# Valutazione Strategica e Tecnica di Firecrawl per l'Acquisizione Dati nell'Ecosistema Digitale UNIVPM: Analisi di Idoneità per Pipeline NLP/RAG

## 1. Introduzione: La Sfida dell'Acquisizione Dati in Ambito Accademico

L'implementazione di architetture di Retrieval-Augmented Generation (RAG) in contesti istituzionali complessi, come quello dell'Università Politecnica delle Marche (UNIVPM), impone un ripensamento radicale delle strategie di acquisizione dati. A differenza dei dataset statici o delle API strutturate, il web accademico rappresenta un ecosistema informativo eterogeneo, stratificato e tecnologicamente ibrido. Il presente rapporto fornisce un'analisi tecnica esaustiva sull'idoneità di Firecrawl come motore primario di ingestione dati per il progetto NLP/RAG dell'ateneo, valutandone le capacità di trasformare un dominio web frammentato in una base di conoscenza coerente e semanticamente ricca.

### 1.1 Il Paradigma "LLM-Ready" nel Web Scraping Moderno

Tradizionalmente, il web scraping è stato dominato da un approccio orientato al layout: strumenti come Selenium o Puppeteer sono stati progettati per replicare l'interazione umana o per l'automazione dei test, trattando la pagina web come un insieme di coordinate pixel o un albero DOM (Document Object Model) da navigare visivamente. Tuttavia, l'avvento dei Large Language Models (LLM) ha spostato il baricentro dall'estrazione visiva all'estrazione semantica. In una pipeline RAG, la fedeltà visiva è irrilevante; ciò che conta è la densità informativa e la struttura logica del testo.
Firecrawl si distingue nel panorama tecnologico attuale per la sua filosofia "LLM-first". La sua funzione primaria non è semplicemente scaricare l'HTML, ma "normalizzarlo" in Markdown pulito.[1] Questa trasformazione è cruciale per il progetto UNIVPM. L'HTML grezzo è ricco di "rumore" (tag , classi CSS, script di tracciamento) che diluisce i vettori di embedding, riducendo la precisione del recupero semantico. Il Markdown, al contrario, preserva la gerarchia (titoli, liste, tabelle) in un formato che gli LLM comprendono nativamente,

permettendo strategie di "chunking" (frammentazione del testo) consapevoli della struttura.[3]

### 1.2 Panoramica dell'Ecosistema Digitale UNIVPM

Per valutare correttamente Firecrawl, è necessario deostruire le caratteristiche tecniche del target, il dominio univpm.it. L'analisi dei materiali di ricerca rivela un'infrastruttura caratterizzata da:

- **Gestione Documentale Dinamica:** L'uso diffuso di script server-side come RAServeFile.php [4] per la distribuzione di documenti suggerisce che i file non risiedono sempre su percorsi statici, ma vengono serviti dinamicamente, spesso richiedendo la gestione corretta di sessioni, cookie e redirect HTTP.
- **Profondità Gerarchica e Frammentazione:** Le informazioni critiche, come i dettagli dei bandi di concorso, non si trovano sulla homepage. Sono spesso annidate a 4-5 livelli di profondità, all'interno di strutture paginate (es. .../n0017826469/2) che richiedono capacità di "crawling" ricorsivo intelligente.[5]
- **Dominanza del Formato PDF:** Una porzione sostanziale della conoscenza istituzionale (bandi, graduatorie, regolamenti didattici, decreti rettorali) è incapsulata in file PDF, spesso scansionati o con layout complessi.[4] Un sistema RAG che ignora i PDF sarebbe cieco rispetto alla maggior parte dei processi amministrativi dell'ateneo.
- **Zone ad Accesso Controllato:** Portali come Esse3 (esse3web.univpm.it) gestiscono i dati di carriera degli studenti e richiedono autenticazione, ponendo sfide specifiche di sicurezza e configurazione del crawler.[6]

L'obiettivo di questo documento è determinare se l'architettura di Firecrawl sia in grado di navigare queste complessità in modo più efficace ed economico rispetto alle soluzioni tradizionali o ai concorrenti diretti.

---

# 2. Architettura Tecnica di Firecrawl nel Pipeline RAG

L'integrazione di Firecrawl nel progetto UNIVPM non deve essere vista come l'adozione di un semplice tool, ma come l'implementazione di un layer di astrazione sopra la complessità del web. Firecrawl espone quattro endpoint principali che mappano direttamente le fasi di una pipeline di ingestione dati moderna: Discovery, Acquisition, Extraction e Monitoring.

