# POLITECNICO DI TORINO

**Corso di Laurea**
**in Ingegneria Informatica**

Tesi di Laurea

# Intelligent Web Crawling for Automotive Information


**Relatori**
prof. Fulvio Corno

**Candidato**
Gaetano Epiro

Anno Accademico 2022-2023

# Sommario

All'interno del processo di sviluppo di un prodotto, a partire dalla fase di concept fino alla realizzazione del prototipo, si possono individuare delle fasi comuni. Perché il prodotto sia competitivo all'interno di un mercato globalizzato, una delle fasi all'interno di tale processo riguarda l'analisi dei prodotti della concorrenza considerati leader nel settore di appartenenza per analizzare quali siano i fattori che ne hanno determinato il successo. In ambito automotive l'analisi delle specifiche che caratterizzano i veicoli della concorrenza ha l'obiettivo supplementare di realizzare delle vetture che siano sempre più sicure e che rispettino l'ambiente durante il loro ciclo di vita.

Per effettuare queste analisi uno degli strumenti a supporto degli specialisti è un database con le caratteristiche dei veicoli di interesse che consenta di fare confronti tra features omogenee. Il popolamento di tale database tuttavia risulta una delle fasi più energivore in termini di tempo da parte di ingegneri altamente specializzati: le informazioni di interesse sono spesso liberamente disponibili sul Web ma i processi di ricerca e di estrazione delle caratteristiche dei veicoli sono processi attualmente svolti in maniera manuale. L'obiettivo del Web Crawler è quello di essere un possibile strumento di supporto agli Weight Specialists per l'automazione di tali processi.

# Ringraziamenti

Vorrei dedicare questa parte per ringraziare chi mi è stato vicino durante tutto il percorso.
Alla mia famiglia per il supporto in tutti i momenti di difficoltà e le gioie condivise.
A tutto il team di AMET perché senza di loro tutto questo lavoro non sarebbe stato possibile. Grazie ad Alessia per il suo aiuto e i suoi consigli durante lo sviluppo del Web Crawler. Un ringraziamento particolare va a Diana e a tutto il team di sviluppo di Libra.
A tutti i miei amici e i miei coinquilini che mi hanno sempre supportato e sopportato.

# Indice

# Capitolo 1

# Contesto

A partire dagli anni ’90 del Novecento il mondo ha assistito al fenomeno della globalizzazione, un processo che riguarda le dimensioni sociali, culturali ed economiche di tutte le aree su scala internazionale che ha portato a una sempre maggiore crescita dell’integrazione economica, sociale e culturale, nonché all’unificazione dei mercati. Insieme all’integrazione tra i mercati internazionali e ai movimenti continui di persone, merci e capitali, beni e servizi prodotti in qualsiasi parte del mondo sono a disposizione degli acquirenti: chiunque si affacci sul mercato deve quindi essere in grado di fornire un prodotto che sia competitivo rispetto ai prodotti della concorrenza che fanno parte della stessa categoria o che si inseriscono all’interno dello stesso segmento di mercato, in modo da poter soddisfare le esigenze dei clienti, interpretando i loro bisogni e le loro necessità. Lo sviluppo di un nuovo prodotto da lanciare sul mercato è un processo che si articola in varie fasi del tutto generali e prevede tre passi fondamentali:

- fase di benchmark: questa fase comprende il benchmarking esterno, ovvero il processo all’interno del quale vengono raccolte e analizzate le prassi aziendali o le attività operative delle imprese ritenute leader nel settore con l’obiettivo di individuare il "best in class" e determinare i fattori di successo, sia il benchmarking interno che consente di individuare quali unità operative o aspetti del prodotto richiedono un miglioramento

- fase di assessment: in questa fase viene studiato il ciclo di vita di un prodotto e come i vari aspetti che lo caratterizzano evolvono nel

tempo sulla base di metriche chiave che vanno da KPI base, tra i quali ad esempio il valore medio, a obiettivi più specifici e rilevanti per il prodotto

- fase di definizione del target: in questa fase vengono usati i risultati ottenuti durante la fase di benchmarking in quanto, dopo aver individuato il best in class per un prodotto, si definisce il target per il nuovo prodotto da sviluppare identificando il segmento di mercato all'interno del quale si inserirà, quali saranno i suoi competitor e come dovrà evolvere rispetto alla concorrenza

Ne deriva che per qualsiasi nuovo prodotto in sviluppo, raccogliere informazioni sull'ambiente circostante, sui prodotti della concorrenza e delle loro caratteristiche risulta un aspetto fondamentale. La rete internet oggi collega milioni di persone, pagine web, portali e server in tutto il mondo e rappresenta una risorsa quasi inesauribile di informazioni condivise liberamente disponibili, in continua espansione e in continuo mutamento. Nonostance ciò, spesso accade che le informazioni di interesse non siano immediatamente disponibili e che, data l'immensa quantità di pagine su internet, la ricerca possa diventare un processo estremamente dispendioso in termini di tempo.

Il caso di studio preso in considerazione riguarda l'ambito automotive in particolare: durante il processo di sviluppo del Body-In-White (BIW) del veicolo, la predizione e l'ottimizzazione della struttura e la gestione del peso sono tra gli aspetti più importanti in quanto influiscono direttamente sul consumo di carburante, sulle emissioni di anidride carbonica nell'aria e sulle performance della vettura. In generale sviluppare un veicolo più leggero consente di ottenere maggiori prestazioni in accelerazione e avere una massa ridotta si traduce nella possibilità di raggiungere prestazioni più elevate riducendo i consumi e di conseguenza la quantità di $CO_2$ emessa. Dall'altro lato però l'alleggerimento della scocca del veicolo generalmente causa un aumento delle vibrazioni, la quantità di rumore percepito e le prestazioni del veicolo stesso in fase di crash test.

Il processo di Weight Management definisce le regole operative e gli obiettivi istituzionali durante le varie fasi di sviluppo di un veicolo, a partire dalla fase concettuale fino alla produzione, determinando la distribuzione delle masse e la loro concentrazione. Questo rappresenta un aspetto fondamentale dello sviluppo in quanto la distribuzione del peso influisce direttamente sulle caratteristiche del veicolo tra le quali l'assetto, la manovrabilità e la sicurezza (un baricentro troppo alto potrebbe caricare maggiormente le ruote esterne e causare il ribaltamento del veicolo), così come la trazione, l'accelerazione e la decelerazione (spostare la concentrazione delle masse leggermente verso l'anteriore consente una miglior frenata e riduce il rischio che le ruote anteriori possano perdere aderenza in fase di accelerazione).

Questo processo si articola in varie fasi che coinvolgono il benchmarking, il target setting, il weight traking, weight reduction e risk management.

Il benchmark viene tipicamente effettuato tra la fase concettuale e la fase di definizione del veicolo target. Lo scopo fondamentale del benchmarking è quello di analizzare i veicoli della concorrenza e di raccogliere quante più informazioni possibili da utilizzare nel processo di definizione del target.

Spesso le informazioni che riguardano le caratteristiche di un veicolo utili alla fase di benchmarking, sono liberamente disponibili sul web sia perché messe a disposizione dalle stesse case automobilistiche sia perché raccolte da testate specializzate. Vengono poi collezionate e selezionate in maniera manuale dagli weight specialist. Questo processo di ricerca però richiede una grande quantità di tempo per cui molto spesso il benchmarking viene effettuato su un sottoinsieme molto ristretto dei veicoli della concorrenza. Inoltre gli weight specialist devono spesso scontrarsi con siti web in lingua straniera e con unità di misura associate alle caratteristiche del veicolo diverse da quelle di interesse per cui sono comuni errori di traduzione o di conversione nella popolazione del database interno.

Da qui nasce la necessità di uno strumento che sia in grado di automatizzare il processo di ricerca delle caratteristiche associate ai veicoli della concorrenza, effettui una traduzione affidabile e associ caratteristiche in

diverse lingue sulla stessa caratteristica di interesse e che sia in grado di rilevare l'unità di misura quando disponibile, effettuando eventualmente in maniera automatica la conversione in modo da avere un database popolato con unità di misura omogenee e confrontabili che consentano agilmente di effettuare le analisi.

## 1.1 Libra

Libra [19] è il Product Data Management (PDM) sviluppato da AMET [20] per il design e lo sviluppo di nuovi prodotti complessi e sostenibili, che oltre a fornire uno strumento informatico per la raccolta delle informazioni relative a un prodotto in ambito automotive, fornisce tutte le funzionalità per seguire lo sviluppo di un nuovo veicolo, a partire dall'analisi di benchmark con la possibilità di confrontare le strutture funzionali e le features di veicoli interni e della concorrenza, fino alla fase di omologazione del veicolo stesso guidando l'utente attraverso la fase di tracciamento e analisi dei rischi e di definizione del target.

Lo sviluppo di un veicolo, segue le stesse fasi di sviluppo di un qualsiasi prodotto: anche in questo caso è quindi fondamentale durante la fase di benchmark analizzare i veicoli della concorrenza confrontando dei dati che siano affidabili e omogenei tra loro. Molto spesso le informazioni necessarie per effettuare queste analisi sono liberamente disponibili sul Web, sia perché vengono messe a disposizione direttamente dalle case produttrici della concorrenza sul loro sito, sul configuratore o attraverso brochures illustrative, o perché vengono raccolte da siti Web specializzati, amatoriali o testate giornalistiche del settore. Tuttavia, il processo di ricerca delle caratteristiche del veicolo di interesse e di popolazione del database di Libra per poter effettuare i confronti, risulta essere estremamente dispendioso in termini di tempo: il Web Crawler nasce con l'obiettivo di proporre una possibile alternativa al processo di ricerca, di estrazione e di aggiornamento delle informazioni in maniera automatica.

## 1.2 Obiettivi e funzionalità del Web Crawler

Il Web rappresenta una fonte inesauribile di informazioni: tuttavia, al crescere del numero di pagine Web accessibili, diventa sempre più difficile per gli utenti trovare una fonte di informazioni affidabile e ricercare all'interno dei documenti quali siano le parti rilevanti e dove le informazioni di interesse siano contenute: l'obiettivo del Web Crawler è quello di supportare gli Weight Specialists nella ricerca delle features che caratterizzano i veicoli della concorrenza, analizzando le informazioni provenienti da più sorgenti, creando un database di supporto costantemente aggiornato e che consenta di intercettare le nuove tendenze sul mercato.

Il Web Crawler si pone come strumento di supporto agli utenti nel senso che l'obiettivo fondamentale è quello di automatizzare un processo che richiede altrimenti tempo dell'utente per essere svolto ed evitare possibili errori che potrebbero derivare proprio dal fatto che il processo di ricerca e di copia dei valori è manuale: per questo motivo le funzionalità che lo strumento mette subito a disposizione seguono il processo logico e mentale di un utente durante la ricerca.

Queste funzionalità prevedono la definizione delle caratteristiche che identificano univocamente il veicolo di interesse per la ricerca, la definizione delle features che caratterizzano il veicolo e che si vogliono estrarre all'interno delle pagine e la definizione dell'elenco dei siti Web pubblici che contengono le features del veicolo, basandosi sulle esperienze precedenti in modo da focalizzare la ricerca su un sottoinsieme ristretto, prendendo in considerazione prima le fonti che in precedenza si sono dimostrate affidabili, complete e accurate (queste fonti comprendono ad esempio le testate specializzate e i siti Web delle case automobilistiche) e solo se l'informazione di interesse non è ancora disponibile, si prendono in considerazione le fonti secondarie che potrebbero essere inaccurate e richiedono una verifica (ne sono un esempio i siti di compravendita o le testate giornalistiche).

L'elenco delle features che caratterizzano i veicoli costituisce un insieme che all'interno dell'applicazione viene definito insieme delle "Libra Features" ed è un elenco in lingua inglese che descrive in maniera generale le proprietà di tutti i veicoli presenti sul mercato. La ricerca del veicolo richiesto all'interno del sottoinsieme di fonti alimentanti scelto e della pagina che contiene le informazioni che lo caratterizzano, non è limitato dalla lingua del sito Web né dalle possibili diverse modalità con cui fonti diverse identificano quella che logicamente è la stessa caratteristica: il dizionario di sinonimi definito all'interno dell'applicazione consente di mappare features identificate in maniera differente da fonti diverse sulla stessa Libra Feature.

Le features da estrarre che caratterizzano i veicoli possono essere classificate in due categorie separate: la prima riguarda l'insieme delle features per cui il valore è un campo di testo che la descrive mentre la seconda riguarda quelle features per cui il valore è un valore numerico caratterizzato da una certa unità di misura. Per quanto riguarda le features di tipo testo, naturalmente il loro contenuto dipenderà dalla lingua della fonte da cui vengono estratte: con l'obiettivo di avere un database omogeneo per tutti i veicoli, il Web Crawler ne rileva innanzitutto la lingua, quindi ne effettua la traduzione in lingua inglese.

La fase di benchmarking e di analisi dei veicoli della concorrenza richiede un confronto tra le features che caratterizzano i veicoli selezionati: effettuare un confronto tra due valori numerici naturalmente richiede che i due valori siano confrontabili e che le unità di misura associate siano omogenee. Essendo ciascuna feature numerica associata a una unità di misura, per ogni Libra Feature numerica viene individuata un'unità di misura obiettivo in modo tale che il Web Crawler possa effettuare l'equivalenza nel caso in cui il valore numerico debba essere convertito: la conversione potrebbe essere necessaria perché per la stessa feature vengono usate due unità di misura diverse all'interno del Sistema Internazionale di misura (nel caso in cui ad esempio una feature che indica una lunghezza venga indicata talvolta in centimetri e talvolta in millimetri) o perché il valore è stato estratto da una fonte che non utilizza il Sistema Internazionale (e quindi considerando ancora una feature che indica una

lunghezza, se all'interno dell'applicazione per quella feature l'unità di misura obiettivo sono i millimetri, ma il valore estratto è indicato in pollici, è necessario effettuare la conversione dal Sistema Imperiale al Sistema Internazionale).

# Capitolo 2

# Cos'è un Web Crawler

Perché l'enorme quantità di informazioni pubbliche presenti nella rete sia accessibile e fruibile agevolmente dagli utenti, è necessario uno strumento che sia in grado di indicizzare i contenuti e che li renda disponibili quasi istantaneamente quando vengono richiesti, filtrando ciò che non è pertinente e ordinando i risultati in base alla corrispondenza con i criteri di ricerca.

Quando un utente effettua una ricerca all'interno di un motore di ricerca, il risultato è un elenco di pagine visitabili, ordinate in base alla pertinenza con i criteri: in altre parole la ricerca non avviene direttamente sul Web, ma avviene all'interno del database degli indici del motore di ricerca. Ciò che ne rende possibile il funzionamento è un Web Crawler, ovvero un procedimento algoritmico che periodicamente scandaglia la rete internet per ricercare nuovi contenuti o per aggiornare gli indici, con l'obiettivo di creare una mappa del World Wide Web, in modo che i contenuti pubblici siano indicizzati e immediatamente disponibili quando vengono richiesti.

Lo scopo di un Web Crawler è quindi quello di ricerca delle informazioni sul Web, copiando il contenuto delle pagine che visita in modo che possano essere analizzate in un secondo momento e catalogate per parole chiave ed argomenti. La rete internet è in costante cambiamento ed evoluzione: non essendo possibile conoscere a priori il numero totale di pagine o una lista completa di tutti gli indirizzi, il processo di ricerca parte da un sotto insieme ristretto di siti Web, denominati "Seeds", e prosegue analizzando le pagine per trovare nuovi link che indirizzano verso altre pagine in modo

da riuscire a raggiungere enormi porzioni del Web.

In particolare, un Web Crawler analizza le pagine alla ricerca di collegamenti ipertestuali, ovvero di link che indirizzano verso altre pagine, che dopo essere stati opportunamente validati, vengono aggiunti ad una lista che viene comunemente chiamata "Crawl Frontier" per poter essere ricorsivamente visitati e per registrare eventuali modifiche o aggiornamenti.

La validazione degli hyperlinks riveste una parte fondamentale nel processo in quanto durante la fase di ricerca all'interno delle pagine indicizzate, un Web Crawler deve essere in grado di ordinare i risultati in base alla pertinenza delle parole chiave e in base alla qualità del sito Web visitato, distinguendo tra siti Web di bassa qualità, siti Web di alta qualità e persino di spam (ovvero indirizzi generici non verificati o sconosciuti il cui scopo è generalmente pubblicitario o di diffusione di malware e quindi pagine il cui scopo è quello di diffondere vettori di attacco). Un esempio di validazione degli hyperlinks è rappresentato dal sistema di PageRanking di Google [18]: la formula ideata e sviluppata dai fondatori di Google Larry Page e Sergey Brin, assegna un punteggio di importanza ad una pagina Web in base a quanti collegamenti esterni puntano alla pagina di interesse, insieme ad altre metriche che tra le altre includono la rilevanza dei risultati all'interno della pagina rispetto ai parametri di ricerca, il numero di visite da parte degli utenti e l'usabilità della pagina dal punto di vista dell'utente.

Il risultato ottenuto a livello logico è una vera e propria ragnatela di pagine Web legate le une alle altre attraverso collegamenti ipertestuali (hyperlinks), da qui il motivo per cui queste procedure algoritmiche vengono spesso comunemente denominate Web Crawlers o Spiders.

### 2.0.1 Classificazione dell'informazione

Nella società moderna e globalizzata, l'informazione rappresenta uno dei beni più preziosi all'interno di ogni organizzazione, sia che si tratti di un'azienda privata, di pubblica amministrazione o di università, ed è una risorsa che tende a crescere di dimensioni e che dura nel tempo, superando anche l'obsolescenza tecnologica. Nell'ambito aziendale in particolare, un'organizzazione in ambito profit, quanto una in ambito no profit, produce valore in termini di prodotti o servizi e possiede una enorme mole di dati che comprende informazioni sui processi aziendali interni, storico dei

prodotti venduti e acquistati, anagrafiche dei dipendenti, dei clienti e dei fornitori. Il patrimonio dati rappresenta però solo una parte del sistema informativo: oltre ai dati sono infatti necessarie l'insieme di procedure per l'acquisizione, il trattamento e la produzione delle informazioni, le risorse umane che sovraintendono a tali procedure e l'insieme dei mezzi e degli strumenti per la loro archiviazione ed il loro trattamento.

