v průměru  $3,5 \pm 0,3$  s, což splňuje nefunkční požadavek na výkon 2.1.2.

Testování prokázalo, že nástroj dosahuje vysoké přesnosti detekce licencí (>90 %) a plní nefunkční požadavky na přesnost a výkon. Zároveň vykazuje značnou odolnost vůči falešně pozitivním detekcím, jak ukázaly testy 50 vygenerovaných textů, které detekci nevyvolaly. Ovšem zjištěné vlastnosti detekce, jako rozpoznávání typu pouze jedné CC licence na stránce a nižší přesnost zpracování nejednoznačných zmínek, ukazují směr k budoucímu vylepšení.

## 4.2 Hodnocení modelů

Druhá část knihovny, zaměřená na sumarizaci a odpovídání na otázky, generuje výstupy pomocí jazykových modelů. Vzhledem k povaze těchto modelů, které produkují textové výstupy závislé na trénovacích datech a kontextu, není klasické testování, jako například jednotkové testy, vhodné pro ověření jejich kvality. Namísto toho je třeba hodnotit sémantickou přesnost, věrnost a srozumitelnost generovaných textů, což vyžaduje lidské vyhodnocení a použití specializovaných metrik.

### 4.2.1 Výběr modelů

Hlavním kritériem pro výběr byl přístup k modelům bez dodatečných nákladů. Z analýzy dostupných možností splňujících toto kritérium tak vzešli open source modely dostupné na platformách Hugging Face či Ollama [36] a Gemini modely od společnosti Google, poskytující bezplatnou verzi API přístupu. Z Gemini modelů byl zvolen *Gemini-2.5-Pro*, který v době psaní práce dosahuje nejlepších výsledků nejen mezi ostatními modely z řady Gemini. Avšak bezplatný přístup k velkým LLM jako Gemini není standardem a zároveň tyto modely neumožňují případné dolaďování (fine-tuning). A tak v rámci práce byla testována vhodnost open source modelů jako možné alternativy.

Pro testování byly zvoleny modely z kategorie „7B“, tedy modely s přibližně 7 miliardami parametrů. Z lokálního testování na AMD Ryzen™ 7 7840HS vyšel tento druh modelů jako ideální kompromis mezi výpočetní náročností a výkonem – a to i bez zapojení výkonného GPU. Mezi modely, které byly vyzkoušeny patří: Mistral, LLaMA 2, Qwen, BERT a Falcon.

Kvůli časové náročnosti testování byly hodnoceny pouze výsledky summarizace, zatímco úloha odpovídání na otázky nebyla zahrnuta. Pro výběr nejvhodnějších modelů bylo provedeno manuální hodnocení, během něhož se posuzovala kvalita generovaných shrnutí z hlediska přesnosti, srozumitelnosti a úplnosti. Na základě tohoto hodnocení byl výběr zúžen na modely *Mistral-7B* a *Llama-2-7B*, u kterých nebyly rozpoznány výrazné rozdíly v kvalitě generovaných shrnutí. Aby bylo možné určit, který z těchto dvou modelů je vhodnější pro použití v knihovně, proběhlo podrobnější měření jejich výkonu pomocí metrik ROUGE, BERTScore a Faithfulness.



### 4.2.2 ROUGE

Metrika *ROUGE* (*Recall-Oriented Understudy for Gisting Evaluation*) slouží k automatickému hodnocení kvality generovaných textů, jako jsou sumaryzace nebo překlady, porovnáním překryvu  $n$ -gramů mezi generovaným a referenčním textem. Zahrnuje varianty, jako *ROUGE-N*, která měří shodu unigramů nebo bigramů, a *ROUGE-L*, založenou na nejdelší společné podposloupnosti, čímž hodnotí strukturální podobnost textů. [37]

### 4.2.3 BERTScore

Metrika *BERTScore* hodnotí kvalitu generovaných textů, jako jsou sumaryzace, porovnáním sémantické podobnosti mezi generovaným a referenčním textem pomocí kontextuálních vektorů získaných z modelu *BERT*. Na základě kosinové podobnosti jednotlivých slovních vektorů poskytuje skóre, které lépe zachycuje sémantickou shodu než tradiční  $n$ -gramové metriky. *BERTScore* zahrnuje tři hlavní metriky: precision, která měří, jak dobře generovaný text odpovídá referenčnímu textu, recall, která hodnotí, jak dobře referenční text pokrývá generovaný text, a F1 score, která je harmonickým průměrem těchto dvou metrik [38].