### 2.1 Endpoint /scrape: Acquisizione di Precisione

L'endpoint /scrape rappresenta lo strumento chirurgico della suite. Accetta un singolo URL e restituisce il contenuto elaborato. Nel contesto UNIVPM, la sua utilità è massima per il monitoraggio di pagine "indice" ad alta frequenza di aggiornamento, come la pagina principale delle "News" o l'elenco dei "Concorsi Attivi".

### 2.1.1 Gestione del Rendering Dinamico (Headless Browser)

A differenza di librerie statiche come requests (Python) o Guzzle (PHP) che scaricano solo il codice sorgente iniziale 8, Firecrawl opera attraverso un cluster di browser headless (basati su Chromium). Questo è fondamentale per univpm.it. Molti portali universitari moderni caricano i contenuti in modo asincrono tramite JavaScript (AJAX) dopo il caricamento iniziale della pagina (Client-Side Rendering).
Un approccio statico vedrebbe solo i contenitori vuoti o gli spinner di caricamento. Firecrawl, invece, attende il completamento del rendering del DOM (Document Object Model), esegue il JavaScript, e solo successivamente converte il risultato in Markdown.2 Questo garantisce che avvisi, scadenze dinamiche o tabelle generate via JS vengano catturati correttamente.

### 2.1.2 Conversione in Markdown Ottimizzato per RAG

La qualità dell'output di /scrape è progettata per massimizzare la "densità semantica".

- **Rimozione del Rumore:** Firecrawl rimuove automaticamente header di navigazione, footer, barre laterali e pubblicità (se presenti), isolando il "Main Content".[9] Per il progetto RAG UNIVPM, questo significa che i vettori di embedding non verranno inquinati da testo ripetitivo come "Copyright 2025 Università Politecnica delle Marche" presente su ogni pagina, migliorando la similarità semantica durante la ricerca.
- **Gestione delle Tabelle:** Le tabelle HTML, frequenti nei bandi per dettagliare i posti disponibili o i requisiti [6], vengono convertite in tabelle Markdown. Sebbene il Markdown abbia limiti nella rappresentazione di celle unite, questa conversione permette agli LLM di interpretare le relazioni riga-colonna molto meglio del testo piatto.

## 2.2 Endpoint /crawl: Discovery Ricorsiva

Per costruire una base di conoscenza iniziale (Knowledge Base Initialization), non è pensabile fornire manualmente migliaia di URL. L'endpoint /crawl automatizza la scoperta dei link.

### 2.2.1 Attraversamento del Grafo UNIVPM

Configurando /crawl sulla root https://www.univpm.it, Firecrawl analizza il DOM alla ricerca di tag <a>, costruisce una coda di frontiera e visita ricorsivamente le pagine.[1]

- **Profondità e Breadth:** Il parametro maxDepth è critico. Una profondità eccessiva (es. >5) rischierebbe di scaricare archivi storici irrilevanti (es. bandi del 2010). Una profondità insufficiente (es. 1) mancherebbe i PDF dei bandi, spesso linkati nelle pagine

di dettaglio.

- **Gestione dei Sottodomini:** L'ecosistema UNIVPM è frammentato in sottodomini (es. dipartimenti, facoltà). L'opzione crawlEntireDomain o allowSubdomains permette al crawler di non fermarsi ai confini di www.univpm.it ma di esplorare anche dii.univpm.it o dicea.univpm.it [11], garantendo una copertura olistica dell'ateneo.

## 2.3 Endpoint /map: Cartografia Strategica

L'endpoint /map è una caratteristica distintiva che separa la fase di "scoperta" da quella di "acquisizione". Restituisce l'elenco degli URL senza scaricarne il contenuto completo.1
Nel contesto del progetto RAG, questo abilita una strategia di "Filtraggio Preventivo". Invece di consumare crediti per scaricare e parsare ogni pagina, si può:

1. Eseguire /map su univpm.it.
2. Ottenere 50.000 URL.
3. Applicare filtri regex locali (es. includere solo URL che contengono /2024/ o /2025/ e parole chiave come bando, regolamento, corso).
4. Inviare solo gli URL filtrati (es. 2.000) all'endpoint /scrape.
   Questa strategia è essenziale per la sostenibilità economica del progetto, dato il modello di pricing basato sui crediti.13

## 2.4 Endpoint /extract: Ingestione Strutturata Agentica

Mentre /scrape restituisce testo non strutturato (Markdown), /extract utilizza LLM integrati per estrarre dati strutturati secondo uno schema JSON definito dall'utente.14
Per UNIVPM, questo trasforma Firecrawl da semplice scraper a motore ETL (Extract, Transform, Load).