Oggi la mole di dati generati dalle grandi organizzazioni aziendali è in rapida crescita: l'informazione prodotta tuttavia molto spesso manca di una struttura predefinita o di uno schema architetturale identificabile. Questa tipologia di informazione prende spesso il nome di "dati non strutturati". I dati non strutturati vengono memorizzati ed archiviati nel loro formato nativo, all'interno di strutture dati quali i data lake, e non vengono elaborati finché questi non vengono utilizzati.

Non avendo un formato e un modello definito, la categoria dei dati non strutturati include oggetti multimediali quali immagini, video, audio, ma anche documenti testuali o e-mail. Inoltre le fonti che generano i dati non strutturati possono essere estremamente eterogenee tra loro: possono essere ad esempio estratti da un linguaggio umano attraverso tecniche di Natural Language Processing, acquisiti da un sensore all'interno di un dispositivo IoT o persino estratti dai social media.

Nonostante un impatto molto consistente in termini di spazio di memorizzazione richiesto per la loro archiviazione nell'ordine di grandezza dei Petabyte e competenze specifiche nell'ambito della data science per ottenere dei risultati di business intelligence rilevanti, le fonti dati non strutturate:

- non pongono limiti sulla tipologia del formato nativo del dato: i dati non strutturati vengono memorizzati nel loro formato nativo e ciò consente di ampliare enormemente il bacino di casi d'uso possibili, consentendo ai Data Scientist di adattare il formato dell'informazione solo quando questa è necessaria
- consentono una velocità di raccolta maggiore non essendo limitati da una struttura o uno schema definito a priori

Tuttavia l'estrazione di informazioni utili da tali fonti richiede la capacità di estrarre, manipolare e analizzare i dati tramite opportune e

specifiche tecniche di data mining. Inoltre la validità e l'efficacia di tale processo è fortemente limitata dalla qualità intrinseca del dato stesso e dal fatto che il dato possa non essere analizzato correttamente per via della sua natura non strutturata.

Molto spesso però in ambito aziendale risulta fondamentale garantire un accesso rapido all'informazione quando questa viene richiesta: ai dati non strutturati si contrappongono i "dati strutturati" ovvero informazioni digitali formattate secondo una struttura architetturale ben definita prima di essere memorizzata all'interno dell'archivio. I dati formati sono quindi memorizzati all'interno di strutture dati suddivise in campi definiti con precisione e accessibili puntualmente: ne sono un esempio i databases relazionali all'interno dei quali l'accesso all'informazione avviene attraverso query SQL. La natura organizzata e puntuale dei dati strutturati risulta una caratteristica fondamentale per un'organizzazione aziendale in quanto:

- consente un accesso facilitato all'informazione da parte di un utente che non deve necessariamente avere una comprensione approfondita dei vari tipi di dati e delle possibili relazioni tra loro
- molti più strumenti sono in grado di accedere a un'informazione strutturata di fatto ampliando il bacino di possibilità di utilizzo

Di conseguenza la definizione e realizzazione di un archivio digitale che memorizzi le informazioni, rappresenta una delle prime fasi dello sviluppo di un sistema informativo. Tale processo include la definizione di una struttura opportuna per la memorizzazione che consenta agevolmente di indicizzare i dati in modo che questi siano velocemente accessibili quando questi vengono richiesti, compatibilmente con il loro formato.

Rispetto alla vasta eterogeneità di tipologie di dati che possono essere di proprietà di una organizzazione, un database multimediale consente la raccolta all'interno di un'unica soluzione di oggetti di varie tipologie e differenti tra loro: all'interno di uno stesso database è quindi possibile includere non solo documenti di tipo testuale, ma anche immagini, video, audio e oggetti 3D, di fatto generalizzando l'oggetto di partenza in tipo di dato multimediale che, nella sua forma digitale, è stato prodotto dal campionamento di un segnale analogico.

Quindi se da un lato i dati strutturati, classificati anche come dati quantitativi sono informazioni memorizzate e accessibili seguendo un linguaggio di query strutturato quale SQL, sono facilmente decifrabili e rapidamente manipolabili, dall'altro i dati non strutturati sono tipicamente classificati come dati qualitativi che non posseggono un modello dati predefinito e memorizzati in forma non elaborata. Esistono però delle tipologie di dati che rappresentano un "ponte" tra i dati strutturati e i dati non strutturati in quanto:

- non hanno una struttura dati e un modello architetturale definito a priori, che li rende di fatto più complessi dei dati strutturati ma più facili da memorizzare rispetto ai dati non strutturati

- utilizzano dei marcatori semantici o "metadati" per identificare specifiche caratteristiche all'interno dell'informazione, agevolando la ricerca e la catalogazione rispetto ai dati non strutturati

Le pagine Web rappresentano il ponte tra i dati strutturati e i dati non strutturati in quanto normalmente il contenuto di una pagina generica non è noto a priori e non è possibile individuare uno schema architetturale predefinito, ma è organizzato secondo uno schema di marcatori semantici definito dallo standard HTML5 per cui è possibile organizzare e interpretare le informazioni secondo logiche strutturate.

### 2.0.2 Web Scraping

Prese in considerazione le diverse tipologie di rappresentazione dei dati acquisiti, per via della loro natura i dati strutturati vengono memorizzati all'interno di strutture con schemi architetturali ben definiti anche estremamente diversi tra loro: ne sono un esempio i dati contenuti all'interno di databases relazionali (indicizzati e accessibili attraverso query SQL) rispetto alle informazioni all'interno di un foglio di calcolo separate da un carattere speciale (ad esempio la virgola all'interno di un file CSV). Viceversa i dati non strutturati vengono memorizzati in maniera non elaborata in strutture quali i data lake e l'informazione non è immediatamente accessibile, ma disponibile solo dopo un processo di elaborazione.

La rete contiene un volume enorme di dati che vengono presentati attraverso i siti web in un formato che non è interrogabile direttamente né segue dei pattern ricorrenti, bensì spesso l'informazione richiesta al server e ricevuta in risposta viene ulteriormente elaborata sulla macchina frontend del client. I dati così elaborati vengono quindi inseriti all'interno di una struttura il cui tipo viene dichiarato all'inizio del documento (HTML e XML ne sono i due principali esempi) che viene interpretata dal browser del client per poter presentare le informazioni all'utente finale in maniera strutturata e intuitiva. Infine a questa struttura vengono aggiunte le regole contenute nei fogli di stile, che comprendono tutte le informazioni sul layout della pagina, colori, animazioni e transizioni, con l'obiettivo di rendere la pagina più gradevole agli utenti.

Una pagina web quindi non presenta direttamente all'utente finale le informazioni in formato grezzo, ma vengono rielaborate e inserite all'interno del documento insieme ad altre informazioni utili al browser per la definizione del layout e la loro visualizzazione.

Dal punto di vista dell'utente, la struttura della pagina e il suo stile sono del tutto trasparenti: durante la navigazione un utente ricerca all'interno della pagina le informazioni di interesse, che si tratti di dati, testo, immagini o contenuti multimediali. Il processo di Web Scraping si pone come obiettivo di svolgere in maniera automatizzata le azioni compiute da un utente durante la navigazione per estrarre i dati e memorizzarli in maniera più facilmente accessibile. Le pagine Web moderne sono spesso costruite in maniera diversa tra loro, usando tecnologie diverse e nella maggior parte dei casi il documento non viene mostrato all'utente così come viene ricevuto dal server, ma viene sempre rielaborato e il suo contenuto cambia dinamicamente. Il lavoro di uno Web Scraper è quello di analizzare la struttura della pagina per individuarne gli elementi che la costituiscono ed estrarre le informazioni contenute all'interno di questi elementi, nella stessa maniera in cui verrebbero lette da un utente. In altre parole il processo di Web Scraping è un processo che a partire da una fonte dati semi-strutturata, quale una pagina Web, estrae e riordina le informazioni per convertirle in dati strutturati, pronti per essere memorizzati all'interno di un database ed essere facilmente interrogati.

# Capitolo 3

# Stack tecnologico

L'obiettivo del Web Crawler è quello di automatizzare il processo di ricerca dei veicoli sul Web e l'estrazione dei dati contenuti all'interno delle pagine visitate, in una soluzione che da un lato sia facilmente accessibile da qualsiasi dispositivo, indipendentemente dal suo sistema operativo e dalla potenza di calcolo disponibile, e che dall'altro lato offra all'utente un'esperienza altamente interattiva. Per queste ragioni la piattaforma è stata sviluppata seguendo un modello di tipo client/server e quindi come un'applicazione web in cui l'utente si connette a un server usando il proprio dispositivo, sia che esso sia un tablet, un pc desktop o un notebook connesso alla rete, mentre il calcolo e quindi la parte più onerosa in termini di prestazioni richieste, viene mantenuta sul server. Pertanto in quanto web application, il Web Crawler non necessita di installazione sul dispositivo degli utenti ma è un servizio accessibile attraverso un comune Web browser, in un contesto in cui:

- il servizio è facile da distribuire e aggiornare in quanto tutte le logiche dell'applicazione risiedono interamente sul server, la pubblicazione quindi coincide con la distribuzione e l'aggiornamento automatico alla nuova versione per tutti gli utenti

- l'accesso è indipendente dall'hardware e dal sistema operativo del dispositivo usato dall'utente per accedere all'applicazione, l'unico requisito è una connessione alla rete internet

- i costi di sviluppo dell'applicazione sono ridotti rispetto allo sviluppo di applicazioni native per i diversi sistemi operativi e le relative versioni che si intende supportare

- è possibile scalare al crescere del numero di utenti orizzontalmente, aumentando il numero di macchine lato server, oppure verticalmente, aumentando la potenza delle macchine già disponibili

Nel contesto di un'architettura distribuita di tipo client/server, i dispositivi degli utenti rappresentano l'insieme dei client che richiedono un servizio o un documento a un server, che può essere fisicamente o logicamente uno (nel secondo caso il server non è unico ma costituito da un insieme di macchine dietro un load balancer, per cui il loro numero è trasparente ai client), comunicando attraverso un protocollo.

L'elemento di intermediazione tra gli utenti e le risorse o i servizi web è rappresentato da una Application Programming Interface (API) sviluppata in maniera conforme ai vincoli e alle linee guida dello stile architetturale "Representational State Transfer" (REST) introdotta da Roy Fielding [21]. La comunicazione tra un client e il core del Web Crawler avviene quindi richiedendo all'API dei servizi di tipo RESTful [22] con una richiesta HTTP attraverso un comune browser.

A tale scopo risulterebbe inefficiente nonché contro produttivo, in termini di costi di sviluppo e di mantenimento dell'applicazione, concentrare tutte le operazioni all'interno di una singola pagina web: per questo motivo l'applicazione è stata sviluppata seguendo il pattern architetturale MVC in cui nonostante il client abbia l'impressione di interagire con una applicazione Web a pagina singola, il codice viene organizzato in maniera logica e suddiviso in tre parti in cui la prima si occupa di fornire i dati (ed è quindi responsabile dei metodi di accesso al database), la seconda è responsabile della costruzione della pagina che viene visualizzata sul browser del client e infine la terza che fa da intermediaria tra le prime due (il suo obiettivo è quindi quello di ricevere le richieste dal client e reagire in maniera opportuna chiamando il metodo per rispondere alla richiesta).

## 3.1 Pattern MVC

Il pattern Model View Controller (MVC) è un modello architetturale concepito nel 1978 da Trygve Reenskaug [1] con l'obiettivo di proporre una soluzione generale a un problema comune: la manipolazione e il controllo di basi di dati sempre più grandi e complesse, fornendo un ponte logico tra il modello mentale dell'utente e il modello digitale in maniera del tutto trasparente. In questo modo l'utente avrà l'illusione di vedere e di manipolare direttamente le informazioni, anche se questi elementi vengono manipolati nello stesso momento da altri utenti in contesti differenti.

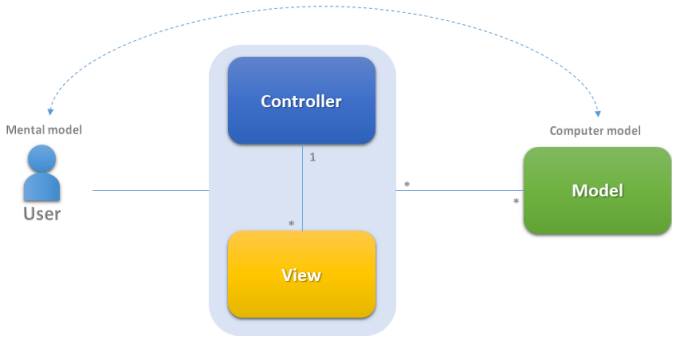


Figura 3.1. Ponte logico tra il modello mentale dell'utente e il modello digitale

Il pattern prevede che l'applicazione consista di un "Model" che rappresenta la parte che contiene le logiche di gestione dei dati, fornendo i metodi per l'accesso e la manipolazione, di una "View" che conosce come accedere al modello dati e come presentarlo all'utente e di un "Controller" che si pone tra il Model e la View, rimanendo in ascolto degli eventi scatenati dall'utente attraverso la View ed eseguendo le operazioni appropriate per quell'evento, generalmente chiamando un metodo del Model. Model, View e Controller non sono solo separati in oggetti differenti, ma anche i loro ruoli sono nettamente distinti: ad esempio il Model conosce come accedere ai dati ma non contiene nessuna logica per presentarli all'utente e viceversa la View sa come accedere ai dati e come presentarli all'utente ma non conosce le operazioni che l'utente può effettuare per manipolarli.

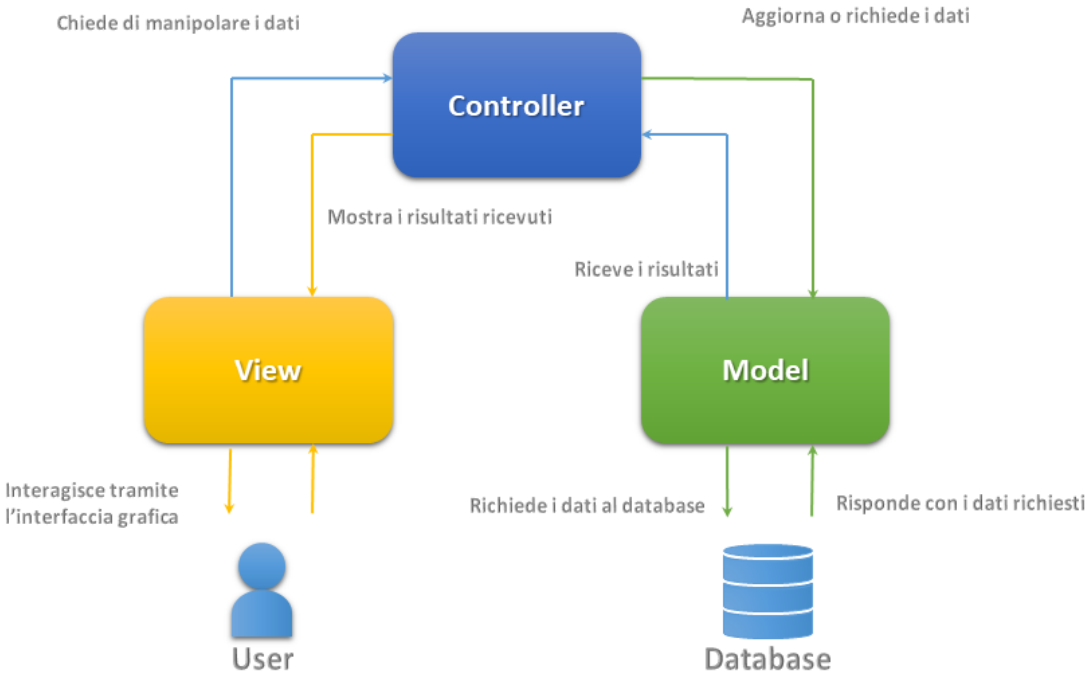


Figura 3.2. Pattern MVC

## 3.2 ASP.NET MVC

ASP.NET MVC è un framework basato sul pattern architetturale MVC per la realizzazione di applicazioni web. Il framework è sviluppato e mantenuto da Microsoft ed è basato sul Core .NET: combina quindi tutte le caratteristiche di ASP.NET e la chiara e netta separazione del codice del pattern MVC.

Tra le funzionalità offerte dal framework di Microsoft, due risultano particolarmente rilevanti nello sviluppo dell'applicazione per le loro caratteristiche di sicurezza e robustezza: Entity Framework e Authentication and Authorization.

Entity Framework è uno strumento Object-Relational Mapping (ORM) per interagire con un SQL Server Database fornendo un'interfaccia orientata agli oggetti per le operazioni di Create, Read, Update and Delete (CRUD), effettuando la conversione del tipo dei dati e favorendo l'integrazione dell'applicazione sviluppata usando un linguaggio di programmazione orientato agli oggetti (C# in questo caso) e un sistema di gestione di basi di dati basato sul modello relazionale (RDBMS) creando di fatto un oggetto di tipo Database Virtuale. In questo modo non solo vengono astratte le caratteristiche implementative dello specifico RDBMS utilizzato, ma si assicura anche la persistenza dei dati (assicura quindi ai dati di non essere legati al ciclo di vita del processo che li ha generati) e un meccanismo di protezione certificato da Microsoft da attacchi di tipo SQL Injection [34].

Authentication and Authorization offre un meccanismo per gestire le richieste al server e gli accessi all'applicazione tramite un processo sicuro in modo che solo gli utenti che sono in grado di risolvere correttamente la sfida di autenticazione possano accedere alle risorse. Inoltre offre un meccanismo sicuro per verificare che l'utente abbia i privilegi minimi per eseguire l'operazione richiesta sulla risorsa secondo il principio del privilegio minimo.