### 4.2.4 Faithfulness

Metrika *Faithfulness* měří, jak fakticky konzistentní je generovaný text, se zdrojovým obsahem, přičemž skóre se pohybuje od 0 do 1 a vyšší hodnota značí lepší konzistenci. Postup zahrnuje identifikaci tvrzení v generovaném textu, ověření, zda je lze odvodit ze zdrojového obsahu, a výpočet skóre jako poměru podpořených tvrzení k jejich celkovému počtu. [39] Tato metrika tak umožňuje posoudit, jak dobře generovaný souhrn zachovává faktickou přesnost a věrnost původnímu obsahu ToS.

Efektivní implementací je metoda *LLM as a Judge*, která využívá jazykový model k automatickému hodnocení pravdivosti tvrzení v textu. Z testování různých metod [40] tato dosáhla nejlepších výsledků na benchmarku *HaluBench*, který slouží k testování schopnosti modelů rozpoznávat faktické nesrovnalosti (tzv. halucinace) v generovaném textu. Konkrétně byl použit model *GPT-4o Mini* se vstupním promptem z open source modelu *Llama-3-Patronus-Lynx-8B-Instruct-v1.1*, který je navržený speciálně pro hodnocení halucinací v textech generovaných LLM. V rámci této práce byl jakožto „Judge“ opět použit jazykový model *Gemini-2.5-Pro*.

### 4.2.5 Výsledky

Hodnocení bylo provedeno na datasetu čtyř extrahovaných ToS textů z webových stránek Wikiwand, GitHub, Common Sense Media a Definitions.net. Da-

taset obsahuje pro každou webovou stránku jedno referenční shrnutí (vytvořeno manuálně) a tři generovaná shrnutí od modelů *Mistral-7B* a *Llama-2-7B*, celkem tedy 12 výstupů na model (dataset je uveden v příloženém repozitáři v podadresáři *evaluation*). Naměřené hodnoty jsou uvedeny v tabulce 4.2. Všechny metriky se pohybují v rozmezí od 0 (žádná shoda nebo konzistence) do 1 (perfektní shoda nebo konzistence):

Metrika	Mistral-7B	Llama-2-7B
ROUGE-1	0.4898	0.3997
ROUGE-L	0.3324	0.2662
BERTScore (F1)	0.8777	0.8625
Faithfulness	0.75	0.50

■ **Tabulka 4.2** Přehled výsledků metrik hodnocení

Model *Mistral-7B* dosáhl vyššího skóre v metrikách ROUGE-1 a BERTScore, což svědčí o lepším překryvu unigramů a vyšší sémantické shodě s referenčními souhrny. V metrice ROUGE-L, která hodnotí strukturální podobnost, byly výsledky obou modelů nižší, přičemž *Mistral-7B* opět převyšoval *Llama-2-7B*. Tyto nižší hodnoty poukazují na omezenou schopnost modelů zachovat strukturu textů typu ToS, což může negativně ovlivnit jejich srozumitelnost. Vyšší skóre Faithfulness u modelu Mistral zároveň naznačuje nižší míru halucinací, což je pro sumarizaci ToS zásadní vzhledem k požadavku na faktickou přesnost.

Na základě měření byl tedy vedle *Gemini-2.5-Pro* vybrán jako reprezentant 7B open source modelů právě *Mistral-7B*. V implementaci je však použit model *Mixtral-8x7B-Instruct-v0.1* z rodiny Mistral modelů, neboť plná verze *Mistral-7B* přesahuje velikost 10 GB, což je maximální limit pro využití bezplatné verze Hugging Face API. Model Mixtral byl zvolen jako nejbližší dostupná alternativa, která tento limit splňuje a zároveň poskytuje srovnatelnou kvalitu výstupu.

## Závěr

Cílem této práce bylo analyzovat metody extrakce informací z webových zdrojů a navrhnout a implementovat nástroj pro identifikaci a analýzu licenčních informací a podmínek použití na webových stránkách. Součástí zadání bylo také vytvoření prototypu webové aplikace, která by demonstrovala funkčnost navrženého nástroje a poskytla uživatelsky přívětivé rozhraní pro interakci s jeho funkcemi.