- **Applicazione Pratica:** Per la sezione "Bandi", si può definire uno schema Pydantic:
  Python
  ```
  class BandoConcorso(BaseModel):
      titolo: str
      scadenza: str
      dipartimento: str
      numero_posti: int
      link_pdf: str
  ```

  Firecrawl navigherà la pagina, interpreterà il testo naturale ("scade il 15 maggio prossimo") e restituirà un oggetto JSON standardizzato {"scadenza": "2025-05-15"}. Questo permette di creare filtri deterministici nel database RAG (es. "Mostrami solo i bandi non ancora scaduti"), cosa impossibile con il solo embedding vettoriale.[14]

---

# 3. Analisi Approfondita del Dominio Target: UNIVPM

L'idoneità di Firecrawl dipende dalla sua capacità di gestire le specificità tecniche del sito dell'Università Politecnica delle Marche.

## 3.1 La Barriera dei PDF e il Document Parsing

La ricerca evidenzia che i documenti ufficiali UNIVPM sono prevalentemente PDF. Snippet 4 mostra un bando per borse di studio ("Bando.pdf"), mentre 6 mostra bandi per professioni sanitarie.
La maggior parte dei crawler tradizionali si ferma all'URL del PDF. Firecrawl integra un motore di parsing documentale.

- **Meccanismo:** Quando il crawler incontra un header Content-Type: application/pdf, non scarica il file binario come blob, ma lo passa a un pipeline di estrazione testo.
- **Qualità del Testo:** Per PDF nativi digitali (generati da Word/LaTeX), l'estrazione è quasi perfetta. Per PDF scansionati (immagini), Firecrawl utilizza OCR (Optical Character Recognition). Questo è vitale per documenti storici o decreti firmati e scansionati.
- **Parsing Tabellare:** I bandi contengono tabelle complesse (es. ripartizione fondi, scadenze temporali). Firecrawl tenta di convertire queste strutture in Markdown. Sebbene non infallibile su layout a più colonne complessi [16], è nettamente superiore all'estrazione "bag of words" standard.

## 3.2 La Gestione dei Link Dinamici (RAServeFile.php)

Una criticità tecnica emersa è l'uso di script per servire i file:
https://www.univpm.it/Entra/Engine/RAServeFile.php/f/....4
Questo pattern URL nasconde la natura del file (non c'è sempre un'estensione .pdf alla fine dell'URL visibile).

- **Comportamento di Firecrawl:** Grazie all'uso di browser reali, Firecrawl segue il link, attende la risposta del server (che contiene gli header corretti) e gestisce il file in base al *MIME type* effettivo, non all'estensione dell'URL. Un parser statico basato su regex (es. "cerca tutti i link che finiscono in.pdf") fallirebbe miseramente su UNIVPM, mancando gran parte dei documenti. Firecrawl, invece, intercetta correttamente questi asset.

## 3.3 Navigazione e Paginazione

La sezione "Concorsi" presenta una paginazione numerica classica.[5] Firecrawl gestisce la

paginazione nel /crawl in modo automatico se configurato per seguire i link "Next" o i numeri di pagina, trattandoli come normali link <a>. Tuttavia, per un controllo granulare, è spesso preferibile utilizzare l'endpoint /scrape all'interno di un loop controllato dall'applicazione client (es. script Python che incrementa il numero di pagina nell'URL), per evitare di consumare crediti su pagine di archivio troppo vecchie.

---

# 4. Gestione dei Documenti Accademici: Analisi Costi-Benefici

Un aspetto cruciale, spesso sottovalutato, è l'impatto economico e tecnico del parsing PDF su larga scala.