Tra le varie possibilità messe a disposizione dal framework ASP.NET MVC, la strategia scelta per effettuare l'autenticazione degli utenti sfrutta il Token Web JWT [32] definito dallo standard industriale RFC 7519 [33]

```
1         namespace WebCrawler.Utilities
2         {
3             public class TokenUtilities
4             {
5                 public string GenerateToken(User user)
6                 {
7                     byte[] key = new SymmetricSecurityKey(
8                         Encoding.UTF8.GetBytes(_configuration
9                             .GetSection("AppSettings:Token")
10                            .Value)
```

```
11                          );
12
13                      JwtSecurityTokenHandler jwtTokenHandler = new();
14
15                      SecurityTokenDescriptor tokenDescriptor = new()
16                      {
17                          Subject = new ClaimsIdentity(new[]
18                          {
19                              new Claim(JwtRegisteredClaimNames.Name,
                                    user.Username)
20                          }),
21                          Expires = DateTime.Now.AddHours(1),
22                          SigningCredentials = new
                                SigningCredentials(new
                                SymmetricSecurityKey(key),
                                SecurityAlgorithms.HmacSha512)
23                      };
24
25                      return jwtTokenHandler.WriteToken(
26                          jwtTokenHandler.CreateToken(tokenDescriptor)
27                      );
28                  }
29              }
30          }
```

## 3.3 Angular

Angular [12] è una piattaforma di sviluppo open-source modulare, flessibile e scalabile per la creazione di applicazioni web su singola pagina, con l'obiettivo di fornire un'esperienza utente più fluida e dinamica, simile alle applicazioni desktop dei sistemi operativi tradizionali, garantendo un prodotto moderno e di qualità al committente. É stato sviluppato nel 2016 da Google, rilasciato sotto licenza MIT [23] e attualmente mantenuto dal Team Angular di Google e da una community di programmatori indipendenti e corporazioni.

Il progetto rappresenta l'evoluzione di AngularJS [29], popolare framework per lo sviluppo di applicazioni web basate su JavaScript, ma differisce da quest'ultimo per il linguaggio di programmazione utilizzato e per le finalità. Il linguaggio principalmente usato per lo sviluppo è infatti il TypeScript [30], sviluppato da Microsoft con l'obiettivo di estendere la struttura esistente di JavaScript, aggiungendo o rendendo più flessibili e potenti varie sue caratteristiche per garantire maggiore sicurezza e robustezza. Inoltre, a differenza di AngularJS, che non fornisce nessun supporto per lo sviluppo di applicazioni per dispositivi mobili, Angular fornisce la possibilità di sviluppare "Progressive Web App" (PWA) con la caratteristica di essere responsive, ovvero applicazioni in grado di adattarsi automaticamente e dinamicamente alla dimensione del display su cui vengono visualizzate.

Angular adotta il principio di interazione con l'utente di tipo Single-Page Application (SPA), che si contrappone alla modalità di visualizzazione standard nella quale il web browser ricarica interamente le pagine. In una applicazione Web a pagina singola l'interazione avviene attraverso una pagina il cui contenuto viene dinamicamente ridisegnato con l'obiettivo di navigare all'interno dell'applicazione in maniera più fluida e veloce, come se fosse effettivamente un software nativo installato sulla macchina del client. Inoltre, nel contesto di applicazione a pagina singola, il layout delle View del Web Crawler è stato pensato per ridisegnare il numero minimo di componenti in seguito all'interazione con l'utente e allo stesso tempo mantenere consistenza tra le pagine. Per questo motivo le viste seguono il paradigma del Master Page Layout per cui una pagina principale contiene gli elementi comuni a tutte le pagine dell'applicazione, tra cui il menù di navigazione, l'header, il footer e le colonne guida che accolgono i componenti figli.

```
1        export const ROUTES: Routes = [
2            {
3                path: 'Login',
4                pathMatch: 'full',
5                component: LoginComponent
6            },
```

```
7               {
8                   path: 'Home',
9                   pathMatch: 'full',
10                  canActivate: [AuthGuardService],
11                  component: LayoutComponent,
12                  data: {
13                      controller: 'home'
14                  },
15                  children: [
16                      {
17                          component: HomeComponent
18                      }
19                  ]
20              },
21              {
22                  path: 'Seeds',
23                  pathMatch: 'full',
24                  canActivate: [AuthGuardService],
25                  component: LayoutComponent,
26                  data: {
27                      controller: 'seeds'
28                  },
29                  children: [
30                      {
31                          component: SeedsComponent
32                      }
33                  ]
34              },
35              ...
36              {
37                  path: '**',
38                  redirectTo: 'Login'
39              }
40          ];
```

Listing 3.1. Master Page layout

In questo modo in base alla route su cui l'utente sta navigando, viene sempre renderizzato il componente principale LayoutComponent che all'interno contiene gli elementi comuni a tutte le pagine e un corpo per accogliere i componenti figli. Prima di attivare la route e consentire all'utente la navigazione, il servizio di AuthGuard verifica che effettivamente l'utente sia autenticato e che abbia i permessi per accedere a un determinato modulo. Il componente figlio che viene richiamato dipende dalla route ed eredita dal componente padre.

# Capitolo 4

# Analisi del processo

Le modalità di ricerca e gli algoritmi di estrazione delle features rappresentano le due attività principali che il Web Crawler si pone come obiettivo di automatizzare. Tuttavia, prima della definizione di tali processi, è essenziale definire le policies e i protocolli propedeutici alle due attività precedenti. Tali processi includono aspetti, tra i quali ad esempio la definizione delle pagine web più utilizzate nella loro ricerca e che per esperienza contengono le informazioni più complete e affidabili, più strettamente legati all'attività di ricerca manuale sul web degli Weight Specialists e quindi alla loro esperienza operativa, aspetti etici nel rispetto del server visitato, e aspetti che riguardano più da vicino il Web Crawler, la sua efficienza e il funzionamento.

## 4.1 Definizione dei Seed

Lo scopo principale di questa prima fase è quello di definire il processo di ricerca attualmente adottato dal team di Weight Management di AMET con l'obiettivo di automatizzare tale processo: in particolare lo scopo è quello di definire il processo di ricerca dei veicoli sul Web e, una volta individuato il veicolo di interesse, di ricerca delle features associate all'interno della pagina. Attualmente la ricerca avviene manualmente e le fonti alimentanti delle features sono di due tipologie:

- fonti con contenuto semi-strutturato: pagine Web dei siti vetrina delle case automobilistiche, siti web specializzati nella raccolta delle

features dei veicoli, testate giornalistiche o siti di compravendita

- fonti con contenuto non strutturato: brochures e schede tecniche messe a disposizione dalle case automobilistiche in formato pdf

Il caso di studio preso in considerazione nella realizzazione del Web Crawler riguarda le strategie e le tecniche per estrarre le informazioni contenute all'interno di pagine Web, quindi la visita in profondità di un sito e la ricerca delle informazioni di interesse all'interno di una fonte semi-strutturata.

La ricerca dei veicoli di interesse avviene su un sotto insieme ristretto di siti Web considerati affidabili e che, per esperienza del team, contengono la maggior parte delle informazioni di interesse. Il Web Crawler non scandaglia la rete internet nella sua interezza e l'elenco completo della pagine pubblicamente disponibili, l'obiettivo è quello di visitare i siti web per cui la probabilità di individuare le informazioni di interesse è alta e per cui l'informazione contenuta all'intero di queste pagine può essere considerata affidabile. A partire dal sotto insieme di fonti alimentanti attualmente utilizzato dal team di Weight Management, è possibile definire l'elenco dei "Seed", ovvero la lista degli URL di partenza da cui iniziare a estrarre le informazioni.

Un URL in questo contesto viene semanticamente definito Seed in quanto non è solo un indirizzo che viene usato per richiedere una pagina Web, bensì rappresenta il punto di partenza della visita e la pagina a partire dalla quale vengono estratti gli hyperlinks per proseguire con la ricerca in profondità.

Per loro natura i siti Web visitati contengono informazioni disponibili pubblicamente ma che non sono sotto il nostro controllo, per cui potrebbero contenere delle informazioni errate (perché il valore associato non corrisponde al valore reale) o inesatte (perché ad esempio nel caso del peso di un veicolo a combustione, non viene specificato se la pesata è stata effettuata con il serbatoio del veicolo pieno). Inoltre due siti Web diversi potrebbero contenere la stessa feature ma a quella caratteristica vengono associati due valori diversi: per questo motivo ai Seed viene assegnato un punteggio sulla base della correttezza delle informazioni ottenute in passato da quella fonte.

Ad ogni sito Web viene assegnato un punteggio che va da un valore minimo di 1 (le informazioni estratte da questa fonte sono poco affidabili e potrebbero essere errate) a un valore massimo di 5 (assegnato alle fonti che per esperienza contengono informazioni affidabili e ai siti Web delle case automobilistiche): in questo modo se due siti Web contengono la stessa feature ma i valori associati sono differenti, il sistema di punteggio consente di discriminare quale dei due valori è il più affidabile.

## 4.2 Il processo di visita

### 4.2.1 Policies

La prima fase nell'estrazione delle features da una pagina Web riguarda la ricerca della fonte che contiene le informazioni di interesse: inizialmente l'unica informazione a disposizione è l'URL del seed che rappresenta il punto di partenza della ricerca, ma la direzione dipende da ciò che il Web Crawler incontra una volta che si allontana dal punto originale.

Un sito web può essere logicamente interpretato come un grafo diretto ciclico, in cui il punto di partenza e le pagine contenenti le features da estrarre rappresentano i nodi di tale grafo: in generale un sito Web ha una struttura a grafo in quanto una pagina potrebbe contenere degli hyperlinks che fanno riferimento a un nodo della struttura visitato in precendenza, generando di fatto un ciclo o degli hyperlinks che fanno riferimento alla stessa pagina creando dei cappi.

Durante la visita è fondamentale che il grafo sia aciclico per evitare di visitare all'infinito le stesse pagine: eliminando i cicli per definizione la struttura dati non sarà più un grafo, bensì un albero diretto di ricerca nel quale logicamente il punto di partenza della ricerca diventa il nodo radice, le pagine che non contengono le informazioni di interesse diventano nodi intermedi, mentre le pagine che contengono le features da estrarre sono il punto di arrivo e rappresentano le foglie di tale albero.

Infine, un ulteriore aspetto da prendere in considerazione riguarda hyperlinks che reindirizzano a siti Web esterni da quello di partenza: questi hyperlinks rappresentano archi verso altri grafi e devono necessariamente essere eliminati.

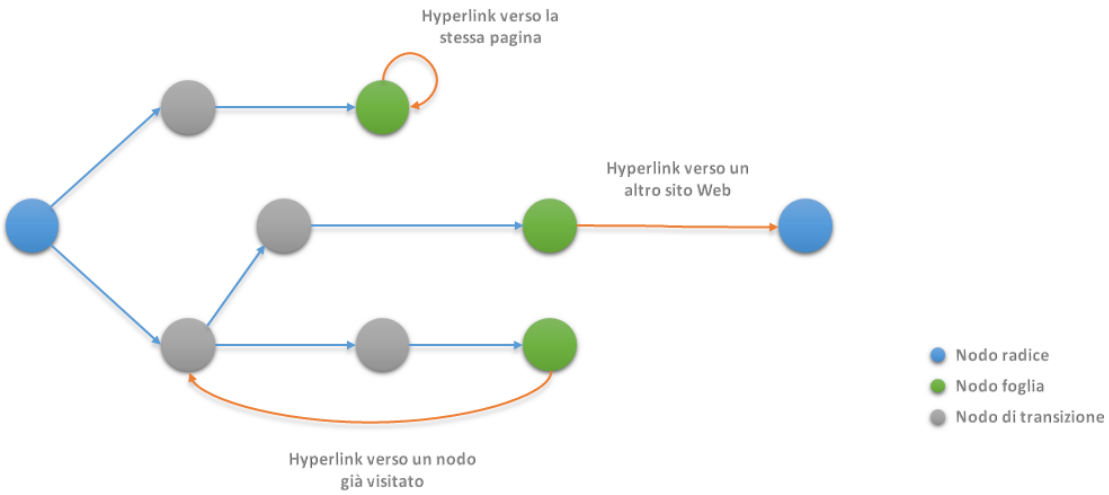


Figura 4.1. Rappresentazione logica di un sito Web generico

Dato lo stesso punto di partenza, gli approcci per la visita sono principalmente due.

Il primo approccio per la visita prevede la ricerca in ampiezza per cui a partire dall'elenco completo degli hyperlinks contenuti all'interno della pagina, vengono visitate prima tutte le pagine al livello sottostante dell'albero prima di procedere più in profondità. La ricerca avviene per livelli: il punto di partenza è il nodo radice dell'albero che contiene gli hyperlinks alle pagine del livello sottostante. Secondo la stessa logica, le pagine di livello 1 contengono gli hyperlinks che puntano alle pagine del livello successivo e così via per tutti i livelli. La ricerca procede un livello alla volta fino ad arrivare alle foglie che non conterranno hyperlinks verso pagine a livelli sottostanti, o conterranno hyperlinks non validi perché puntano a nodi già visitati, a risorse di altro tipo o reindirizzano su altri siti Web.

Il secondo, invece, prevede la visita in profondità per cui a partire da un nodo vengono esauriti tutti gli hyperlinks in profondità prima di procedere con la visita degli altri hyperlinks del nodo allo stesso livello di quello di partenza: in questo modo a partire dal nodo radice dell'albero o a partire da un nodo arbitrario, vengono visitati gli hyperlinks che esplorano il più lontano possibile lungo ogni ramo prima del backtracking.

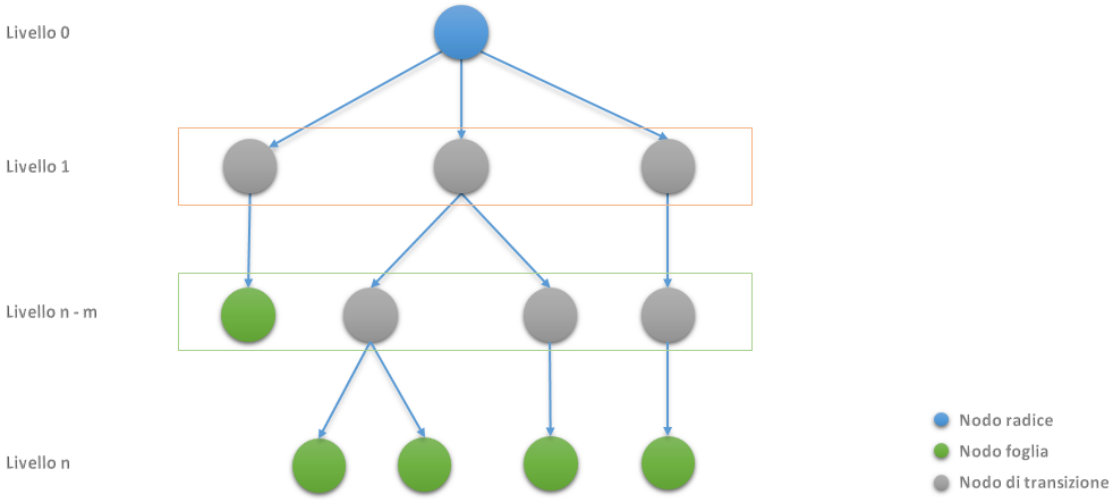


Figura 4.2. Visita in ampiezza

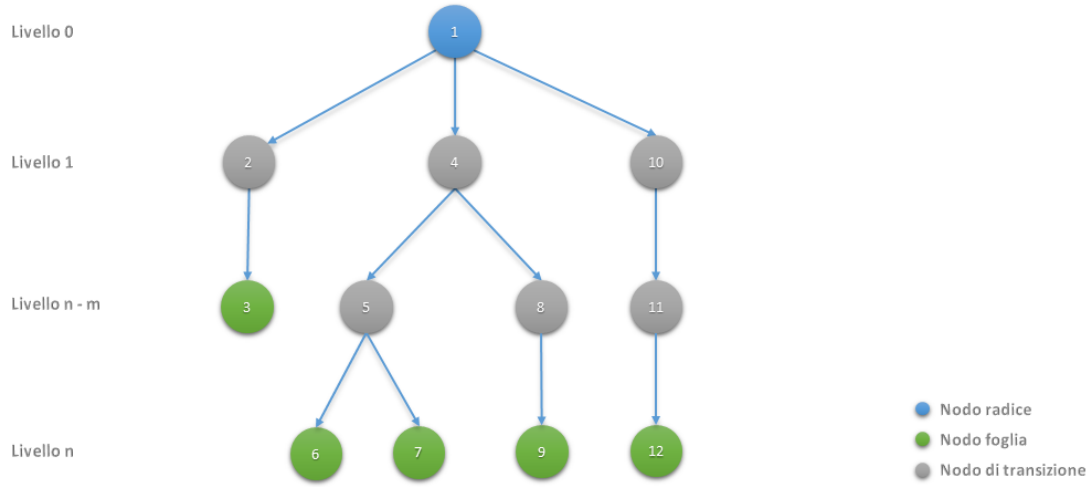


Figura 4.3. Visita in profondità

Tipicamente, per un crawler il cui scopo è quello di indicizzare i contenuti della rete, la scelta di visita ottimale consiste nel visitare il grafo in ampiezza se le pagine sono strettamente interconnesse: se un hyperlink indirizza verso una pagina già visitata allora è probabile che sia già stata visitata recentemente, memorizzata all'interno di una memoria di cache e quindi non è necessario visitarla nuovamente, ottimizzando i tempi di

visita.

In questo caso però l'obiettivo del Web Crawler è quello di effettuare una ricerca mirata della pagina che contiene le informazioni relative al veicolo di interesse: queste informazioni sono contenute in profondità nell'albero, tipicamente sulle foglie. Per questo motivo la policy di visita migliore nel caso di studio preso in considerazione risulta essere la visita in profondità: avendo a disposizione le foglie durante la ricerca, è possibile subito verificare se la foglia contiene le informazioni cercate e quindi interrompere anticipatamente la ricerca. In secondo luogo, considerando che quando viene effettuata una ricerca le pagine che contengono le informazioni più rilevanti e che hanno una migliore corrispondenza con la richiesta vengono presentate per prime, inviare richieste molto accurate e specifiche riduce la probabilità di ricadere nel caso peggiore di ricerca e di dover quindi analizzare tutte le foglie dell'albero.

### 4.2.2 Protocollo di esclusione dei robots

Il Web Crawler vuole essere uno strumento di supporto agli Weight Specialists che automatizzi un processo altrimenti estremamente dispendioso in termini di tempo, senza però violare le restrizioni che potrebbero essere applicate dagli amministratori del Web Server sulla risorsa web visitata.