Práce postupovala od analýzy klíčových aspektů problémů, jako jsou licence a podmínky použití na webu, techniky web scrapingu a využití jazykových modelů, přes návrh architektury nástroje a jeho implementaci až po testování. V rámci analýzy byly identifikovány časté textové vzory a klíčová slova, které se využily při návrhu detekčních mechanismů. Implementace zahrnovala vývoj klíčových funkcionalit, jako je správa požadavků, extrakce dat, identifikace licencí a zpracování textů ToS pomocí jazykových modelů. Testování následně ověřilo přesnost detekce licencí a kvalitu generovaných shrnutí.

Vytvořený nástroj je implementován jako knihovna v jazyce Python, která implementuje klíčové funkcionality. Prototyp webové aplikace následně reprezentuje tyto funkcionality, tedy umožňuje zadávání URL adres stránek, zobrazování výsledků analýzy, generování shrnutí a odpovídání na otázky týkající se ToS.

Během testování knihovny byly identifikovány některé nedostatky, jako například omezená schopnost detekce více licencí na jedné stránce nebo nižší přesnost při zpracování nejednoznačných zmínek o licencích. Navazující práce by se mohly zaměřit na vylepšení těchto aspektů, například rozšířením použitých vzorů pro detekci nebo integrací pokročilejších jazykových modelů, které by byly schopny lépe rozpoznávat licence v různých kontextech.

Ačkoli licenční informace zpravidla nejsou skryty za JavaScriptem, absence využití headless prohlížeče a spoléhání se na standardní strukturu DOM omezuje množství webových stránek, pro které je nástroj efektivně použitelný. Budoucí práce by tak mohly tento nástroj rozšířit o podporu zpracování stránek vykreslovaných pomocí JavaScriptu.

Dalším směrem rozvoje by mohlo být zapojení moderních AI agentů, kteří

představují relativně nedávný pokrok v oblasti umělé inteligence. Tito agenti jsou navrženi tak, aby dokázali autonomně plánovat a vykonávat komplexní úkoly, jako je správa požadavků, extrakce a analýza dat nebo generování výstupů. Jejich integrace by mohla výrazně rozšířit schopnosti nástroje, avšak zároveň s sebou přinést nové výzvy, včetně etických otázek spojených s autonomním rozhodováním a zvýšením nároků na výpočetní prostředky.

# Bibliografie

1. OBAR, Jonathan A.; OELDORF-HIRSCH, Anne. The biggest lie on the Internet: ignoring the privacy policies and terms of service policies of social networking services. *Information, Communication & Society*. 2020, roč. 23, č. 1, s. 128–147. Dostupné z DOI: 10.1080/1369118X.2018.1486870.
2. SPDX. *SPDX License List | Software Package Data Exchange (SPDX)* — *spdx.org* [online]. 2024. Dostupné také z: <https://spdx.org/licenses/>. Citováno 18. 3. 2025.
3. SMITH, Jamie. *What is a software license? 5 Types of Software Licenses* / *Snyk* — *snyk.io* [online]. 2025. Dostupné také z: <https://snyk.io/articles/navigating-software-license/>. Citováno 21. 3. 2025.
4. PITTSBURGH LIBRARY, University of. *Guides: Using Creative Commons and Open Software Licenses: Creative Commons* — *pitt.libguides.com* [online]. 2023. Dostupné také z: <https://pitt.libguides.com/openlicensing/creativecommons>. Citováno 21. 3. 2025.
5. COMMONS, Creative. *About CC Licenses - Creative Commons* — *creativecommons.org* [online]. 2019. Dostupné také z: <https://creativecommons.org/share-your-work/cclicenses/>. Citováno 18. 3. 2025.
6. DAVIS, Ziff. *Definition of terms of service* — *pcmag.com* [online]. 2020. Dostupné také z: <https://www.pcmag.com/encyclopedia/term/terms-of-service>. Citováno 19. 3. 2025.
7. DATAFORSEO. *Top 1000 Websites By Ranking Keywords* — *DataForSEO* — *dataforseo.com* [online]. 2025. Dostupné také z: <https://dataforseo.com/free-seo-stats/top-1000-websites>. [Citováno 28. 1. 2025].
8. CONSORTIUM, World Wide Web. *4.4.9 The footer element 2014; HTML5: Edition for Web Authors* — *w3.org* [online]. 2011. Dostupné také z: <http://www.w3.org/TR/2011/WD-html5-author-20110809/the-footer-element.html>. Citováno 19. 3. 2025.