## 4.1 Il Modello di Pricing per i PDF

Firecrawl adotta un modello a crediti. Mentre lo scraping di una pagina HTML costa 1 credito, il parsing di un PDF costa **1 credito per pagina del documento**.[17]

- **Implicazioni per UNIVPM:** Un singolo "Decreto Rettorale" di 50 pagine costerà 50 crediti. Se il sito ospita 10.000 PDF con una media di 10 pagine ciascuno, un crawl completo costerebbe 100.000 crediti. Questo esaurirebbe interamente il piano "Standard" mensile ($83/mese) in una singola operazione.[19]
- **Rischio di "Bill Shock":** Senza un filtraggio aggressivo, il crawler potrebbe scaricare migliaia di verbali, vecchi orari delle lezioni o slide di corsi non pertinenti, facendo esplodere i costi.

## 4.2 Strategie di Mitigazione

Per rendere Firecrawl sostenibile per il progetto RAG, è imperativo non utilizzare la modalità "brute force".

1. **Filtraggio Estensioni/MIME:** Configurare il crawler per ignorare i PDF in fase di discovery iniziale se il budget è stretto, oppure scaricarli solo se l'URL o il testo del link contengono parole chiave come "Bando Attivo" o "Regolamento Vigente".
2. **Preprocessing Locale (Opzione Ibrida):** Usare Firecrawl per scoprire gli URL dei PDF (/map), ma scaricarli e processarli con una libreria locale open-source (come pdfplumber o Nougat per paper accademici) se si dispone di infrastruttura GPU. Tuttavia, questo reintroduce la complessità di gestione infrastrutturale che Firecrawl promette di eliminare. La scelta dipende dal trade-off tra budget Opex (crediti Firecrawl) e budget Capex/Engineering (sviluppo parser locale).

---

# 5. Strategie di Integrazione e Workflow Proposti

Sulla base delle caratteristiche di Firecrawl e UNIVPM, si propongono tre architetture di integrazione.

## 5.1 Strategia A: Ingestione Massiva (Knowledge Base Statica)

Questa strategia mira a creare una copia completa del sito per un chatbot generalista.

- **Workflow:** /crawl su univpm.it con profondità elevata.
- **Pro:** Massima copertura. Il chatbot saprà tutto, dalle news di ieri ai regolamenti del 2015.
- **Contro:** Costi proibitivi (PDF) e alto rischio di "allucinazioni" dovute a dati obsoleti (es. rispondere con le date di un bando scaduto nel 2018 perché semanticamente simile alla query).
- **Idoneità:** Bassa, a meno che non si disponga di budget Enterprise e logiche di post-processing avanzate per filtrare date vecchie.

## 5.2 Strategia B: Ingestione Chirurgica (Targeted RAG)

Questa strategia è raccomandata per casi d'uso specifici (es. "Assistente ai Bandi" o "Orientamento Studenti").

- **Workflow:**
    1. Esecuzione periodica (giornaliera) di /map per rilevare nuovi URL.
    2. Confronto con un database di URL già processati.
    3. Invio dei soli nuovi URL rilevanti (filtrati per pattern) a /scrape o /extract.
- **Pro:** Efficienza dei costi ottimale. Dati sempre freschi.
- **Contro:** Richiede lo sviluppo di una logica di orchestrazione esterna (script Python/Node.js).
- **Idoneità:** Alta. È il miglior compromesso tra qualità del dato e costo.

## 5.3 Strategia C: RAG "Just-in-Time" (Ricerca Live)

Utilizza l'endpoint /search di Firecrawl.[20]

- **Workflow:** L'utente chiede "Quali sono i bandi per Ingegneria?". Il sistema invoca /search con query site:univpm.it bandi ingegneria 2025. Firecrawl esegue una ricerca Google-like, scarica i primi 5 risultati al volo e li passa all'LLM.
- **Pro:** Nessun database vettoriale da mantenere. Informazioni in tempo reale. Nessun costo di storage.

- **Contro:** Latenza più alta (l'utente attende lo scraping). Meno controllo sulla qualità delle risposte (dipende dall'indicizzazione dei motori di ricerca).
- **Idoneità:** Media. Ottima come fallback o per query su eventi correnti, ma meno robusta per risposte complesse basate su documenti lunghi.