Il protocollo di esclusione dei robots [31] è uno standard de facto ed è il metodo utilizzato dai gestori di un sito Web per indicare le direttive sull'accesso rispetto a determinate cartelle o risorse. L'insieme delle regole che bloccano o consentono l'accesso a una particolare risorsa all'interno del dominio viene comunicata ai motori di ricerca e ai crawler in generale, attraverso il file di robots: il file di robots è una risorsa testuale in codifica UTF-8 che risiede nella directory principale dell'host del sito a cui si applica e la cui sintassi deve essere conforme al protocollo di esclusione dei robots in modo da poter essere interpretato ed elaborato in maniera automatica.

Lo standard prevede che il file venga organizzato in righe, ciascuna delle quali contiene un record costituito dalla coppia "<chiave> <valore>", separata da ":". Inoltre è previsto che il file di robots contenga obbligatoriamente una o più regole "user-agent" che identificano il gruppo a cui viene applicata quella regola, e obbligatoriamente almeno una o più

voci "Disallow" che indicano la risorsa o la directory a cui non si intende consentire l'accesso.

Dato l'URL di una specifica risorsa, il file di robots viene sempre richiesto prima di accedere a quella risorsa per verificare che l'amministratore del Web Server ne consenta l'accesso, sia nel caso in cui la risorsa viene visitata per la prima volta che nel caso in cui sia stata visitata in precedenza in modo da accertarsi che le policies siano cambiate rispetto all'ultima richiesta.

```
1     /**
2      *  Checks if Web Master allows the access to the resource
3      *
4      * @param   source          The base URL of the Website
5      * @param   resourceUrl     The URL of the resource to get
6      * @output  true            If the resource is accessible
7      * @output  false           Otherwise
8      */
9     private async Task<bool> CanBeCrawled(string source, string
          resourceUrl)
10    {
11        await RequestRobotsFile(source);
12
13        if (System.IO.File.Exists(robotsFile))
14        {
15            string userAgent = GetUserAgent(robotsFile);
16
17            if (string.Equals(userAgent, "*",
                  StringComparison.OrdinalIgnoreCase))
18            {
19                List<string> robotsRules = GetRules(robotsFile);
20
21                foreach (string rule in robotsRules)
22                {
23                    if (Exclude(rule, resourceUrl))
24                    {
25                        return false;
```

```
26                      }
27                  }
28                  return true;
29              }
30          }
31          return false;
32      }
33
34      private async Task RequestRobotsFile(string source)
35      {
36          ...
37          HttpClient httpClient = new();
38          HttpRequestMessage httpRequestMessage = new()
39          {
40              Method = HttpMethod.Get,
41              RequestUri = new Uri(source + "/robots.txt")
42          };
43          HttpResponseMessage httpResponseMessage = await
                httpClient.SendAsync(httpRequestMessage);
44
45          if (httpResponseMessage.IsSuccessStatusCode)
46          {
47              if (null != httpResponseMessage.Content)
48              {
49                  string result = await
                        httpResponseMessage.Content.ReadAsStringAsync();
50                  ...
51              }
52          }
53      }
```

Essendo il file di robots una risorsa statica, è sufficiente richiederla al Web Server tramite una richiesta HTTP GET. Una volta ottenuta, il file viene parsificato e vengono analizzate solo le regole che fanno riferimento a tutti, ovvero tutte le regole per cui il valore della chiave "user-agent"

sia "*" in quanto diversamente indicherebbe un AdsBot (per cui i nomi devono essere indicati espressamente). Vengono quindi analizzate tutte le regole di tipo "Disallow" confrontandole con la risorsa richiesta: se quest'ultima non compare tra le regole, implicitamente l'amministratore consente l'accesso altrimenti la risorsa non viene richiesta.

Tipicamente le risorse protette e non pubblicamente accessibili possono essere richieste e visualizzate dietro una pagina di login e quindi fruibili solo da utenti autenticati e che possiedono i requisiti necessari per il loro accesso: nonostante gli strumenti utilizzati e descritti di seguito siano in grado di completare la procedura di autenticazione nello stesso modo in cui un utente potrebbe autenticarsi tramite un sistema di riconoscimento basato su username e password, se la risorsa è accessibile solo quando è stata risolta la sfida di autenticazione, ragionevolmente l'accesso viene limitato anche nel protocollo di esclusione dei robots, per cui il processo di ricerca viene interrotto e la risorsa non richiesta.

## 4.3 Definizione e gestione degli utenti

Lo scopo principale del Web Crawler è quello di essere uno strumento di supporto agli utenti per la ricerca sul Web delle features che caratterizzano un veicolo di interesse: a tale scopo è necessario definire un utente autorizzato e quali operazioni quell'utente può eseguire sui dati sia che si tratti di visualizzazione, modifica o cancellazione. All'interno dell'applicazione sono state definite quattro aree: un'area riguarda le informazioni legate ai seed, la seconda il dizionario e le associazioni tra le informazioni estratte dalle pagine e il database delle features di interesse, la terza fa riferimento alle features che vengono estratte dalle pagine mentre l'ultima riguarda le operazioni sui veicoli.

Per ogni utente a livello di database e per ognuna di queste quattro aree, è associato un valore intero crescente che indica le operazioni che quell'utente può effettuare all'interno di quell'area: in particolare 0 indica che l'utente non è autorizzato ad accedere ai dati di quell'area, 1 che l'utente può accedere in sola lettura, 2 se l'utente è autorizzato a modificare le informazioni, 3 nel caso in cui le possa anche creare e cancellare e 4 nel

caso in cui l'utente sia un amministratore. Un livello di permesso n implica che l'utente possiede anche i privilegi dei livelli n-m sottostanti per una specifica area, quindi ad esempio un utente con livello di permesso 3 a livello di dizionario, può anche modificare e visualizzare i dati relativi alle features ma se lo stesso utente possiede un livello di permesso 1 per i seed, allora potrà solo visualizzare l'elenco degli URL.

Un utente è identificato tramite username e si autentica tramite una password: per ragioni di sicurezza le password non sono mai memorizzate in chiaro all'interno del database, bensì viene memorizzata una coppia costituita da un sale crittografico e l'hash calcolato utilizzando una funzione di hash sicura, la password dell'utente e il sale. Una funzione di hash è una funzione matematica che a partire da una sequenza di bit, ne genera un'altra che ha le caratteristiche di una stringa casuale (dato un bit, non è possibile prevedere il valore del bit successivo e tutti i bit hanno la stessa probabilità di essere 0 o 1, hanno cioè una densità di probabilità uniforme) e tale per cui non è possibile risalire alla sequenza di partenza. In questo contesto l'hash della password viene calcolato utilizzando il sale crittografico e l'algoritmo SHA-512 [2] sviluppato dalla National Security Agency e pubblicato dal NIST. L'utilizzo del sale crittografico rende del tutto inefficaci gli attacchi basati su hash precalcolati di password note e consente di generare due hash diversi a partire dalla stessa sequenza di bit, per cui per due utenti che scelgono la stessa password verranno generati due hash differenti.

---

```
1     public class SecurityUtilities
2     {
3         public bool VerifyPassword(string password, string
              passwordHash, string passwordSalt)
4         {
5             using HMACSHA512 hmac = new(
6                 Convert.FromBase64String(passwordSalt)
7             );
8
9             byte[] computedHash = hmac.ComputeHash(
10                Encoding.UTF8.GetBytes(password)
11            );
```

```
12
13              return Convert.ToBase64String(computedHash)
14                  .SequenceEqual(passwordHash);
15          }
16      }
```

Listing 4.1. Comparazione dell'hash calcolato con l'hash memorizzato

```
1       public class SecurityUtilities
2       {
3           public void CreatePasswordHash(string password, out string
                passwordHash, out string passwordSalt)
4           {
5               using HMACSHA512 hmac = new();
6
7               passwordSalt = Convert.ToBase64String(hmac.Key);
8               passwordHash = Convert.ToBase64String(
9                   hmac.ComputeHash(Encoding.UTF8.GetBytes(password))
10              );
11          }
12      }
```

Listing 4.2. Generazione dell'hash e del sale crittografico associato

## 4.4 Definizione del dizionario

Una volta definite le fonti alimentanti, il passo successivo consiste nel definire quali sono le informazioni di interesse da estrarre dalle pagine Web prese in considerazione. L'obiettivo principale nella definizione del dizionario è quello di creare un elenco esaustivo di features che caratterizzano un veicolo a partire dalle features che attualmente vengono estratte manualmente dal team di Weight Management. Inoltre, questo elenco deve essere generale, cioè ogni feature deve essere identificata in maniera univoca indipendentemente dalla fonte da cui viene estratta, dalla lingua originale della fonte e dal nome con qui quella determinata caratteristica viene identificata all'interno delle diverse fonti: tali features prendono il

nome di "Libra Features" e la loro definizione è riservata solo agli utenti che hanno un livello di privilegio minimo che autorizzi alla creazione all'interno di quest'area.

Successivamente alla definizione del dizionario e alla definizione della lista delle fonti alimentanti, questa prima fase di inizializzazione prevede che vengano create le associazioni tra le features presenti all'interno delle pagine Web di ciascuna fonte e le features di interesse, ovvero le Libra Features che caratterizzano il dizionario.

Il dizionario tiene traccia del momento in cui viene aggiunta una determinata Libra Feature in modo da poter effettuare delle analisi temporali (ad esempio il momento rispetto al quale una certa feature è diventata di interesse per effettuare il benchmarking o ancora il momento al partire dal quale quella feature è diventata una caratteristica per i veicoli o per una certa categoria di veicoli come nel caso di un nuovo optional che viene lanciato sul mercato).

## 4.5 Definizione dei veicoli

La definizione dei veicoli è uno degli aspetti più importanti del processo in quanto avere molti più vincoli e una descrizione più accurata consente di individuare con maggiore precisione il veicolo corretto e di interrompere anticipatamente la visita ricorsiva in profondità dell'albero di ricerca. Così come avviene per la creazione dei seed e delle Libra Features, la caratterizzazione dei veicoli è un processo riservato solo agli utenti con un livello di permesso che ne consenta la creazione.

Le proprietà che descrivono un veicolo sono delle features loro stesse ma note a priori in quanto rappresentano le caratteristiche che consentono di distinguerlo univocamente: tali features includono delle caratteristiche molto generali e di alto livello tra le quali ad esempio il brand e il modello, e altre più specifiche tra cui la cilindrata, i cavalli e il tipo di alimentazione per discriminare correttamente il veicolo di interesse anche rispetto a veicoli simili della stessa famiglia ma con allestimenti e motorizzazioni diverse. Le features definite dall'utente per riconoscere e identificare un veicolo sono le proprietà che vengono ricercate all'interno di un documento Web per selezionarlo come pagina candidata da essere sottoposta

all'utente, che potrà eventualmente sceglierla come fonte da parsificare nel passo successivo in modo da estrarne le informazioni di interesse.

L'algoritmo effettua l'associazione tra un veicolo definito all'interno del Web Crawler e il contenuto di una pagina web basandosi su una misura di probabilità e in particolare sul numero di proprietà del veicolo che occorrono all'interno del documento. Ciascuna proprietà viene considerata presente se e solo se è presente almeno una volta nel documento, ma ogni occorrenza viene contata una volta: per cui se il brand si ripete più volte all'interno della pagina, l'algoritmo assegna alla feature brand la proprietà "found" ma la presenza del brand all'interno dello stesso documento non viene più verificata.

La correlazione tra il veicolo definito e il contenuto della pagina viene definito su una scala di accettabilità che si basa sul numero di features che caratterizzano il veicolo selezionato e la loro occorrenza all'interno del documento, in cui:

- se il rapporto tra il numero totale delle features che caratterizzano il veicolo e il numero di features presenti all'interno della pagina è inferiore al 50%, la pagina potrebbe contenere delle informazioni generali che potrebbero non appartenere univocamente al veicolo: basti pensare ad esempio a una pagina contenente un elenco di veicoli, ciascuno dei quali caratterizzato da brand e modello, entrambe delle quali sono delle features che definiscono il veicolo, ma che non sono abbastanza per associare con certezza quella pagina. Per questa ragione le pagine che contengono meno della metà delle features che caratterizzano il veicolo vengono scartate e non proposte all'utente come pagine che potrebbero contenere le informazioni di interesse
- se il numero di associazioni è compreso tra il 50% e il 75%, la correlazione può già essere considerata accettabile per cui tale pagina viene restituita all'utente se durante il processo di ricerca non viene trovata una pagina con un livello di associazione migliore o il numero massimo di pagine visitabili viene raggiunto
- se la pagina contiene più del 75% delle features che caratterizzano il veicolo definito, la pagina ha una corrispondenza che può essere

considerata molto buona: in tal caso quindi la ricerca viene interrotta e tale pagina viene presentata all'utente come una possibile pagina da cui poter estrarre le informazioni

La definizione del veicolo rappresenta quindi una parte centrale del processo in quanto definire in maniera precisa, accurata e dettagliata le features che caratterizzano il veicolo da ricercare, consente di ottenere migliori associazioni con le pagine.

# Capitolo 5

# Ricerca delle features e gestione dei veicoli di interesse

Quando un utente che possiede i privilegi minimi per la creazione nell'area delle features si autentica ed accede all'applicazione, ha la possibilità di effettuare due tipologie di ricerca e di estrazione delle informazioni dalle pagine Web:

- la prima modalità consente di estrarre in maniera automatica le informazioni da una singola pagina Web: il processo prevede che l'utente ricerchi manualmente il veicolo di interesse e inserisca all'interno del Web Crawler l'URL della pagina che contiene le informazioni da estrarre
- la seconda rende automatico anche il processo di ricerca dei veicoli: questa modalità prevede che l'utente selezioni un veicolo obiettivo dall'elenco dei veicoli definiti all'interno dell'applicazione, quindi il Web Crawler invia la richiesta per quel veicolo ai siti Web che sono stati definiti come fonti alimentanti, visita in profondità l'albero di ricerca dei risultati a partire dal nodo radice e propone all'utente l'elenco delle pagine da cui è possibile estrarre le features in base alla corrispondenza migliore tra le caratteristiche del veicolo richiesto e quelle presenti all'interno della pagina. Infine l'utente può decidere

da quali fonti estrarre le informazioni, quali siti Web escludere dalla ricerca, visualizzare i risultati e salvare le features estratte o un loro sottoinsieme.

## 5.1 Processo di estrazione da pagina singola

L'estrazione delle features da singola pagina consente agli utenti autorizzati di estrarre le informazioni contenute all'interno di una pagina Web dato il suo indirizzo. Durante sviluppo dell'applicazione la definizione di questo processo ha fissato gli obiettivi da raggiungere per richiedere e scaricare una pagina Web da un sito Web generico e le modalità con cui filtrare le informazioni non rilevanti.

### 5.1.1 Download di una pagina Web

Negli ultimi anni abbiamo assistito a un enorme sviluppo tecnologico e a un netto miglioramento dell'infrastruttura tecnologica: la diffusione dell'ADSL ha rappresentato una vera novità in quanto era una tecnologia che consentiva il trasferimento di bit su un comune doppino telefonico a velocità nell'ordine dei Mbit/s. Ciò voleva dire non dover più limitarsi al trasferimento di pochi bit, ma sfruttando velocità di comunicazione più elevate e nuovi algoritmi per la compressione, potevano essere trasferiti e condivisi anche suoni, immagini e video attraverso una comune rete domestica.

Insieme al miglioramento delle connessioni, un'altra novità nello sviluppo delle tecnologie per il Web è stata lo sviluppo delle tecnologie Ajax per la comunicazione tra più client e un server: in questo modo un client ha la possibilità di inviare una richiesta a un server Web e ricevere in maniera asincrona le informazioni in vari formati senza la necessità di ricaricare la pagina. Presto insieme al miglioramento dell'infrastruttura e delle possibilità di comunicazione è nata la necessità di un'evoluzione degli standard in modo da rispecchiare le nuove necessità: questo ha portato la W3C, la principale organizzazione internazionale per il World Wide Web, a definire e pubblicare nel 2006 le specifiche per le XMLHttpRequests che

consentissero di ricevere dal server informazioni in un formato standard, tipicamente HTML, XML o JSON.

Definiti gli standard di comunicazione tra i client ed il server, la fase successiva ha visto la nascita di standard di supporto per l'organizzazione delle pagine Web insieme alla cura per la grafica in modo da realizzare pagine sempre più accattivanti, visivamente piacevoli e di facile utilizzo per l'utente medio: viene rilasciato nel 2008 HTML5 che definisce il modo in cui strutturare e presentare il contenuto delle pagine Web e il CSS per la definizione di classi che in maniera separata ne definiscono lo stile e il layout.

Gli sviluppatori hanno quindi a disposizione HTML per definire la struttura logica di una pagina Web e CSS per renderla visivamente più piacevole lavorando sui colori, sul posizionamento degli elementi, sui caratteri e sulle forme ma le pagine sono ancora statiche e prive di logiche di interazione e di contenuti dinamici: per questo motivo le pagine Web iniziano ad includere una parte di script, internamente o richiamata da un file esterno, che viene eseguita direttamente all'interno del browser del client. L'introduzione di codice JavaScript o TypeScript permette di definire le logiche di interazione della pagina e di elaborare i dati ricevuti dal server per presentarli all'utente senza la necessità di ricaricare la pagina proprio perché a differenza di logiche scritte usando linguaggi come PHP o ASP che vengono eseguite sul server e poi il contenuto presentato all'utente, questi script vengono elaborati sulla macchina dell'utente garantendo velocità e alleggerendo il server dall'elaborazione dei dati per la presentazione.

In generale le pagine dei siti Web moderni contengono una parte di logica di script eseguita dal browser all'interno della macchina del client che agisce sul contenuto, sulla struttura e sulla presentazione della pagina stessa. Ciò vuol dire che quando un utente invia al server la richiesta per una certa pagina attraverso il browser, il server risponderà con i documenti che contengono il codice HTML e con i fogli di stile che contengono il CSS. A questi verranno ancora aggiunte in risposta eventuali immagini e contenuti multimediali in generale, i file che contengono i sorgenti con il codice di script da eseguire e i dati che il browser dovrà elaborare secondo le logiche degli script per popolare la pagina prima di presentarla

all'utente.

