Produkt scraper s využitím strojového učení

Oliver Morgan
ČVUT–FIT
morgaoli@fit.cvut.cz

24. prosince 2025

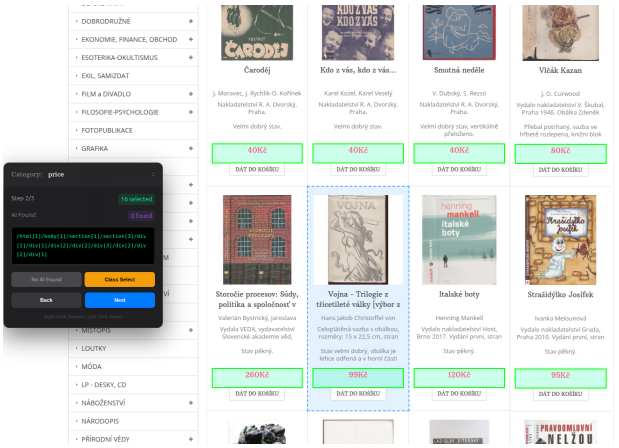
1 Úvod

Extrakce dat z webových stránek je tradičně řešena pomocí pevně definovaných selektorů, které jsou však náchylné k chybám při sebemenší změně struktury webu [1, 3]. Cílem této práce je vytvořit robustnější nástroj **Product Scraper**, který využívá strojové učení k identifikaci klíčových prvků na stránce (název produktu, cena, obrázek) na základě jejich vizuálních a strukturálních vlastností, nikoliv pouze jejich pevné pozice v DOM stromu.

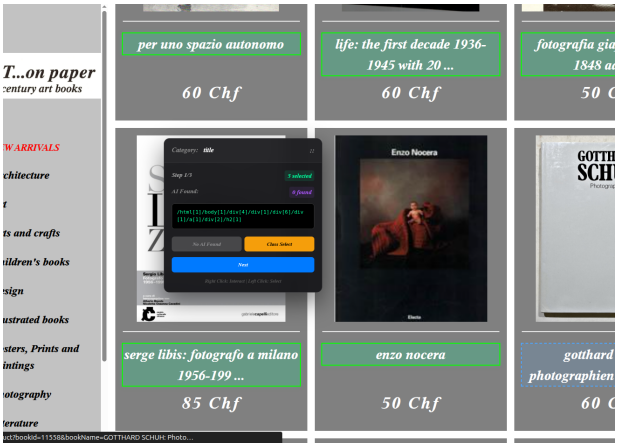
Projekt se zaměřuje na vytvoření "end-to-end"pipeliny: od interaktivního sběru trénovacích dat pomocí vlastního UI, přes extrakci příznaků z HTML, až po trénování klasifikátoru a následné shlukování detekovaných prvků do jednotlivých produktů.

1.1 Interaktivní sběr dat

Pro efektivnější anotaci dat jsem vyvinul interaktivní nástroj postavený na knihovně **Playwright** [4]. Tento nástroj umožňuje uživateli klikáním označovat elementy (např. "toto je cena", "toto je titulek"). Systém automaticky generuje unikátní XPath selektory pro označené elementy a ukládá je do ve formátu YAML . Pro testovaní a ladění jsem pomocí tohoto nástroje označil 52 stránek s kategoriemi: titul, cena a obrazek, jelikož získání kvalitních trénovacích dat je pro tento typ úlohy kritické.



(a) Označení cen



(b) Označení titulů

Obrázek 1: Ukázka interaktivního sběru dat pomocí vyvinutého nástroje.

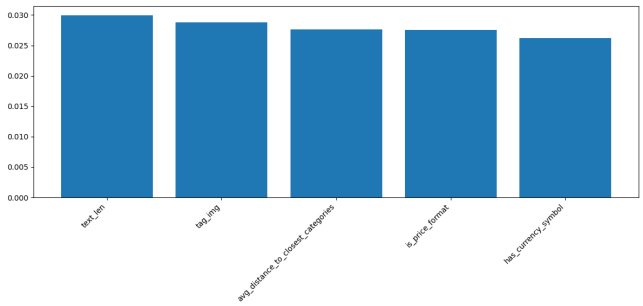
1.2 Extrakce příznaků

V této práci byl zvolen přístup zpracování DOM stromu, kde je každý HTML element převeden na vektor příznaků.

Experimentoval jsem s několika kategoriemi příznaků, abych modelu dodal nejen informace o obsahu, ale i o kontextu:

- **Kontextová vzdálenost (`avg_distance...`):** Tato metrika počítá průměrnou strukturální vzdálenost (DOM tree distance) zkoumaného elementu k nejbližším známým elementům jiných kategorií. Model se tak učí prostorové vztahy – např. že validní cena se obvykle vyskytuje v těsné blízkosti titulku, zatímco náhodné číslo v patičce stránky je od titulku daleko.
- **Vizuální váha (`visual_weight`):** Místo renderování stránky jsem implementoval heuristický výpočet vizuální důležitosti na základě parsování CSS stylů (font-size, font-weight). To umožňuje modelu rozlišit nadpisy od běžného textu bez nutnosti spouštět prohlížeč.
- **Regex detektory:** Pro zpřesnění klasifikace cen a tlačítek byly implementovány regex vzory [2], které detekují přítomnost měnových symbolů a klíčových slov.