---

# 6. Analisi Comparativa: Firecrawl vs Alternative

Per validare la scelta, confrontiamo Firecrawl con le alternative tecnologiche nel contesto specifico UNIVPM.

| Caratteristica | Firecrawl | Apify | Librerie Locali (Scrapy/Selenium) | LlamaParse |
|---|---|---|---|---|
| **Handling RAServeFile.php** | Eccellente (Headless nativo) | Ottimo (Configurabile) | Complesso (Richiede gestione manuale header/cookie) | N/A (Solo parsing, no crawling) |
| **Output per RAG** | Markdown nativo e pulito | JSON/HTML (Richiede post-processing) | HTML Grezzo (Richiede heavy cleaning) | Markdown Ottimizzato per documenti complessi |
| **Costo PDF** | Alto (1 credito/pagina) | Variabile (Tempo di esecuzione) | Basso (Solo costo infrastruttura) | Alto (Modello a pagine) |
| **Manutenzione** | Nulla (SaaS) | Bassa (Platform) | Alta (Gestione rotazione IP, User-Agents) | Nulla (SaaS) |
| **Estrazione Tabelle** | Buona (Markdown) | Dipende dall'Actor scelto | Scarsa (Richiede logica custom) | Eccellente (Specializzato) |

**Verdetto Comparativo:**

- Rispetto a **Apify**: Firecrawl è più verticale sul caso d'uso RAG. Apify è più flessibile ma richiede più lavoro per ottenere un Markdown pulito pronto per l'embedding.[21]
- Rispetto a **LlamaParse**: LlamaParse è superiore nel parsing di PDF estremamente complessi (es. bilanci finanziari con tabelle annidate), ma manca della componente di crawling. Una combinazione vincente potrebbe essere usare Firecrawl per il web crawling e LlamaParse (tramite API integrata o esterna) per i PDF più ostici, se il budget lo consente.[16]

---

# 7. Deployment: Cloud vs Self-Hosted

Una decisione critica riguarda la modalità di deployment. Firecrawl è open-source, ma con riserve significative.

### 7.1 Limitazioni della Versione Self-Hosted

Sebbene il repository GitHub permetta il self-hosting tramite Docker, i materiali di ricerca indicano chiaramente che la versione open-source manca del "Fire Engine".[14]

- **Fire Engine:** È il componente proprietario che gestisce la rotazione avanzata dei proxy, il bypass dei sistemi anti-bot (come Cloudflare turnstile), e le ottimizzazioni di performance per il rendering headless.
- **Parsing PDF:** La capacità di parsing PDF avanzata e senza configurazione è spesso legata a servizi cloud o microservizi non inclusi nella distribuzione base open-source, che potrebbe limitarsi a un'estrazione testo base senza OCR o ricostruzione layout avanzata.[24]
- **Affidabilità:** Gestire un cluster di browser headless (Chromium) in produzione è notoriamente difficile (memory leaks, zombie processes). La versione Cloud scarica questa complessità operativa.

**Raccomandazione:** Per il progetto UNIVPM, iniziare con la versione **Cloud** è imperativo per validare il POC (Proof of Concept) senza attriti infrastrutturali. Il self-hosting dovrebbe essere considerato solo in una seconda fase, e solo se i costi del Cloud diventano insostenibili e si dispone di un team DevOps capace di gestire l'orchestrazione di browser su Kubernetes.

---

## 8. Considerazioni Etiche, Legali e di Sicurezza

Lo scraping di un sito istituzionale richiede aderenza a protocolli rigorosi.

### 8.1 Rispetto del robots.txt e Rate Limiting

È fondamentale verificare il file https://www.univpm.it/robots.txt. Firecrawl, di default, può essere configurato per rispettare queste direttive. Anche in assenza di divieti espliciti, è etico impostare un pollInterval adeguato (es. 1000-2000ms) per non degradare le prestazioni del server universitario, specialmente durante i periodi di picco (es. scadenze iscrizioni).

### 8.2 Dati Personali (GDPR)

I documenti universitari (graduatorie, verbali) contengono spesso dati personali (nomi studenti, matricole, esiti esami).

- **Rischio:** Inserire questi dati in un RAG (e quindi in un Vector DB e potenzialmente nel

contesto di un LLM pubblico) costituisce una violazione della privacy (GDPR).
- **Mitigazione:** La pipeline deve includere uno step di "Sanitizzazione" *post-Firecrawl* e *pre-Embedding*. Utilizzare librerie di NLP (es. Microsoft Presidio) per rilevare e offuscare nomi, codici fiscali e matricole prima che i dati vengano indicizzati. Firecrawl estrae il dato fedelmente; la responsabilità della pulizia PII (Personally Identifiable Information) ricade sull'implementatore.