Il processo di elaborazione della pagina Web richiede pertanto risorse multiple: una singola richiesta HTTP non è sufficiente in quanto la risposta ottenuta dal server sarebbe un documento contenente la struttura HTML statica della pagina, priva dei dati e delle informazioni di interesse ottenibili solo dopo l'esecuzione degli script. Richiese HTTP per ottenere esclusivamente i dati in molti casi non rappresentano una soluzione valida in quanto i siti Web potrebbero non esporre servizi o API per rispondere a tali richieste. Supponendo che tutti i siti Web definiti come seed, ovvero definiti come fonti alimentanti valide per il Web Crawler, espongano API per richiedere le features dei veicoli, effettuare richieste HTTP per richiedere queste risorse non rappresenterebbero ancora una strategia valida in quanto non sarebbe generalizzabile: bisognerebbe effettuare richieste specifiche per ogni sito Web e conoscere la struttura dati che descrive logicamente le informazioni ottenute. Inoltre, la soluzione non sarebbe robusta relativamente a possibili aggiornamenti dei siti Web: ad ogni modifica nelle API o nelle strutture dati del sito Web deve necessariamente corrispondere una modifica nell'applicativo del Web Crawler.

La soluzione migliore risulta quindi quella di definire un processo che durante l'elaborazione di una pagina Web sfrutti gli standard definiti da HTML5 in modo tale da essere robusta rispetto a cambiamenti delle strutture dati e delle API dei siti Web visitati e che si adatti agli aggiornamenti e alle modifiche degli elementi all'interno di una pagina.

Per scaricare una pagina Web in maniera automatica all'interno dell'applicazione Web Crawler sono quindi necessarie più richieste HTTP per ottenere tutte le risorse necessarie, un tempo di elaborazione e l'esecuzione degli script che vengono normalmente processati dal browser quando un utente richiede una pagina.

Un browser è un applicativo software che consente ad un utente di richiedere una pagina Web per un determinato sito web attraverso un'interfaccia grafica in modo che i contenuti siano facilmente fruibili da un essere umano. L'interfaccia grafica che include pulsanti, icone e finestre per l'interazione con l'utente rappresenta solo una parte dell'applicativo che non risulta essenziale per la navigazione se a navigare le pagine non è un umano.

Un "headless browser" è un normale browser applicativo in grado di eseguire le stesse operazioni di un browser tradizionale, tra cui interpretare i documenti HTML, gli elementi di stile, eseguire script ed effettuare chiamate Ajax in un contesto privo di interfaccia grafica. L'utilizzo di un headless browser ha numerosi vantaggi di cui alcuni risultano particolarmente rilevanti nello sviluppo del Web Crawler: sono più veloci di un browser tradizionale non dovendo renderizzare la parte da presentare all'utente, possono essere eseguiti e comandati in maniera automatica attraverso un'interfaccia e possono essere utilizzati per automatizzare l'interazione con un sito Web, sia per eseguire test automatici sulle pagine che per eseguire determinate operazioni simulando il comportamento di un utente che sta navigando la pagina.

Chromium [3] è browser libero il cui progetto è open source, sviluppato e mantenuto da Google. Il codebase [4] fornisce la maggior parte delle funzionalità ai browser moderni tra i quali Google Chrome, Edge e Opera e può essere utilizzato come browser con la propria interfaccia grafica come qualsiasi altro applicativo, ma può essere utilizzato anche in modalità headless [5] fornendo tutte le funzionalità di una piattaforma web moderna accessibili da riga di comando [6].

In modalità headless, per definizione, Chromium non può essere controllato sfruttando nessuna interfaccia grafica ma tramite apposite interfacce: Puppeteer [7] [8] è una libreria per Node [9] sviluppata dal team di Chrome che offre una API di alto livello per controllare Chrome sia nella sua versione completa con interfaccia grafica che nella modalità headless. La versione rilasciata e mantenuta da Google segue sempre l'ultima versione Long Term Support (LTS) di Node. Puppeteer non è disponibile solo per JavaScript Runtime Environments, ma la versione PuppeteerSharp [10] [11] consente di accedere alle stesse funzionalità offerte dalla libreria ufficiale di Chrome per applicazioni .NET in C#.

```
1        /**
2         * Calls the Headless Browser to download the Web Page after
              the JavaScript code is executed
3         * @input sourceURL     URL of the page to be downloaded
4         * @input output        Output path for the downloaded page
5         */
```

```
6           private async Task DownloadHTMLAfterJavascript(string
                sourceURL, string output)
7           {
8               // Download Chromium revision if it doesn't already exist
9               await new BrowserFetcher().DownloadAsync(
10                  BrowserFetcher.DefaultChromiumRevision
11              );
12
13              // Create an instance of the browser and configure
                    launch options
14              Browser browser = await Puppeteer.LaunchAsync(new
                    LaunchOptions
15              {
16                  ExecutablePath = crawlerConfig.Get_BROWSER_PATH(),
17                  Headless = true
18              });
19
20              // "Open" a new page
21              Page page = await browser.NewPageAsync();
22
23              page.DefaultTimeout = crawlerConfig.Get_TIMEOUT();
24
25              // Navigate to the selected web page
26              await page.GoToAsync(
27                  sourceURL, WaitUntilNavigation.Networkidle2
28              );
29
30              // Get the page content as string
31              string content = await page.GetContentAsync();
32
33              // Write the page content to the output file
34              utils.WriteToFile(output, content);
35
36              // Destroy unused resources
37              page.Dispose();
38              browser.Dispose();
```

```
39          }
```

Listing 5.1. Richiesta, elaborazione e download di una pagina singola

In questo modo il Web Crawler crea una nuova istanza di un headless browser e apre una nuova connessione all'interno di una nuova pagina verso un URL specifico ricevuto come parametro. Il browser richiede quindi tutte le risorse necessarie per costruire la pagina nella stessa maniera in cui verrebbe elaborata da un browser tradizionale prima di renderla disponibile per la visualizzazione all'utente. Le pagine richieste vengono scaricate per essere visionate, lette e navigate come se fossero state appena richieste ma vengono effettivamente conservate come snapshots, ovvero come istantanee dello stato della pagina in un certo momento. Le pagine scaricate potranno essere poi elaborate per estrarre gli hyperlinks se si tratta di una pagina di transizione nell'albero di visita del sito Web o le features nel caso in cui la pagina sia una foglia. Nel primo caso quando le informazioni sono state estratte, la pagina viene distrutta mentre nel secondo caso la pagina viene effettivamente archiviata tenendo traccia all'interno del database del suo indirizzo sorgente, il riferimento al seed e un timestamp che identificare il momento in cui quella pagina è stata estratta.

### 5.1.2 Estrazione delle informazioni da una pagina Web

Una pagina richiesta a un sito Web dato il suo URL può essere logicamente di due tipologie diverse, ovvero può essere una pagina di transizione (sia che essa sia il nodo radice dell'albero e quindi il punto di partenza della ricerca, sia che sia un nodo di transizione e quindi un nodo intermedio) o una pagina che contiene le informazioni di interesse che si vogliono estrarre per popolare le Libra Features (e quindi il nodo foglia dell'albero in quanto non sarà necessario scendere più in profondità).

A livello logico questa distinzione è necessaria per due motivi: il primo riguarda l'archiviazione delle pagine essendo il mantenimento degli snapshots riservato unicamente alle pagine che contengono le informazioni a cui siamo interessati, mentre il secondo motivo riguarda il processo di elaborazione sulla pagina stessa. Tutte le pagine richieste contengono

informazioni rilevanti nel processo di ricerca ma è necessario elaborarle in maniera diversa in funzione del loro contenuto logico: per le pagine di transizione le informazioni da estrarre sono gli hyperlinks da inserire all'interno della coda di visita che consentono di proseguire con la visita e quindi ottenere l'indirizzo verso la pagina del veicolo, viceversa per le pagine foglia il processo di elaborazione prevede l'estrazione delle features del veicolo. Sia l'elaborazione di una pagina per estrarne gli hyperlinks sia l'elaborazione per estrarre le features naturalmente vengono effettuate su pagine il cui contenuto è già stato processato dagli script in modo da ottenere la stessa pagina che un utente visualizzerebbe su un browser tradizionale e in entrambi i casi il processo di basa sugli standard definiti in HTML5 in modo da essere generale, indipendente dalla fonte, dalla lingua della fonte e da eventuali modifiche che potranno essere apportate all'interno della pagina in versioni successive.

L'analisi (o parsing) in entrambi i casi prevede una fase preliminare in cui il contenuto della pagina viene ordinato, eliminando gli a capo non necessari, e pulito da eventuali regole CSS presenti (non necessarie in quanto l'obiettivo è analizzare il contenuto e non il posizionamento degli elementi all'interno della pagina o il loro stile grafico): il risultato della prima fase di pulizia è un documento di testo che contiene i tag di HTML5 uno per riga.

Nel caso la pagina sia stata richiesta da un utente nella modalità di estrazione delle informazioni da pagina singola o nel caso in cui il suo contenuto consenta di associarla al veicolo di interesse, dopo la fase di pulizia è possibile analizzare i tag all'interno dei quali la probabilità di trovare le features è più alta: dall'analisi delle fonti alimentanti più utilizzate effettuata insieme ai Weight Specialists di AMET, le features sono generalmente presentate in forma strutturata all'interno di tabelle o elenchi. Prendendo ad esempio in considerazione una tabella generica realizzata in HTML, lo standard prevede l'utilizzo di un tag che ne identifica l'inizio e la fine, un tag che ne identifica l'header e un numero variabile di tag per costruire le righe della tabella stessa:

```
1         <!--Table section-->
2         <table id="tableId" class="tableClass">
3             <!--Table header-->
```

```
4            <tr id="headerId" class="headerClass">
5                <!--First column-->
6                <th class="headerCellClass">Feature</th>
7                <!--Second column-->
8                <th class="headerCellClass">Value</th>
9            </tr>
10           <!--First row of the table-->
11           <tr id="row0">
12               <td class="cellClass">Brand</td>
13               <td class="cellClass">Renault</td>
14           </tr>
15           <!--Generic row of the table-->
16           <tr id="rowN">
17               <td class="cellClass">Model</td>
18               <td class="cellClass">Clio ENERGY TCe 90</td>
19           </tr>
20       </table>
```

Listing 5.2. Tabella HTML generica

Lo standard prevede che i tag al loro interno contengano attributi aggiuntivi che vengono interpretati e usati dai browser per renderizzare a schermo il contenuto della pagina tra i quali ad esempio un id per identificare univocamente il componente, un attributo classe all'interno del quale viene specificato l'elenco di classi CSS o un campo style per definire il layout dei componenti. Nel caso di studio preso in considerazione però, questi attributi non rappresentano l'informazione di interesse in quanto l'obiettivo non è renderizzare la pagina a schermo, bensì effettuare un parsing dei tag in maniera simile all' elaborazione effettuata da un comune browser, individuando il tag di inizio compreso tra "<>" e identificato da una parola chiave opportuna e il corrispondente tag di chiusura caratterizzato dalla stessa parola chiave e compreso tra "</>": l'informazione di interesse da estrarre è compresa tra il tag di inizio e il corrispondente tag di chiusura.

Il risultato della fase di pulizia della pagina in cui vengono eliminati gli a capo (sia quelli identificati dal carattere speciale di new line che il carriage return) e tutto ciò che non rappresenta la struttura HTML della

pagina (come ad esempio regole CSS incluse nel documento e non caricate da un foglio di stile esterno), consente un parsing agevole del documento e l'estrazione delle parti di interesse identificate da una certa parola chiave. Nell'esempio preso in considerazione in cui l'obiettivo è quello di estrarre una tabella, all'interno del documento le righe di una generica tabella sono comprese tra il tag iniziale "<table>" e finale "</table>". Dopo la fase di pulizia, la generica riga di una tabella sarà del tipo:

```
1        <tr><td>Modell</td><td>Clio ENERGY TCe 90</td></tr>
```

Listing 5.3. Generica riga di una tabella HTML dopo la fase di pulizia

Il risultato fondamentale della fase di pulizia è quindi quello di ottenere un documento di testo che contenga una per riga le informazioni di interesse, in un formato generalizzato, indipendente dalla fonte e dal contenuto e che risulti agevolmente interpretabile: nell'esempio risulta immediatamente chiara la posizione della chiave che rappresenta la feature da estrarre e del corrispondente valore associato.

```
1    private List<ExtractedFeature> ExtractFeaturesFromTable(string
         path)
2    {
3        ...
4        foreach (string line in File.ReadLines(path))
5        {
6            string[] parts = line.Split("</td><td");
7
8            if (2 == parts.Length)
9            {
10               string featureName = ExtractFeatureName(parts[0]);
11               string featureValue = ExtractFeatureValue(parts[1]);
12           }
13       ...
14   }
15
16   private static string ExtractFeatureName(string line)
17   {
```

```
18          string[] parts = line.Split("\">");
19
20          if (2 < parts.Length)
21          {
22              return parts[2].Trim();
23          }
24
25          return string.Empty;
26      }
27
28      private static string ExtractFeatureValue(string line)
29      {
30          string[] parts = line.Split("\">");
31
32          if (1 < parts.Length)
33          {
34              return parts[1].Replace("</td></tr>", "").Trim();
35          }
36
37          return string.Empty;
38      }
```

Listing 5.4. Estrazione della chiave e del valore da una generica riga di una tabella HTML

### 5.1.3 Elaborazione delle features estratte

Il passo successivo all'estrazione consiste nel verificare che la chiave estratta, ovvero che l'informazione contenuta all'interno della tabella, sia effettivamente una feature di interesse associabile a una delle Libra Features definite e nel riconoscere il tipo di feature estratta: le features possono essere di due tipi differenti e in funzione del tipo, il processo di elaborazione è differente. Il processo di associazione della feature estratta alla Libra Feature corrispondente, se tale associazione esiste, viene ricercata

all'interno del dizionario che viene definito a priori da un amministratore o da un utente che abbia i privilegi minimi per definire tali associazioni.

Ogni feature estratta all'interno del Web Crawler viene rappresentata, seguendo il pattern MVC, dal Model ExtractedFeature il quale descrive le proprietà deve logicamente possedere una feature:

```
1         public class ExtractedFeature
2         {
3             public string FeatureName { get; set; }
4             public MEASURE_TYPE FeatureType { get; set; }
5             public string ExtractedValueString { get; set; }
6             public string FeatureValueString { get; set; }
7             public double FeatureValueDouble { get; set; }
8             public string FeatureUnitOfMeasure { get; set; }
9             public EXTRACT_REQUEST ExtractionStatus { get; set; }
10        }
```

Listing 5.5. Model di una feature estratta

Sia nella modalità di estrazione delle features da pagina singola, quanto nel processo di estrazione che considera l'elenco completo dei seed definiti all'interno dei quali ricercare le informazioni, l'utente ha la possibilità di definire un filtro di ricerca: un utente può decidere se effettuare una ricerca che estragga indistintamente tutte le features dalla pagina Web scaricata oppure se estrarne solo un sottoinsieme definito al momento della ricerca o all'interno di una lista configurabile e memorizzabile in modo da poter essere riusata per ricerche future. Questo sottoinsieme può ad esempio comprendere l'estrazione solo delle features di tipo numerico, ma è possibile definire dei filtri a livello di dettaglio maggiore e quindi visualizzare tra le features numeriche ad esempio solo quelle che indicano un peso o una lunghezza.

A tale scopo l'attributo EstractionStatus è di tipo enum può assumere valori interi che per ogni feature estratta indicano se quella feature estratta non era stata richiesta dall'utente, se era stata richiesta ma non è stata trovata all'interno della pagina, se era stata richiesta, è presente all'interno della pagina ma non è stato possibile estrarla perché non è presentata

in un formato strutturato o se è stata richiesta, è presente all'interno della pagina ed è stata correttamente estratta. Un utente a livello di interfaccia grafica ha quindi la possibilità di filtrare tra le categorie o visualizzarle tutte: durante l'estrazione risulta interessante estrarre tutte le features presenti all'interno della pagina nel caso in cui il processo incontri delle features che non sono presenti all'interno del dizionario, e che quindi non sono ancora una Libra Feature, perché il dizionario è ancora incompleto o perché si tratta di una nuova caratteristica lanciata sul mercato.

Il campo valore associato ad ogni feature dopo il processo di estrazione subisce un'ulteriore elaborazione con lo scopo di memorizzare all'interno del database delle features omogenee, confrontabili nel caso di valori numerici e in lingua inglese nel caso di campi di testo. Facendo riferimento ai valori numerici, perché l'applicazione sia in grado di riconoscere l'unità di misura eventualmente associata a quel valore, all'interno del database è stata definita una tabella che contiene l'elenco delle unità di misura di interesse e a ciascuna di esse è associato il tipo: quindi ad esempio al tipo "lunghezza" saranno associate tutte le unità di misura che rappresentano delle lunghezze, sia nel Sistema Internazionale di misura che nel Sistema Imperiale. Inoltre per ogni Libra Feature viene definita anche l'unità di misura target in modo che, se la feature estratta è presentata usando un'altra classe all'interno dello stesso tipo, il Web Crawler sia in grado di effettuarne l'equivalenza.

Il primo passo nel processo di elaborazione del campo valore consiste nel verificare se il valore estratto è un valore numerico e, in quel caso, se è presente l'unità di misura associata. Un campo viene considerato un valore numerico se rispetta una tra due condizioni:

- è una stringa di due elementi separati da spazio in cui il primo elemento è un valore numerico e il secondo è una stringa definita come unità di misura supportata dal Web Crawler: in questo caso il primo elemento viene interpretato come valore numerico, mentre il secondo rappresenta l'unità di misura associata a quel valore.

- il campo valore contiene un solo elemento ed è un valore numerico: in questo caso è presente il valore ma non l'unità di misura ad esso

associata. Per ogni Libra Feature è definita l'unità di misura target, per cui se durante il processo di estrazione questa non dovesse essere presente, al valore viene assegnata un'unità di misura di default che corrisponde all'unità di misura target

Se il primo elemento è un valore numerico ma gli elementi all'interno del campo sono più di due o in generale non vengono rispettate le due condizioni precedenti, il campo viene interpretato come una stringa.

```
1         private bool IsLength(string value)
2         {
3             string[] parts = value.Split(" ");
4
5             if(2 == parts.Length)
6             {
7                 for(int i = 0; i < lengthMeasures.Count; i++)
8                 {
9                     if
                          (parts[1].Trim().ToLower().Equals(lengthMeasures[i]))
10                    {
11                        return true;
12                    }
13                }
14            }
15
16            return false;
17        }
18
19        private bool IsWeight(string value) { ... }
20        private bool IsCapacity(string value) { ... }
21        ...
22
23        public MEASURE_TYPE MeasureType(string featureValue)
24        {
25            if (IsLength(featureValue))
26            {
27                return MEASURE_TYPE.LENGHT;
```

```
28              }
29              else if (IsWeight(featureValue))
30              {
31                  return MEASURE_TYPE.WEIGHT;
32              }
33              ...
34              else
35              {
36                  return MEASURE_TYPE.STRING;
37              }
38          }
```

Listing 5.6. Controllo del tipo di unità di misura associata alla grandezza

Viceversa, per i campi valore che contengono delle stringhe il processo di elaborazione è differente: a priori non è nota la lingua originale della pagina né la lingua del contenuto del campo. Così come per i campi di tipo numerico per cui l'obiettivo è quello di avere dei valori rappresentati dalla stessa unità di misura per una specifica feature in modo da essere omogenee e quindi confrontabili, per i campi di testo l'obiettivo è quello di memorizzare all'interno del database descrizioni delle features in lingua inglese.