- **Hustota sourozenců (Sibling Density):** Metrika, která počítá, kolik elementů se stejným tagem se nachází v okolí. Toto se ukázalo jako klíčové pro detekci mřížek (gridů) produktů, kde se opakují struktury jako `div` nebo `li`.



Obrázek 2: 20 Nejdůležitějších příznaků

Data jsou silně nevyvážená. Na stránce je tisíce `div` elementů, ale jen desítky cen. Proto byl při předzpracování použit *down-sampling* negativních příkladů.

2 Metody a Algoritmy

Jádrem systému je klasifikační model a následný heuristický algoritmus pro seskupování.

2.1 Random Forest Klasifikátor

Jako klasifikační model byl zvolen **Random Forest** [**?**]. Důvody pro tuto volbu:

- Schopnost pracovat s kombinací numerických a kategorických dat.
- Interpretovatelnost (Důležitost Příznaků).
- Odolnost vůči problému dimenzionality.

Model klasifikuje každý HTML element do jedné z kategorií např.: `price`, `title`, `image` nebo `other`.

2.2 Seskupování produktů (Greedy Clustering)

Samotná klasifikace nestačí, protože model nám řekne "toto je cena"a "toto je název", ale neví, že patří k sobě. Pro rekonstrukci produktů byl implementován **Greedy Nearest-Neighbor algoritmus** [**?**]. Algoritmus funguje v následujících krocích:

1. **Určení kotvy:** Vybere se kategorie, která je na stránce nejspolehlivější (typicky ta, která má nejvíce detekcí, např. `title`). Každý nalezený element této kategorie inicializuje nový produkt.
2. **Výpočet vzdáleností:** Pro každou další kategorii (např. `price`) se vypočítá strukturální vzdálenost v DOM stromu ke všem kotvám.
3. **Přiřazování:** Elementy jsou přiřazovány k nejbližšímu produktu. Byla implementována i podpora pro relace 1:N (např. jeden produkt může mít více obrázků v galerii), zatímco jiné atributy (cena) zůstávají 1:1.

3 Výsledky

Implementovaný scraper byl testován na sadě odlišných e-shopů. Aby nedošlo ke zkreslení výsledků, bylo rozdělení dat na trénovací a testovací množinu provedeno na úrovni celých domén. Testovací množina tedy obsahovala e-shopy (11 domén), které model během trénování nikdy neviděl.

Tabulka 1: Výsledky klasifikace na datasetu c

Kategorie	Precision	Recall	F1	Support
image	0.93	1.00	0.96	329
other	0.91	0.95	0.93	972
price	0.98	0.95	0.96	304
title	0.92	0.77	0.84	331
Accuracy			**0.93**	1936

Jak ukazuje tabulka 1, model dosahuje velmi vysoké přesnosti u cen a obrázků. Nižší recall u kategorie `title` (0.77) je pravděpodobně způsoben velkou variabilitou v HTML struktuře názvů produktů, jelikož některé jsou uvnitř `<h3>`, jiné pouze v `<a>` tagu bez výrazného formátování, což je pro model těžší odlišit od běžného textu.

4 Závěr

Vytvořený nástroj *Product Scraper* demonstruje, že kombinace Random Forest klasifikátoru a strukturálního shlukování je efektivní metodou pro extrakci dat z e-shopů.

Hlavní prostor pro zlepšení spočívá v implementaci grafových neuronových sítí (GNN) nebo 2D CRF modelů, které by mohly lépe chápat prostorové vztahy mezi elementy bez nutnosti explicitního heuristického shlukování.

Také bych mohl využit Simple HTML DOM pro odrstranění hluku z dat.

Reference

[1] Miroslav Batchkarov. A benchmark comparison of extraction from html

pages. online, 2018. [cit. 2025–12–24] https://medium.com/@mbatchkarov/a-benchmark-comparison-of-extraction-from-html-pages-98d7c1229f51.

[2] Google Gemini. "generování regex pro ceny" prompt. Gemini, verze z 24. 12., Google, 2025. [cit. 2025–12–24] https://gemini.google.com/.

[3] Matt McDonnell. Dragnet: Content extraction from diverse feature sets. 2013. [cit. 2025–12–24] https://moz.com/devblog/dragnet-content-extraction-from-diverse-feature-sets.

[4] Microsoft. Playwright: Fast and reliable end-to-end testing for modern web apps. online, 2025. [cit. 2025–12–24] https://playwright.dev/.

[5] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.