---

# 9. Analisi Economica Dettagliata

Sviluppiamo una proiezione dei costi basata sui dati di pricing raccolti.[19]
**Scenario Ipotetico: Monitoraggio Mensile UNIVPM**
- **Pagine HTML scansionate:** 5.000 (News, pagine dipartimentali). Costo: 5.000 crediti.
- **Nuovi PDF scansionati:** 500 documenti/mese.
    - Lunghezza media stimata: 8 pagine.
    - Costo PDF: 500 doc * 8 pag * 1 credito = 4.000 crediti.
- **Totale Mensile Stimato:** ~9.000 crediti.

Piano Consigliato: Il piano "Hobby" ($16/mese) offre 3.000 crediti, che sono insufficienti per questo scenario. È necessario il piano "Standard" ($83/mese) che offre 100.000 crediti. Questo piano offre un ampio margine di manovra (buffer di 91.000 crediti) che permette di:
1. Effettuare un "recrawl" completo mensile per aggiornare i contenuti modificati.
2. Assorbire picchi di attività (es. pubblicazione massiva di bandi a inizio anno accademico).
3. Sperimentare con l'endpoint /extract per dati strutturati (acquistando pacchetti token aggiuntivi se necessario).

**Analisi del Valore:** $83/mese è un costo trascurabile rispetto alle ore uomo necessarie per sviluppare, mantenere e debuggare uno scraper personalizzato basato su Selenium e OCR locale.

---

# 10. Conclusioni e Raccomandazioni Operative

Alla luce dell'analisi tecnica, economica e strategica, si conclude che **Firecrawl è altamente idoneo** per il progetto NLP/RAG dell'Università Politecnica delle Marche, a condizione che venga implementato con una strategia consapevole dei costi sui PDF.

## Punti di Forza Decisivi

1. **Markdown-Native:** Risolve il problema della "garbage-in, garbage-out" tipico dei RAG, fornendo chunk semantici di alta qualità dai documenti amministrativi.
2. **Gestione Ibrida Web/PDF:** Unifica in un'unica API la gestione di pagine HTML

dinamiche (RAServeFile.php) e documenti statici, semplificando drasticamente l'architettura della pipeline.
3. **Resilienza:** La gestione automatica di proxy e rendering dinamico garantisce continuità operativa anche a fronte di aggiornamenti del CMS universitario.

## Roadmap di Implementazione Consigliata

| Fase | Azione | Configurazione Firecrawl |
|---|---|---|
| **1. Discovery** | Mappatura del dominio per identificare le aree di interesse. | Endpoint /map su univpm.it. Output analizzato per filtrare percorsi irrilevanti (calendari, login). |
| **2. Setup Pipeline** | Creazione script di ingestione selettiva. | Utilizzo di /scrape in batch sulle URL filtrate. Esclusione iniziale dei PDF > 20MB o storici (>2 anni). |
| **3. Normalizzazione** | Strutturazione dei Bandi. | Endpoint /extract con schema Pydantic per estrarre scadenze e requisiti dai bandi più recenti. |
| **4. Sicurezza** | Redazione PII. | Integrazione di un layer di anonymization sui dati Markdown prima dell'invio al Vector DB. |

In conclusione, Firecrawl trasforma il sito UNIVPM da una collezione disorganizzata di file e pagine web in una sorgente dati strutturata e "queryable", abbattendo le barriere tecniche che tradizionalmente ostacolano i progetti di AI in ambito accademico. L'investimento nel piano Cloud Standard è ampiamente giustificato dal risparmio in termini di sviluppo e qualità del risultato finale.