Risulta pertanto fondamentale identificare innanzitutto la lingua originale con cui viene descritto il campo valore e in seguito effettuarne la traduzione: entrambi gli obiettivi vengono realizzati da processi ausiliari al Web Crawler che avviano uno script per il riconoscimento della lingua e un secondo che ne effettua la traduzione verso la lingua target, nel caso di studio preso in considerazione, l'inglese.

```
1           import sys
2           from langdetect import detect
3
4           if __name__ == "__main__":
5               NUM_PARAMS = len(sys.argv)
6               features = ''
7
```

```
8               for param in range(1, NUM_PARAMS):
9                   features += sys.argv[param] + ' '
10
11              lang = detect(features)
12
13              print(lang)
```

Il primo script realizzato in python viene lanciato dal Controller del Web Crawler come un processo .NET indipendente, che così come il browser, viene lanciato senza la necessità di un'interfaccia grafica. Il risultato dello script è una stringa che normalmente verrebbe visualizzata all'interno di un terminale come standard output, ma essendo il processo privo di GUI, il suo output viene rediretto e il risultato semplicemente salvato all'interno di una variabile di tipo string in .NET. Il riconoscimento della lingua utilizza la libreria langdect [13] [14] che rappresenta una reimplementazione in Pyhton della libreria Java di riconoscimento del testo di Google [15] [16], in grado di riconoscere con una precisione superiore al 99% le 50 lingue supportate, tra lingue europee e asiatiche: la libreria, così come nella sua versione originale, genera il dizionario per rilevare la lingua a partire dall'abstract database di Wikipedia e a partire da articoli giornalistici di varie fonti e in varie lingue, tra i quali ad esempio articoli di Google News in 24 lingue diverse, e restituisce in output due lettere che rappresentano il codice della lingua rilevata secondo lo standard ISO 693 [17], che rappresenta la nomenclatura internazionale per la classificazione delle lingue. L'algoritmo per il riconoscimento è un algoritmo non-deterministico in quanto l'output viene restituito sulla base del numero di corrispondenze all'interno del database utilizzano un classificatore Naive Bayes in cui la probabilità a posteriori viene espressa come:

$$P(C_k \mid X)^{(m+1)} \propto P(C_k \mid X)^{(m)} \cdot P(X_i \mid C_k) \tag{5.1}$$

in cui $C_k$ rappresenta la lingua, $X$ il documento da cui vengono estratti i termini per creare il dizionario usato per effettuare il riconoscimento e $X_i$ è l'i-esimo termine all'interno del documento.

Il risultato è pertanto una probabilità, nel senso che se viene eseguito su un testo troppo breve o ambiguo, il risultato potrebbe essere ogni volta diverso. All'interno dell'applicazione lo script riceve in input la lista completa contenente i nomi originali delle features e i campi di tipo stringa in modo che la probabilità di riconoscere correttamente la lingua sia più alta possibile.

```
1         private static string TranslateToEnglish(string str) { ... }
2
3         import argostranslate.package, argostranslate.translate
4         import sys
5
6         if __name__ == "__main__":
7             NUM_PARAMS = len(sys.argv)
8             features = ''
9
10            for param in range(2, NUM_PARAMS):
11                features += sys.argv[param] + ' '
12
13            from_code = sys.argv[1]
14            to_code = "en"
15
16            # Download and install Argos Translate package
17            available_packages =
                 argostranslate.package.get_available_packages()
18            available_package = list(
19                filter(
20                    lambda x: x.from_code == from_code and x.to_code
                         == to_code, available_packages
21                )
22            )[0]
23            download_path = available_package.download()
24            argostranslate.package.install_from_path(download_path)
25
26            # Translate
```

```
27              installed_languages =
                    argostranslate.translate.get_installed_languages()
28              from_lang = list(filter(
29                      lambda x: x.code == from_code,
30                      installed_languages))[0]
31              to_lang = list(filter(
32                      lambda x: x.code == to_code,
33                      installed_languages))[0]
34
35              translation = from_lang.get_translation(to_lang)
36              translatedText = translation.translate(features)
37
38              print(translatedText)
```

Allo stesso modo, un secondo processo lanciato come processo parallelo all'applicativo Web Crawler, lancia un secondo script python che riceve in ingresso il codice della lingua secondo lo standard ISO 693 ottenuto come risultato del processo di rilevazione della lingua precedentemente eseguito, e una stringa di testo che rappresenta il campo valore della feature di tipo testo da tradurre verso la lingua obiettivo, che nel caso di studio in considerazione è l'inglese. Il risultato della traduzione è ancora una stringa di testo ritornata dallo script come standard output del processo che viene di nuovo intercettato e rediretto all'interno di una variabile all'interno del processo principale dell'applicativo Web Crawler.

Lo script che si occupa di effettuare la traduzione usa un dizionario offline, richiamando la libreria python Argos Translate [25] [26] che a sua volta si basa su OpenNMT [27], progetto open source sviluppato nel 2016 dal gruppo NLP di Harvard e ad oggi mantenuto da SYSTRAN e Ubiqus.

Entrambi gli script vengono eseguiti da processi ausiliari lanciati dall'applicativo .NET principale: ciascuno dei due processi necessita del nome dell'eseguibile da lanciare, in questo caso il path completo all'eseguibile di python, il nome dello script da eseguire e gli argomenti necessari allo script stesso. Oltre ai parametri fondamentali per il funzionamento, il processo viene configurato per essere eseguito senza un terminale e per

intercettare l'output risultante dall'esecuzione.

```
1     private static string DetectLanguage(string str)
2     {
3         CrawlerConfig crawlerConfig = new();
4
5         ProcessStartInfo languageDetector = new()
6         {
7             FileName = crawlerConfig.Get_PYTHON_PATH(),
8             UseShellExecute = false,
9             RedirectStandardOutput = true,
10            Arguments = string.Format("{0} {1}",
                  crawlerConfig.Get_LANGUAGE_DETECTOR_PATH(), str)
11        };
12
13        Process process = new()
14        {
15            StartInfo = languageDetector
16        };
17
18        process.Start();
19
20        using StreamReader sr = process.StandardOutput;
21
22        string languageDetected = sr.ReadToEnd();
23
24        process.Kill();
25        sr.Close();
26
27        return languageDetected;
28    }
```

Listing 5.7. Processo ausiliario per eseguire script python

Il risultato viene quindi salvato all'interno di una variabile di tipo stringa e ritornato alla funzione chiamante, mentre il processo viene terminato per liberare le risorse che erano state allocate per la sua esecuzione.

### 5.1.4 Salvataggio delle features

L'ultimo passo del processo di estrazione delle features consiste nel mostrare i risultati ottenuti all'utente che ha richiesto il servizio: a questo punto potrà decidere se salvare tutte le features estratte o solo un sottoinsieme e se è il caso di effettuare delle modifiche prima di effettuare il salvataggio.

Per ogni Libra Feature salvata il Web Crawler memorizza come chiave primaria all'interno del database il riferimento alla feature in modo da poter estrarre il valore e la sua unità di misura quando richiesto, il riferimento al seed da cui è stata estratta la feature per poter associare il livello di affidabilità al dato estratto e naturalmente l'identificativo del veicolo al quale è associata la feature estratta. Il riferimento al seed da cui è stata estratta l'informazione in chiave primaria consente la possibilità di memorizzare per uno stesso veicolo, la stessa feature estratta da siti diversi ma con possibili livelli di affidabilità: in fase di confronto, la visualizzazione di default prevede che il valore mostrato di default all'utente sia il valore con il livello di affidabilità maggiore tra quelli disponibili.

| | vehicleId | featureId | seedId | featureValueString | featureValueNumber | featureTimestamp | featureUnitOfMeasure | featureSource |
|---|---|---|---|---|---|---|---|---|
| 25 | 1002 | 28 | 1 | 3 | NULL | 31/10/2022 14:35:37 | | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 26 | 1002 | 29 | 1 | Einspritzung | NULL | 31/10/2022 14:35:37 | | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 27 | 1002 | 30 | 1 | 4 | NULL | 31/10/2022 14:35:37 | | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 28 | 1002 | 31 | 1 | NULL | 898 | 31/10/2022 14:35:37 | ccm | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 29 | 1002 | 32 | 1 | 66 kW (90 PS) / 140 ... | NULL | 31/10/2022 14:35:37 | | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 30 | 1002 | 33 | 1 | 5250 U/min | NULL | 31/10/2022 14:35:37 | | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 31 | 1002 | 34 | 1 | 2250 U/min | NULL | 31/10/2022 14:35:37 | | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 32 | 1002 | 35 | 1 | Serie | NULL | 31/10/2022 14:35:37 | | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 33 | 1002 | 40 | 1 | NULL | 4063 | 31/10/2022 14:35:37 | mm | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 34 | 1002 | 41 | 1 | NULL | 1732 | 31/10/2022 14:35:37 | mm | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 35 | 1002 | 42 | 1 | NULL | 1945 | 31/10/2022 14:35:37 | mm | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 36 | 1002 | 43 | 1 | NULL | 1448 | 31/10/2022 14:35:37 | mm | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 37 | 1002 | 44 | 1 | NULL | 2589 | 31/10/2022 14:35:37 | mm | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 38 | 1002 | 45 | 1 | NULL | 120 | 31/10/2022 14:35:37 | mm | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 39 | 1002 | 46 | 1 | NULL | 106 | 31/10/2022 14:35:37 | m | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 40 | 1002 | 47 | 1 | NULL | 300 | 31/10/2022 14:35:37 | l | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 41 | 1002 | 48 | 1 | NULL | 1146 | 31/10/2022 14:35:37 | l | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 42 | 1002 | 49 | 1 | Serie | NULL | 31/10/2022 14:35:37 | | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 43 | 1002 | 50 | 1 | NULL | 1157 | 31/10/2022 14:35:37 | kg | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 44 | 1002 | 51 | 1 | NULL | 1621 | 31/10/2022 14:35:37 | kg | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 45 | 1002 | 52 | 1 | NULL | 464 | 31/10/2022 14:35:37 | kg | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 46 | 1002 | 53 | 1 | NULL | 1200 | 31/10/2022 14:35:37 | kg | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 47 | 1002 | 54 | 1 | NULL | 575 | 31/10/2022 14:35:37 | kg | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 48 | 1002 | 56 | 1 | NULL | 75 | 31/10/2022 14:35:37 | kg | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 49 | 1002 | 57 | 1 | NULL | 80 | 31/10/2022 14:35:37 | kg | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 50 | 1002 | 58 | 1 | Schrägheck | NULL | 31/10/2022 14:35:37 | | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |
| 51 | 1002 | 59 | 1 | 5 | NULL | 31/10/2022 14:35:37 | | https://www.adac.de/rund-ums-fahrzeug/autokatalog/marken-modelle/renault/clio/iv-facelift/258057/#technische-daten |

Figura 5.1. Libra Features memorizzate all'interno del database

Per ciascuna Libra Feature inoltre, vengono memorizzate delle informazioni ausiliarie che includono il link alla pagina da cui è stata estratta la feature e il momento in cui il record è stato inserito all'interno della tabella in modo da tenere traccia sia della fonte che dell'informazione temporale, nel caso in cui in futuro si voglia effettuare la stessa estrazione per verificare se è disponibile una versione aggiornata del dato.

### 5.1.5 Selezione ed estrazione da pagina singola

Gli aspetti di automazione su cui il Web Crawler si focalizza riguardano la ricerca della pagina contenente le features e l'estrazione delle features stesse. Un utente potrebbe quindi poter aver già identificato all'interno di uno dei siti web definiti come fonti alimentanti del Web Crawler, la pagina contenente le features.

Inizialmente viene proposto all'utente l'elenco completo delle features presenti all'interno del database e che caratterizzano tutti i veicoli, anche se una certa feature non è sempre necessariamente presente per ogni veicolo (un esempio è la capacità della batteria per un veicolo con motore a combustione interna). Da questo elenco l'utente ha la possibilità di filtrare sul tipo di feature e quindi di filtrare sulle due macro-categorie di features (numeriche o di tipo stringa) e per ciascuna micro-categoria, ha la possibilità di effettuare un filtro più mirato, potendo selezionare ad esempio tra le features numeriche solo quelle che rappresentano una distanza o un peso.

Questa selezione può essere salvata all'interno dell'applicazione in modo da dare la possibilità all'utente di effettuare la stessa ricerca, sullo stesso veicolo o su veicoli diversi, ricercando lo stesso sottoinsieme di features senza la necessità di dover ri-effettuare il filtro.

Una volta selezionato l'insieme di features da ricercare, l'utente può scegliere all'interno della modale il veicolo definito all'interno dell'apposita sezione, la fonte da cui le features verranno estratte e il link alla pagina che verrà scaricata e parsificata.

Il processo di parsing elabora il contenuto della pagina, eseguendo gli script che popolano dinamicamente il contenuto del documento, ne estrae

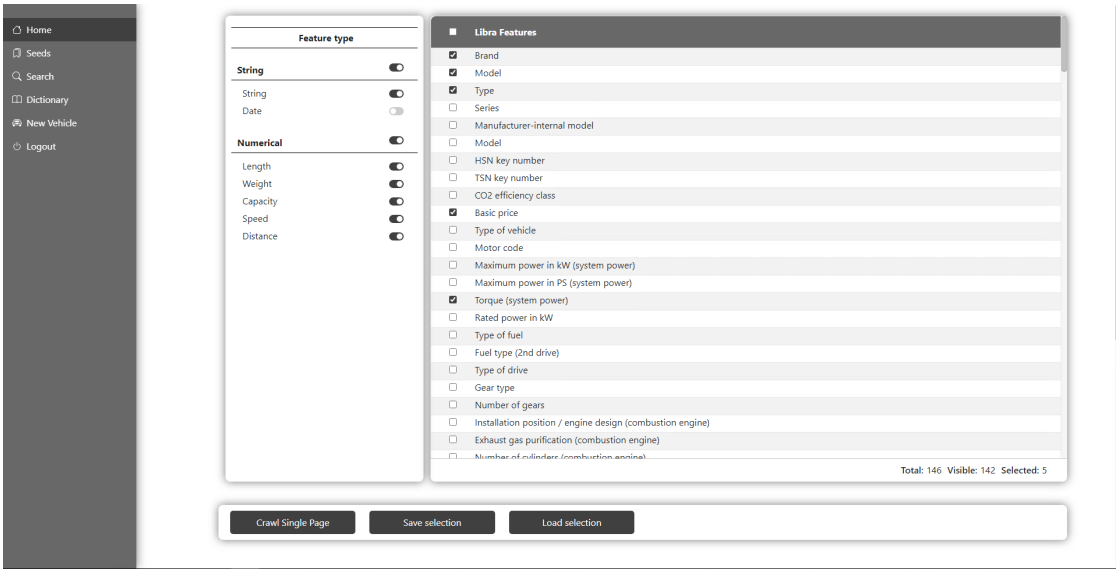


Figura 5.2. Elenco delle Libra Features

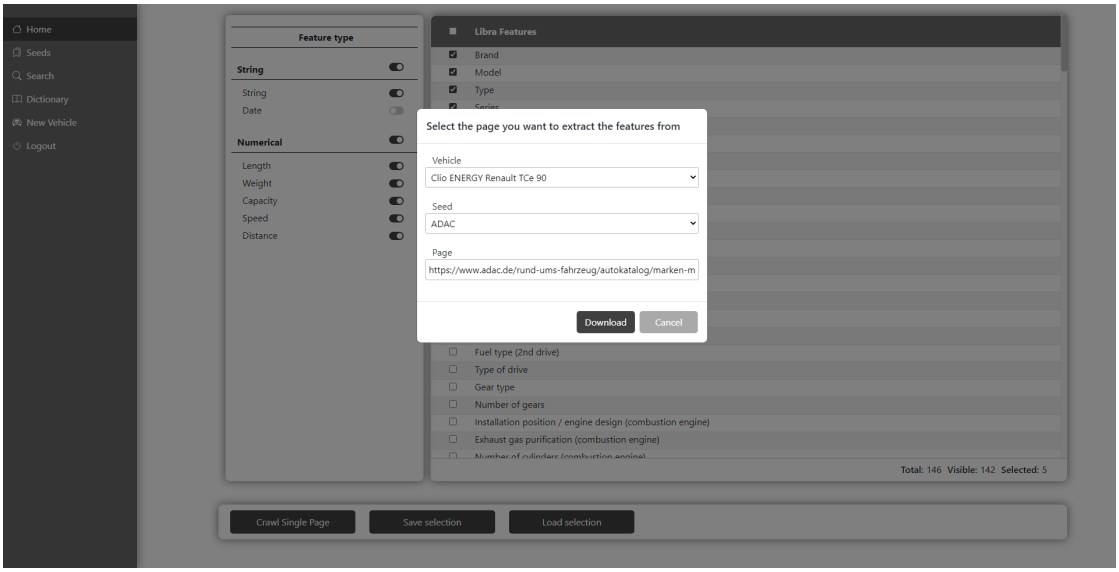


Figura 5.3. Download di una pagina dal web

le features e le elabora in funzione del loro tipo. Al termine del processo, viene mostrato all'utente il risultato ottenuto: la visualizzazione di default prevede che vengano mostrate all'utente le features che erano state

richieste e sono effettivamente state estratte dalla pagina.