9. LOTFI, Chaimaa; SRINIVASAN, Swetha; ERTZ, Myriam; LATROUS, Imen. Web Scraping Techniques and Applications: A Literature Review. In: 2021, s. 381–394. ISBN 9789391842086. Dostupné z DOI: 10.52458/978-93-91842-08-6-38.
10. DOCS, MDN Web. *Robots.txt — MDN Web Docs Glossary: Definitions of Web-related terms / MDN — developer.mozilla.org* [online]. 2024. Dostupné také z: <https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt>. Citováno 25. 3. 2025.
11. ISHA KON DURKAR Akanksha Raj, D. Lakshmi. Advanced Applications of Generative AI and Natural Language Processing Models. In: IGI Global, 2023, s. 186–227. ISBN 9798369305041. Dostupné z DOI: 10.4018/979-8-3693-0502-7.ch010.
12. ZUBIAGA, Arkaitz. Natural language processing in the era of large language models. *Frontiers in Artificial Intelligence*. 2023, roč. 6. ISSN 2624-8212. Dostupné z DOI: 10.3389/frai.2023.1350306.
13. C. V. SURESH BABU, P. M. Akshara. Advanced Applications of Generative AI and Natural Language Processing Models. In: IGI Global, 2023, s. 228–248. ISBN 9798369305041. Dostupné z DOI: 10.4018/979-8-3693-0502-7.ch011.
14. LM ARENA. *LM Arena — lmarena.ai* [online]. 2025. Dostupné také z: <https://lmarena.ai/>. Citováno 6. 4. 2025.
15. LOTT, Maxim. *Tracking AI — trackingai.org* [online]. 2025. Dostupné také z: <https://trackingai.org/>. Citováno 6. 4. 2025.
16. INC., FOSSA. *FOSSA - Control Your Software Supply Chain — fossa.com* [online]. 2025. Dostupné také z: <https://fossa.com/>. [Citováno 3. 2. 2025].
17. INC., FOSSA. *FOSSID — fossid.com* [online]. 2025. Dostupné také z: <https://fossid.com/>. [Citováno 3. 2. 2025].
18. BLACK DUCK SOFTWARE, Inc. *Application Security Software (AppSec) / Black Duck — blackduck.com* [online]. 2025. Dostupné také z: <https://www.blackduck.com/>. [Citováno 3. 2. 2025].
19. WORKGROUP, FOSSology. *FOSSology — fossology.org* [online]. 2017. Dostupné také z: <https://www.fossology.org>. [Citováno 1. 2. 2025].
20. ILICH, S. *License-aware web crawling* [online]. 2024. Dostupné také z: <http://essay.utwente.nl/100804/>. Citováno 15. 2. 2025.
21. OPENWEBSEARCH.EU. *Project LAW4OSAI — OpenWebSearch.eu — Promoting Europe's Independence in Web Search — openwebsearch.eu* [online]. 2024. Dostupné také z: <https://openwebsearch.eu/partners/ows-eu-project-law4osai/>. [Citováno 6. 4. 2025].