### Bibliografia


1. Quickstart | Firecrawl, accesso eseguito il giorno novembre 28, 2025, https://docs.firecrawl.dev/introduction
2. Scrape - FireCrawl API Documentation, accesso eseguito il giorno novembre 28, 2025, https://docs.firecrawl.dev/features/scrape
3. Best Chunking Strategies for RAG in 2025 - Firecrawl, accesso eseguito il giorno novembre 28, 2025, https://www.firecrawl.dev/blog/best-chunking-strategies-rag-2025
4. Bando per l'erogazione del contributo ministeriale per le spese di locazione abitativa sostenute dalle studentesse e dagli stu - UNIVPM, accesso eseguito il

giorno novembre 28, 2025, https://www.univpm.it/Entra/Engine/RAServeFile.php/f/borse_studio/2025/Bando.pdf
5. Ultime notizie dei concorsi - UNIVPM, accesso eseguito il giorno novembre 28, 2025, https://www.univpm.it/Entra/Concorsi
6. CORSI DI LAUREA DELLE PROFESSIONI SANITARIE - UNIVPM, accesso eseguito il giorno novembre 28, 2025, https://www.univpm.it/Entra/Engine/RAServeFile.php/f/BANDO_PROFESSIONI_SANITARIE_2025-26_per_pubblicazione.pdf
7. BANDO DI CONCORSO PER L'AMMISSIONE AI CORSI DI DOTTORATO DI RICERCA XLI CICLO - UNIVPM, accesso eseguito il giorno novembre 28, 2025, https://www.univpm.it/Entra/Engine/RAServeFile.php/f/Allegato_1_%28bando_e_allegati%29.pdf
8. PHP Web Scraping for Beginners: A Step-by-Step Guide - Firecrawl, accesso eseguito il giorno novembre 28, 2025, https://www.firecrawl.dev/blog/php-web-scraping
9. Advanced Scraping Guide - FireCrawl API Documentation, accesso eseguito il giorno novembre 28, 2025, https://docs.firecrawl.dev/v0/advanced-scraping-guide
10. Crawl - Firecrawl API Documentation, accesso eseguito il giorno novembre 28, 2025, https://docs.firecrawl.dev/features/crawl
11. Crawl - Firecrawl Docs, accesso eseguito il giorno novembre 28, 2025, https://docs.firecrawl.dev/api-reference/v1-endpoint/crawl-post
12. Firecrawl - The Web Data API for AI, accesso eseguito il giorno novembre 28, 2025, https://www.firecrawl.dev/
13. An honest look at Firecrawl pricing and features in 2025 - eesel AI, accesso eseguito il giorno novembre 28, 2025, https://www.eesel.ai/blog/firecrawl-pricing
14. Mastering the Extract Endpoint in Firecrawl, accesso eseguito il giorno novembre 28, 2025, https://www.firecrawl.dev/blog/mastering-firecrawl-extract-endpoint
15. Extract - Turn Websites into Structured Data with AI - Firecrawl, accesso eseguito il giorno novembre 28, 2025, https://www.firecrawl.dev/extract
16. 5 Best Document Parsers in 2025 (Tested) - F22 Labs, accesso eseguito il giorno novembre 28, 2025, https://www.f22labs.com/blogs/5-best-document-parsers-in-2025-tested/
17. Document Parsing | Firecrawl, accesso eseguito il giorno novembre 28, 2025, https://docs.firecrawl.dev/features/document-parsing
18. Search - Firecrawl API Documentation, accesso eseguito il giorno novembre 28, 2025, https://docs.firecrawl.dev/features/search
19. The Web Data API for AI - Firecrawl, accesso eseguito il giorno novembre 28, 2025, https://www.firecrawl.dev/pricing
20. Mastering Firecrawl Search Endpoint: Web Search and Data Extraction in One API Call, accesso eseguito il giorno novembre 28, 2025, https://www.firecrawl.dev/blog/mastering-firecrawl-search-endpoint
21. Firecrawl vs. Apify: Web scraping, automation and AI data workflows compared - Data4AI, accesso eseguito il giorno novembre 28, 2025,

https://data4ai.com/blog/tool-comparisons/firecrawl-vs-apify/
22. Parsing PDFs with LlamaParse: a how-to guide - LlamaIndex, accesso eseguito il giorno novembre 28, 2025, https://www.llamaindex.ai/blog/pdf-parsing-llamaparse
23. devflowinc/firecrawl-simple: Stripped down, stable version of firecrawl optimized for self-hosting and ease of contribution. Billing logic and AI features are completely removed. Crawl and convert any website into LLM-ready markdown. - GitHub, accesso eseguito il giorno novembre 28, 2025, https://github.com/devflowinc/firecrawl-simple
24. Firecrawl vs. BeautifulSoup: Which is better for web scraping? - Apify Blog, accesso eseguito il giorno novembre 28, 2025, https://blog.apify.com/firecrawl-vs-beautifulsoup/