Al termine dell'estrazione, le features selezionate dall'utente possono assumere stati diversi. Una feature infatti:

- può essere stata richiesta dall'utente come una caratteristica del veicolo da cercare all'interno della pagina che effettivamente è presente ed è stato possibile estrarla
- viene richiesta dall'utente, potrebbe essere presente all'interno della pagina in quanto la sua occorrenza viene trovata ma non è stato possibile estrarla perché ad esempio si trova all'interno di un paragrafo all'interno di un testo come informazione non strutturata
- viene richiesta e non è presente all'interno della pagina

A queste 3 condizioni possibili per le features richieste, si aggiunge un quarto stato che riguarda le features estratte che non vengono richieste dall'utente. La selezione dell'utente riguardo alle features riguarda la visualizzazione delle stesse quando queste vengono renderizzate all'interno della GUI. Tuttavia dalla pagina vengono sempre estratte tutte le features disponibili: alla base di tale scelta, la prima ragione risiede nel fatto che l'utente dopo aver selezionato le features da estrarre potrebbe richiedere l'estrazione anche di altre features per cui il Web Crawler dovrebbe chiedere nuovamente la pagina al server, effettuare nuovamente il parsing e tornare il risultato all'utente generando un overhead in termini prestazionali.

La seconda motivazione invece fa riferimento al fatto che il Web Crawler potrebbe incontrare all'interno della pagina una nuova feature che potrebbe non essere stata richiesta in quanto non presente all'interno dell'elenco delle Libra Features, perché è una caratteristica che non era ancora stata presa in considerazione e quindi da aggiungere all'elenco o perché rappresenta una novità sul mercato.

Al termine dell'estrazione le features vengono presentate all'utente: in particolare per ogni feature viene presentata la chiave con cui quella caratteristica viene identificata all'interno della pagina di provenienza, la Libra Feature associata, il valore della feature così come è contenuto all'interno della pagina, la tipologia di feature e nel caso in cui la feature

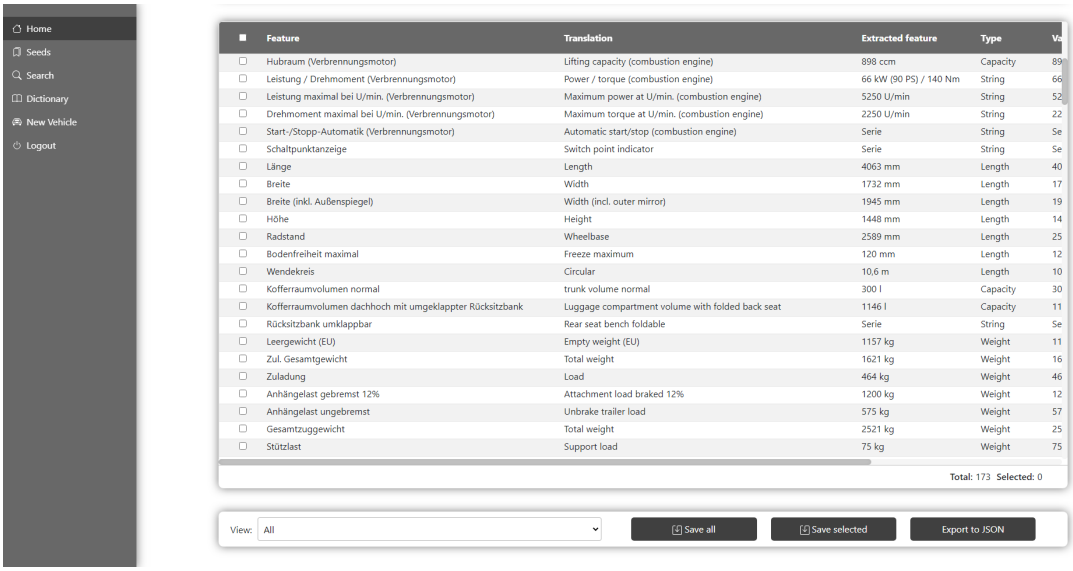


Figura 5.4. Elenco delle features estratte

sia una caratteristica numerica che rispetta le condizioni di parsing, il suo valore e l'unità di misura. Per le features di tipo stringa il Web Crawler ne effettua la traduzione in lingua inglese, indipendentemente dalla lingua originale del documento da cui è stata estratta, mentre le features di tipo numerico vengono opportunamente convertite nell'unità di misura definita per la Libra Feature associata.

Per ciascuna delle features all'interno della tabella, l'utente ha la possibilità di modificarle e di selezionare quali caratteristiche salvare e quindi associare al veicolo oppure esportarle in formato JSON.

Al termine viene riepilogato in forma grafica [1] il risultato dell'estrazione, restituendo all'utente la percentuale di features che sono state richieste e correttamente estratte, le features che sono state richieste e possibilmente presenti all'interno della pagina ma non estratte, quelle che sono state richieste ma non sono presenti e infine il numero di caratteristiche che sono state estratte ma che non facevano parte della selezione dell'utente.

[1] Tutti i grafici all'interno dell'applicazione Web Crawler sono stati realizzati come files multimediali in SVG e si basano sulla libreria JavaScript D3.js [28]

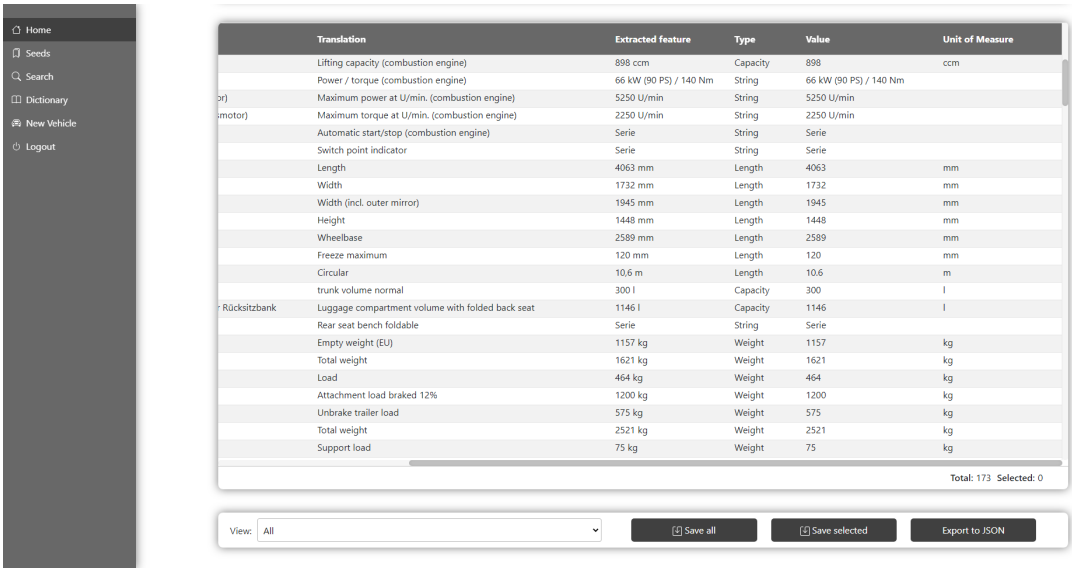


Figura 5.5. Elenco delle features estratte correttamente riconosciute

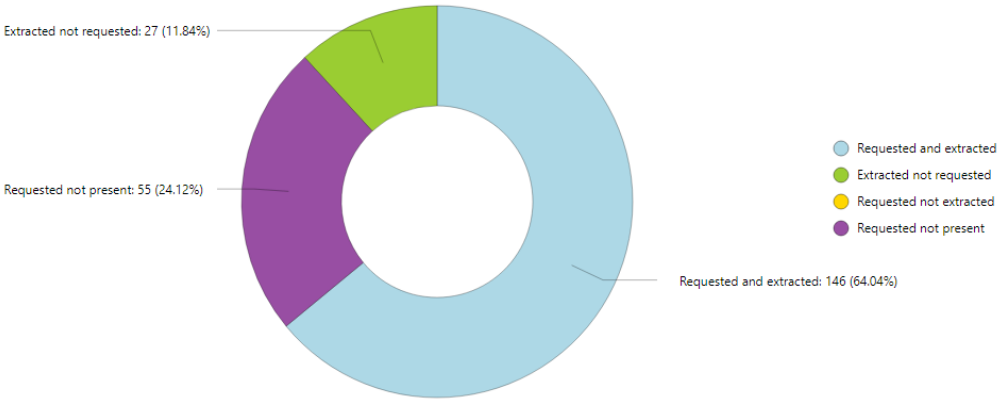


Figura 5.6. Riepilogo del processo di estrazione

### 5.1.6 Ricerca nel sottoinsieme di seeds

Il secondo macro-processo che il Web Crawler si pone come obiettivo di automatizzare riguarda la visita dei siti Web identificati come possibili fonti alimentanti e quindi il processo di ricerca del veicolo, se disponibile, all'interno degli alberi generati dai seeds.

L'utente ha quindi la possibilità di scegliere dal menù a tendina uno dei veicoli precedentemente definiti e del quale ricercare le features. Per il veicolo selezionato verranno quindi visitate tutte le fonti in un ordine di priorità dato dal livello di affidabilità della fonte stessa.

Al termine del processo di ricerca viene restituito all'utente l'elenco delle pagine identificate come possibili fonti alimentanti da cui estrarre le informazioni: l'elenco rappresenta un sottoinsieme dei seed di partenza in quanto un veicolo potrebbe non essere necessariamente presente all'interno di tutte le fonti. Per ciascuna pagina che rispetta le policies definite per l'associazione del veicolo, vengono riportati il link di provenienza, la fonte da cui quella pagina è stata estratta, il relativo livello di confidenza e un indicatore sul numero di associazioni tra il veicolo e la pagina.

L'ordine con cui le pagine vengono presentate all'utente parte dal livello di confidenza della fonte in modo che vengano presentate per primi i documenti con il livello di affidabilità maggiore. A parità di livello di affidabilità, il secondo criterio di ordinamento prevede che vengano mostrate per prime le pagine con il numero maggiore di associazioni e quindi una probabilità maggiore di corrispondenza tra il veicolo richiesto e la pagina identificata.

Tutte le pagine possono quindi essere sottoposte al controllo da parte dell'utente che può decidere se procedere con l'estrazione delle features da quei documenti o scartarli. Per tutte le pagine selezionate dall'utente quindi il Web Crawler procede con il processo di estrazione e i risultati ottenuti vengono visualizzati con le stesse modalità delle informazioni ricavate dal parsing di una pagina singola.

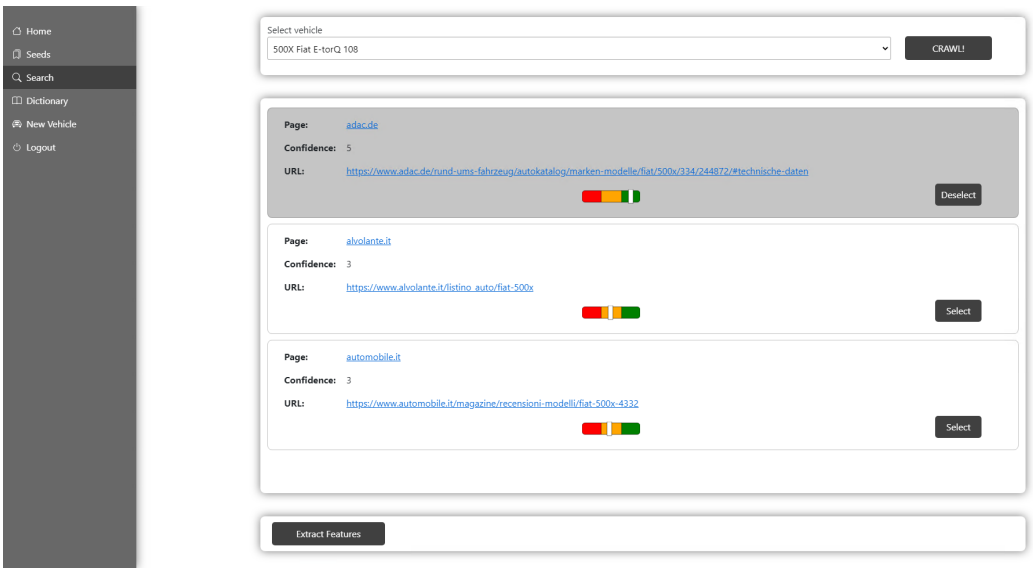


Figura 5.7. Risultati del processo di ricerca

# Capitolo 6

# Analisi dei risultati

L'obiettivo principale del Web Crawler è quello di essere un possibile strumento a supporto del lavoro manuale di ingegneri altamente specializzati per automatizzare un processo attualmente manuale ed estremamente energivoro in termini di tempo richiesto. Così come lo sviluppo iniziale che riguarda la definizione dei siti Web da visitare considerati più affidabili e la costruzione del dizionario contenente le Libra Features è stato guidato dalla loro esperienza operativa, i risultati ottenuti dal Web Crawler sono stati confrontati con i risultati ottenuti in maniera manuale in modo da effettuare un fine tuning dell'algoritmo al fine di ottenere una soluzione che risponda quanto più possibile alle loro necessità.

Gli aspetti principali che riguardano l'automazione sono principalmente due: il primo riguarda la ricerca del veicolo di interesse all'interno del sottoinsieme di siti web definito mentre il secondo fa riferimento al processo di estrazione, elaborazione e salvataggio delle informazioni.

### 6.0.1 Processo di ricerca

Ciascun sito Web definito all'interno del sottoinsieme di fonti considerate come fonti da cui sono state estratte il maggior numero di informazioni affidabili, può essere logicamente interpretato come un grafo bidirezionale ciclico in quanto ciascuna pagina potrebbe contenere degli hyperlinks che indirizzano a se stessa, a nodi già visitati in precedenza o su siti web differenti da quello visitato. Il primo passo consiste nel trasformare il grafo generico in un albero in cui il punto iniziale della ricerca rappresenta la

radice, ogni pagina rappresenta un nodo intermedio e gli hyperlinks gli archi. A questo punto la visita di tale albero può essere effettuata seguendo due diverse modalità: una visita in ampiezza in cui vengono visitati prima i nodi fratelli sullo stesso livello o una visita in profondità in cui il nodo fratello allo stesso livello viene visitato solo quando è stata visitata una foglia, ovvero, nel caso di studio di interesse, quando una pagina non contiene più hyperlinks validi.

Durante la prima fase di sviluppo del Web Crawler è stata esplorata innanzitutto una soluzione in cui l'albero viene visitato in ampiezza con l'obiettivo di ampliare quanto più possibile lo spazio di ricerca e aumentare la probabilità di visitare una pagina che effettivamente contenesse tutte le informazioni necessarie. Se da un lato però uno spazio di ricerca decisamente più ampio aumenta la probabilità di visitare la pagina contenente le features di interesse, all'aumentare delle pagine visitate il tempo necessario alla ricerca cresce proporzionalmente con un fattore di proporzionalità almeno pari al tempo di attesa tra una richiesta e l'altra per evitare di sovraccaricare il server visitato, a cui viene aggiunto il tempo di esecuzione degli script della pagina e di elaborazione per estrarne gli hyperlinks. A questo tempo di attesa potrebbe addizionarsi anche un tempo di rivisita nel caso in cui la richiesta precedente non sia andata a buon fine, fino a un massimo di tre richieste per ogni hyperlink.

Oltre all'elevato tempo di visita e all'elevata quantità di risorse necessarie per la ricerca in termini di memoria, la struttura dei siti Web definiti all'interno del sottoinsieme di seed è tale per cui l'informazione di interesse spesso è contenuta all'interno delle foglie, mentre i nodi intermedi contengono hyperlinks verso veicoli simili a quello correntemente visualizzato che potrebbero essere di interesse durante la navigazione di un utente. Il rischio è quindi molto spesso quello di visitare rami che in realtà a livello logico sarebbero nodi radice di altri alberi e quindi di ricercare features per veicoli differenti da quello di interesse, saturando le risorse e dilatando il tempo necessario al punto che nella maggior parte dei casi l'algoritmo di visita non arrivi a convergere in tempi ragionevoli. Il motivo principale per cui la visita in ampiezza non genera un output interessante in tempi utili risiede nel fatto che se il veicolo suggerito si

trova ancora all'interno dello stesso dominio del sito Web e quel nodo non è ancora stato visitato, le condizioni per fermare la visita anticipatamente e quindi per tagliare solo la piccola porzione di ramo che contiene il veicolo di interesse, non vengono rispettate. Nel caso peggiore, quindi, finché il sito Web visitato suggerisce veicoli sempre differenti, la visita prosegue finché non vengono interrogati tutti i veicoli presenti all'interno del database.

Nelle occasioni in cui il sito web presenti solo un sottoinsieme molto ristretto di veicoli visitati, l'algoritmo di visita in ampiezza riesce a convergere in quanto a questo punto l'albero non ha più nodi da visitare.

Nonostante ciò, da questa prima fase di analisi sono emersi due modelli di comportamento comuni a tutte le fonti dati visitate:

- all'interno di una pagina il contenuto principale riguarda il veicolo ricercato mentre i suggerimenti su veicoli simili che potrebbero essere interessanti per l'utente vengono posizionati più in fondo nella pagina

- l'informazione contenuta nei database dei siti web visitati deve poter essere immediatamente accessibile in seguito alla richiesta di un utente ed è memorizzata in maniera strutturata, per cui un metodo che effettua la ricerca di un veicolo su un database interno è in grado di indicizzare correttamente le informazioni. Ciò implica che se un veicolo da ricercare viene descritto in maniera dettagliata e precisa, molto probabilmente le informazioni relative a quel veicolo verranno presentate in cima alla lista dei risultati

Viceversa, effettuando le stesse modifiche al grafo in modo da eliminare a livello logico gli archi che puntano a nodi già visitati o al nodo corrente in modo da trasformarlo ancora in albero, la visita in profondità visita le foglie prima di visitare il nodo fratello del nodo corrente.

In base alle due considerazioni precedenti, è molto più probabile che le prime foglie visitate siano effettivamente i nodi che contengono le informazioni da estrarre. In questo caso il tempo di ricerca dipende ancora dal numero di pagine visitate e dal tempo necessario alla loro elaborazione, quindi nel caso peggiore di ricerca l'algoritmo di visita esplorerebbe

comunque tutto l'albero. Tuttavia rispetto al grafo visitato è possibile utilizzare la conoscenza a priori acquisita durante la prima fase e quindi definire dei criteri di stop che consentano di potare lo spazio di ricerca. Tali condizioni sono dei criteri di accettazione della pagina che consentono di stabilire se il documento visualizzato è associabile al veicolo cercato in modo da interrompere anticipatamente la ricerca. Naturalmente non avendo a disposizione l'intero spazio di ricerca, l'ottimo non sarà necessariamente un ottimo globale, bensì locale.