22. REITZ, Kenneth. *Requests: HTTP for Humans Requests 2.32.3 documentation* — *requests.readthedocs.io* [online]. 2024. Dostupné také z: <https://requests.readthedocs.io/en/stable/>. [Citováno 28. 1. 2025].
23. RICHARDSON, Leonard. *Beautiful Soup 4.13.0 documentation* — *crummy.com* [online]. 2025. Dostupné také z: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>. [Citováno 28. 1. 2025].
24. HUGGING FACE, Inc. *Hub client library* — *huggingface.co* [online]. 2025. Dostupné také z: [https://huggingface.co/docs/huggingface\\_hub/index](https://huggingface.co/docs/huggingface_hub/index). [Citováno 9. 3. 2025].
25. GOOGLE. *Google Gen AI SDK documentation* — *googleapis.github.io* [online]. 2025. Dostupné také z: <https://googleapis.github.io/python-genai/>. [Citováno 5. 5. 2025].
26. PALLETES. *Flask Documentation (3.1.x)* — *flask.palletsprojects.com* [online]. 2024. Dostupné také z: <https://flask.palletsprojects.com/en/stable/>. [Citováno 29. 1. 2025].
27. SOLEM, Ask; CONTRIBUTORS. *Celery - Distributed Task Queue Celery 5.5.2 documentation* — *docs.celeryq.dev* [online]. 2023. Dostupné také z: <https://docs.celeryq.dev/en/stable/>. [Citováno 7. 3. 2025].
28. INC., Redis. *redis-py 6.0.0 documentation* — *redis.readthedocs.io* [online]. 2025. Dostupné také z: <https://redis.readthedocs.io/en/stable/index.html>. [Citováno 7. 3. 2025].
29. PATEL, Anubhav. *Pypi Protego* — *pypi.org* [online]. 2025. Dostupné také z: <https://pypi.org/project/pypi-protego/>. [Citováno 11. 2. 2025].
30. PALLETES. *Jinja Documentation (3.1.x)* — *jinja.palletsprojects.com* [online]. 2025. Dostupné také z: <https://jinja.palletsprojects.com/en/stable/>. [Citováno 29. 1. 2025].
31. FOUNDATION, Wikimedia. *Wikipedia, the free encyclopedia* — *en.wikipedia.org* [online]. 2025. Dostupné také z: <https://en.wikipedia.org/>. [Citováno 19. 1. 2025].
32. FOUNDATION, Wikimedia. *Wikiwand - Wikipedia, and beyond* — *wikiwand.com* [online]. 2025. Dostupné také z: <https://www.wikiwand.com/>. [Citováno 28. 1. 2025].
33. FOUNDATION, Mozilla. *MDN Web Docs* — *developer.mozilla.org* [online]. 2025. Dostupné také z: <https://developer.mozilla.org/>. [Citováno 19. 1. 2025].
34. PETERKA, Tomáš. *[Generate license texts containing mentions of various versions of specified licenses...]* [online]. Google, Gemini, 2025. Dostupné také z: <https://g.co/gemini/share/19bc587ea49b>. [Citováno 28. 4. 2025].

35. PETERKA, Tomáš. *[Generate texts that could trigger false positives during detection of license mentions. . .]* [online]. Google, Gemini, 2025. Dostupné také z: <https://g.co/gemini/share/966d7285b313>. [Citováno 28. 4. 2025].
36. OLLAMA. *Ollama — ollama.com* [online]. 2025. Dostupné také z: <https://ollama.com/>. [Citováno 4. 2. 2025].
37. LIN, Chin-Yew. ROUGE: A Package for Automatic Evaluation of Summaries. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004, s. 74–81. Dostupné také z: <https://aclanthology.org/W04-1013/>.
38. ZHANG, Tianyi; KISHORE, Varsha; WU, Felix; WEINBERGER, Kilian Q.; ARTZI, Yoav. *BERTScore: Evaluating Text Generation with BERT*. 2020. Dostupné z DOI: <https://doi.org/10.48550/arXiv.1904.09675>.
39. RAGAS. *Faithfulness Ragas — docs.ragas.io* [online]. 2025. Dostupné také z: [https://docs.ragas.io/en/stable/concepts/metrics/available\\_metrics/faithfulness/](https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/faithfulness/). [Citováno 7. 3. 2025].
40. CAPITELLA, Donato. *Llm-chronicles-6.7-hallucination-detection-faithfulness* [online]. 2024. Dostupné také z: <https://colab.research.google.com/drive/1UTHHrwHmXxgStR5Q4SlcCChuu50IztLR>. [Citováno 7. 3. 2025].



## Obsah příloh

/	
└ deployment.....	adresář se spustitelnou formou implementace
└ other .....	dodatečné soubory a data
└ src	
└ examples.....	příklady použití knihovny a implementace prototypu
└ license_checker.....	zdrojové kódy implementace knihovny
└ tests.....	testy pro ověření funkčnosti knihovny
└ thesis.....	text práce
└ LaTeX.....	zdrojová forma práce ve formátu L <sup>A</sup> T <sub>E</sub> X
└ thesis.pdf.....	text práce ve formátu PDF
└ README.md.....	popis knihovny a postup spuštění prototypu