Il primo criterio di stop si basa sulla definizione del veicolo: ciascun veicolo in fase di creazione viene caratterizzato da un insieme di parametri che consentono di identificarlo univocamente sia rispetto ad altri veicoli che rispetto a versioni differenti dello stesso (basti pensare ad esempio a due veicoli dello stesso modello ma tali per cui la loro configurazione differisce rispetto alla motorizzazione). In base a questo principio la ricerca prosegue sulla base di quanti parametri che caratterizzano il veicolo sono presenti in percentuale all'interno della pagina:

- se la pagina contiene un numero di caratteristiche al di sotto del 50% delle caratteristiche che identificano il veicolo, l'associazione è troppo debole: la pagina potrebbe contenere informazioni, quali ad esempio il brand e il modello, che sono comuni a molti veicoli della stessa casa produttice
- se il numero di caratteristiche è compreso tra il 50% e il 75%, l'associazione può già considerarsi accettabile per cui il riferimento alla pagina viene mantenuta in memoria ed eventualmente rimpiazzato da un'altra pagina con una associazione più alta in percentuale
- al di sopra del 75% l'associazione si considera buona: il riferimento alla pagina viene salvato in memoria per essere sottoposto all'utente e la ricerca all'interno dell'albero per quella fonte viene interrotta

Mentre il primo criterio definisce una condizione di stop sulla base di una soglia che fa riferimento al numero di parametri che caratterizzano il veicolo, il secondo assicura che l'algoritmo converga: potrebbe infatti verificarsi che il veicolo richiesto non sia presente all'interno del database della fonte inchiestata. Ciò vorrebbe dire visitare il grafo senza trovare una soluzione e nel caso peggiore visitare l'elenco di tutti i veicoli correlati

alla ricerca proposti dal sito web, saturando le risorse del Web Crawler quanto quelle del server visitato. Per tale ragione il secondo criterio interrompe anticipatamente l'algoritmo se viene raggiunta una certa soglia di nodi visitati. Questo secondo criterio, insieme alla conoscenza preliminare per cui è più probabile che l'informazione di interesse sia contenuta su una foglia e in particolare che i risultati più pertinenti vengono generalmente presentati prima, garantisce una convergenza dell'algoritmo in un tempo che nel caso migliore dipende solo dalla profondità dell'albero (l'informazione di interesse si trova sulla prima foglia, che contiene quindi un numero di caratteristiche almeno pari al 75% dei parametri che identificano il veicolo) mentre nel caso peggiore il tempo è fissato superiormente ed è una funzione del numero massimo di nodi visitabili definito come parametro, del tempo di elaborazione e del delay temporale tra una richiesta all'altra per rispettare il carico sul server visitato. Nel caso medio la ricerca prosegue finché una delle due condizioni non è soddisfatta, e quindi viene trovata una pagina con un numero di associazioni in percentuale alto oppure viene raggiunto il numero massimo di nodi visitabili.

Al termine della visita di una determinata fonte, se nessuna pagina contiene un numero di associazioni superiore alla soglia minima del 50% quella fonte non viene proposta all'utente come una possibile fonte alimentante da cui estrarre le informazioni per il veicolo scelto, altrimenti viene presentata all'utente la pagina come una possibile fonte che potrebbe contenere le features.

### 6.0.2 Processo di estrazione delle features

Il secondo aspetto dell'automazione del processo riguarda l'estrazione delle features dalla pagina: l'elenco delle Libra Features comprende l'insieme delle proprietà che caratterizzano un veicolo di interesse che vengono analizzate dagli Weight Specialists durante la fase di Benchmark prima e nella fase di definizione del target successivamente. Le Libra Features tuttavia identificano una determinata feature all'interno del contesto dell'applicazione, ma siti Web differenti potrebbero usare nomenclature diverse o sinonimi per riferirsi ad una stessa caratteristica. Per questa

ragione la definizione dell'elenco delle Libra Features non è una condizione sufficiente per identificare tutte le features da estrarre ma è essenziale definire all'interno di un dizionario le associazioni tra il nome con cui viene identificata una feature da una certa fonte e la Libra Feature corrispondente.

Rispetto alla fase di visita dell'albero, il processo di costruzione del dizionario delle associazioni è un processo dinamico e iterativo di apprendimento: al momento della definizione, inizialmente il dizionario delle associazioni non conterrà nessun collegamento tra i nomi con cui vengono identificate le features all'interno di una pagina Web e la Libra Feature corrispondente. La una prima fase del processo di estrazione delle features da pagina singola è stata quindi rilevante per la costruzione del dizionario delle associazioni.

La ricerca di tali associazioni ha visto come prima fonte alimentante adac [24], testata specializzata in lingua tedesca, considerata la più affidabile in termini di correttezza delle informazioni riportate e la più completa in termini di quantità di dati.

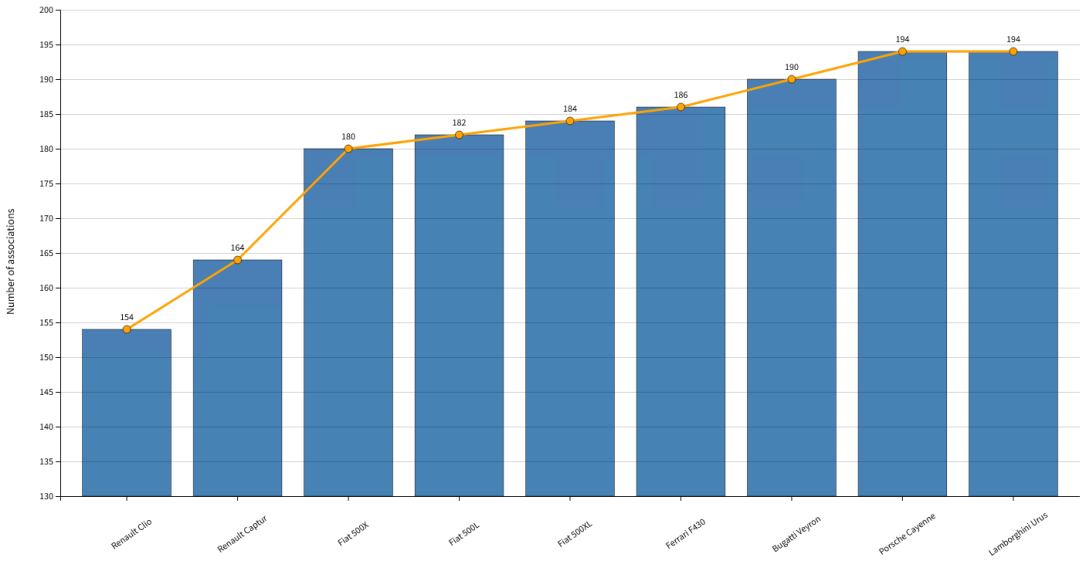


Figura 6.1. Numero di associazioni tra le features di una pagina Web generica e le Libra Features

Il grafico riporta il numero di associazioni tra le features estratte e le Libra Features in funzione dei veicoli ricercati secondo un ordinamento di ricerca temporale. Ad eccezione del picco iniziale dovuto al fatto che il database della associazioni è inizialmente vuoto, si può notare come all'interno della stessa fonte per veicoli differenti ma appartenenti allo stesso segmento di mercato, il numero di features che vengono incontrate per la prima volta diminuisce progressivamente, per poi tornare ad aumentare cambiando la categoria di veicolo. Il motivo di tale comportamento è dovuto al fatto che anche all'interno della stessa fonte, una stessa feature potrebbe essere identificata in maniera diversa nonostante a livello logico entrambe rappresentino la stessa proprietà del veicolo.

Si può notare come, presa in considerazione ad esempio la categoria dei Veicoli Commerciali Leggeri (LCV) con il motore a combustione interna, il numero di nuove features e di conseguenza il numero di nuove associazioni, diminuisca progressivamente in quanto molte features naturalmente risulteranno comuni a tutti i veicoli (basti pensare alle dimensioni, alla cilindrata o al costo): le differenze in molti casi sono dovute al fatto che una stessa caratteristica comune potrebbe essere riportata con due definizioni diverse (ne è un esempio la distanza tra gli assi che potrebbe essere identificata come passo o come wheelbase), ma in altre situazioni indicava una feature che non era presente nei veicoli esaminati in precedenza, perché non era una feature caratterizzante per quel veicolo o perché non era stata riportata per altri veicoli. Tale comportamento risulta ancora più evidente ancora all'interno della categoria di veicoli LCV, con il passaggio all'analisi dei veicoli elettrici: in questo caso, ad esempio, tra le features di cui costruire una nuova associazione, è possibile citare la capacità della batteria che caratterizza i veicoli elettrici che non viene riportata per veicoli a combustione interna.

A valle della fase di definizione e di inizializzazione usando adac come fonte alimentante, il dizionario delle Libra Features contiene circa 150 features che caratterizzano i veicoli. Le associazioni definite in questa fase, nonostante siano state costruite sulla base di adac utilizzata come fonte alimentante, risultano valide per riconoscere ed associare features estratte da altre fonti. Il processo rimane tuttavia valido anche per nuove features estratte da fonti differenti e rimane rilevante durante il funzionamento

dell'applicativo in quanto con la stessa strategia è possibile individuare e segnalare agli utenti con privilegi di amministratore sul dizionario delle associazioni, nuove features che vengono incontrate per la prima volta. Incontrare nuove features e segnalarne l'aggiunta risulta fondamentale per due ragioni: la prima in quanto consente di avere un database completo e far sì che il Web Crawler sia in grado di riconoscere la stessa feature in futuro, la seconda perché in questo modo è possibile segnalare potenziali nuove caratteristiche che vengono lanciate sul mercato (ad esempio un nuovo componente che viene integrato di serie nei veicoli).

### 6.0.3 Caso d'uso: confronto di veicoli

Al termine della fase di inizializzazione, il Web Crawler può passare alla sua fase più prettamente operativa in cui effettuare in maniera automatica il processo di riconoscimento ed estrazione delle features. All'utente viene data la possibilità di effettuare una ricerca per l'insieme completo di Libra Features oppure se effettuare una ricerca più mirata su un sottoinsieme di features di interesse. Un utente potrebbe voler ad esempio effettuare uno studio che riguarda categorie precise di features e quindi limitare la ricerca per esempio alle caratteristiche che descrivono le dimensioni e il peso del veicolo. Al termine dell'estrazione vengono presentati i risultati all'utente il quale ha la possibilità di modificarli e scegliere quindi se salvarli e/o esportarli in un formato standard definito da uno schema JSON. L'algoritmo prevede che alle features di tipo numerico presentate all'utente venga assegnata l'unità di misura estratta, se presente e conforme ai criteri definiti, altrimenti viene assegnata l'unità di misura di default per quella feature. Nel caso in cui l'unità di misura del dato estratto non sia quella di default, l'utente potrebbe avere la necessità di effettuare l'equivalenza prima di salvarne il valore, col l'obiettivo di inserire all'interno del database dei valori omogenei per ciascuna feature in modo da poterne effettuare dei confronti e delle analisi.

Nel caso di studio d'esempio preso in considerazione, un utente durante la fase di benchmarking inserisce all'interno della propria selezione un insieme di veicoli di cui vuole effettuare il confronto. In particolare tale analisi riguarda un confronto tra il peso dei veicoli ed il loro volume che

consenta di prevedere dove si potrebbe il veicolo target rispetto ai veicoli della competizione.

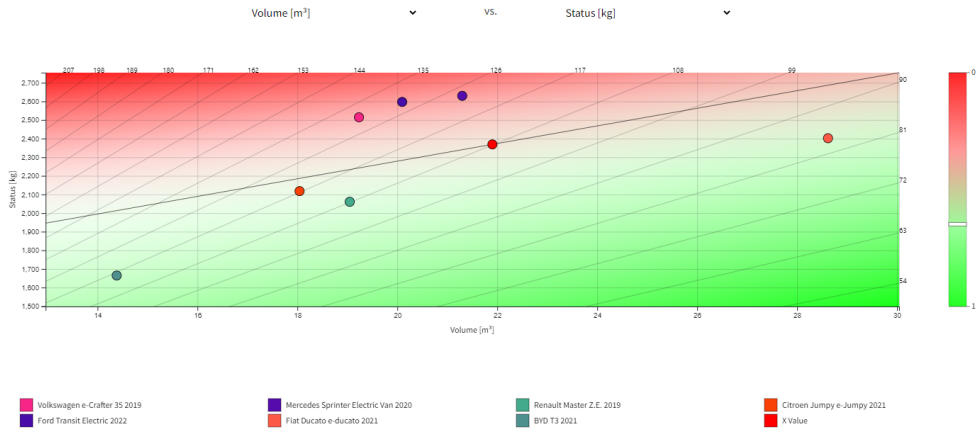


Figura 6.2. Confronto di veicoli

Per tutti i veicoli all'interno della selezione, sia il volume quanto il peso sono memorizzati all'interno del database con la stessa unità di misura, rispettivamente metri cubi e chilogrammi: essendo le features omogenee queste possono essere confrontate tra loro in modo da confrontare tra loro le densità dei veicoli, calcolare l'indice di correlazione tra le misure ed effettuare una previsione sulla densità del nuovo veicolo basandosi ad esempio sulla regressione lineare semplice.

# Capitolo 7

# Conclusioni

L'obiettivo iniziale che il Web Crawler si era prefissato di raggiungere era quello di agevolare e velocizzare il processo di ricerca di un veicolo sul Web, restringendo il campo a un sottoinsieme di fonti affidabili, e di estrazione dei parametri che caratterizzano tale veicolo. Al fine di ottenere una soluzione che rispecchiasse quanto più possibile le necessità operative degli esperti che si occupano di estrarre manualmente le informazioni rilevanti all'interno delle pagine Web, è stato fondamentale analizzare gli aspetti del processo più energivori in termini di tempo in modo da definire le esigenze ed essere in grado di estrapolare dei requisiti funzionali adeguati che guidassero lo sviluppo dell'applicazione.

La scelta di optare per lo sviluppo di una applicazione Web rispetto a una applicazione nativa, è risultata vantaggiosa in termini di portabilità consentendo al Web Crawler di essere accessibile su piattaforme diverse indipendentemente dal loro sistema operativo, offrendo quindi agli utenti un'interfaccia leggera accessibile attraverso un comune browser mentre le logiche più onerose in termini di calcolo vengono eseguite su una macchina server all'interno della quale sfruttare i vantaggi di una applicazione nativa e più vicina al sistema operativo. Un altro vantaggio consiste nella facilità di integrazione e possibilità di comunicazione con il tool aziendale Libra per la condivisione delle features.

L'applicativo Web Crawler nonostante sia stato sviluppato all'interno

di un contesto che riguarda prettamente l'ambito automotive, implementa un processo di estrazione delle informazioni da una pagina Web basato sul parsing di un documento che sfrutta HTML per la formattazione e l'impaginazione del documento stesso. Ciò implica quindi che definendo opportunamente un dizionario delle informazioni da ricercare e le associazioni tra le informazioni riportate all'interno di una pagina web e quelle definite all'interno dell'applicazione, potenzialmente le procedure sono generalizzabili ad altri contesti in cui risulti rilevante estrarre ed analizzare le caratteristiche di prodotti della concorrenza, considerati leader nel proprio settore di mercato, definendo opportunamente il dizionario delle caratteristiche e le associazioni tra gli elementi del dizionario e le caratteristiche così come vengono definite all'interno delle pagine.

# Elenco delle figure

# Listings

# Bibliografia


[1] Trygve m. h. reenskaug. URL https://folk.universitetetioslo.no/trygver/.

[2] Sha-512. URL https://csrc.nist.gov/csrc/media/publications/fips/180/4/final/documents/fips180-4-draft-aug2014.pdf.

[3] Chromium. URL https://www.chromium.org/Home/.

[4] Chromium repository. URL https://chromium.googlesource.com/chromium/src.

[5] Headless chromium. URL https://chromium.googlesource.com/chromium/src/+/lkgr/headless/README.md.

[6] Headless chrome cli. URL https://developer.chrome.com/blog/headless-chrome/.

[7] Puppeteer. URL https://developer.chrome.com/docs/puppeteer/.

[8] Puppeteer repository. URL https://github.com/puppeteer/puppeteer.

[9] Node.js. URL https://nodejs.org/en/.

[10] Puppeteer-sharp. URL http://www.puppeteersharp.com/api/index.html.

[11] Puppeteer-sharp repository. URL https://github.com/hardkoded/puppeteer-sharp.

[12] Angular. URL https://angular.io/.

[13] langdetect. URL https://pypi.org/project/langdetect/.

[14] langdetect repository. URL https://github.com/Mimino666/langdetect.

[15] Google language-detection. URL https://code.google.com/archive/p/language-detection/.

[16] Google language-detection repository. URL https://github.com/shuyo/language-detection.

[17] Codici iso 693. URL https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

[18] Howsearchworks - ranking results. URL https://www.google.com/search/howsearchworks/how-search-works/ranking-results.

[19] Amet s.r.l. URL https://www.amet.it/en/engineering-product-development/.

[20] Libra. URL https://www.amet.it/en/libra/.

[21] R. fielding, architectural styles and the design of networkbased software architectures, phd thesis, 2000, chapter 5. URL https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

[22] L. richardson, s. ruby "restful web services", o'reilly, 2007.

[23] The mit license. URL https://mit-license.org/.

[24] adac. URL https://www.adac.de/.

[25] Argos translate. URL https://www.argosopentech.com/.

[26] Argos translate repository. URL https://github.com/argosopentech/argos-translate.

[27] Argos translate. URL https://opennmt.net/.

[28] D3.js. URL https://d3js.org/.

[29] Angularjs. URL https://angularjs.org/.

[30] Typescript. URL https://www.typescriptlang.org/.

[31] Protocollo di esclusione dei robots. URL https://www.robotstxt.org/.

[32] Token web jwt. URL https://jwt.io/.

[33] Rfc 7519. URL https://www.rfc-editor.org/rfc/rfc7519.

[34] Sql injection attack. URL https://owasp.org/www-community/attacks/SQL_Injection.