Documentazione programma *WebScraping.py*

Che cos'è il programma *WebScraping.py*?

Il programma `WebScraping.py` è un programma in linguaggio `Python` che serve per estrapolare i dati linguistici contenuti nel sito del *"Vocabulario de Comercio Medieval"* per poi salvarli in un file di foglio elettronico che poi potrà essere usato come base per la creazione di un database.

Contenuto

Nella cartella della release sono presenti due programmi:

- `WebScraping.py`: il programma che estrae i dati linguistici.
- `CheckResults.py`: il programma che controlla che il foglio elettronico sia consistente con i dati registrati dal programma.

Oltre ai due programmi sono presenti due cartelle: `data` e `result`. Queste due cartelle contengono reciprocamente **i file per l'esecuzione del programma** e il **foglio elettronico** dove verranno salvati i dati estrapolati.
Più nello specifico nella cartella `data` sono presenti i seguenti file:

- `links.txt`: è un file di testo che contiene l'elenco dei **vocaboli che sono stati salvati nel foglio elettronico**. I dati salvati sono gli indirizzi URL delle pagine che trattano i vocaboli e questi vengono salvati uno per riga.
- `lastrecord.txt`: è un file di testo che contiene il **link dell'ultimo vocabolo che è stato salvato durante la fase di estrapolazione dei dati**. Questo file viene utilizzato per ripristinare la sessione se questa fosse stata interrotta. Ciò avviene durante la fase di ripristino precedente sessione *(vedere più avanti)*.
- `lastsearch.txt`: è un file di testo che contiene il **link dell'ultima pagina ispezionata durante la fase di ricerca**. Questo file viene utilizzato per ripristinare la sessione precedente se questa fosse stata interrotta. Ciò avviene durante la fase di ripristino precedente sessione *(vedere più avanti)*.

- `logerror.txt`: è un file che contiene il registro degli errori che sono avvenuti durante l'esecuzione del programma. I dati vengono scritti uno per riga riportando la **data**, **l'interessato dall'errore** e la **tipologia d'errore**.

Più nello specifico nella cartella `result` è presente il seguente file:

- `data.xlsx`: è un foglio elettronico che contiene tutti i dati estratti dal programma `WebScraping.py` sotto forma di tabella. I dati estratti vengono salvati per riga e sono: **vocabolo**, **definizione**, **documenti referenziati dall'archivio del professor Gual**, **documenti referenziati dalla fondazione J. March**, **numero di documenti referenziati**, **URL** della pagina e **data di salvataggio**.

Struttura del programma

Librerie utilizzate

Per il suo funzionamento vengono usate le seguenti librerie:

- `BeautifulSoup` o `bs4` - per la manipolazione del codice `HTML` delle pagine web.
- `Requests` - per la connessione ad Internet.
- `Datetime` - per la gestione delle date.
- `Openpyxl` - per la gestione dei file `.xlsx`.
- `System` - per accedere alle funzionalità specifiche del sistema.
- `Os` - per la gestione delle *directory.*
- `Msvcrt` - per l'accesso ad alcune funzioni specifiche.

Funzioni utilizzate

Sono state implementate le seguenti funzioni *(verranno approfondite più avanti)*:

- `test`.
- `initialize`.
- `searching`.
- `scraping`.

- `table_write`.
- `update`.
- `swap`.
- `save`.
- `logerror`.
- `remove_link`.

Come funziona?

Il funzionamento del programma si può suddividere in *cinque* macro-funzioni:

- **Modalità test**.
- **Ripristino precedente sessione**.
- **Ricerca**.
- **Estrapolazione dati**.
- **Apertura link che hanno avuto un errore di connessione**.
- **Aggiornamento dei vocaboli**.

Ogni macro-funzione viene eseguita dopo la precedente **ad eccezione** della modalità di test che è opzionale e che può essere saltata.

Modalità test

Che cosa fa?

La modalità di test è opzionale e può essere avviata dall'utente. Durante questa fase, il programma esegue tutte le macro-funzioni ma con un campione ridotto di vocaboli da processare. Questa modalità è utile per dimostrazioni o test. I file `links.txt`, `lastsearch.txt`, `lastrecord.txt` e `data.xlsx` vengono **cancellati** durante la modalità di test.

Come funziona?

Dopo l'avvio del programma, all'utente viene chiesto se vuole avviare questa modalità. Se sì, digita "***y***". Successivamente, i file elencati vengono cancellati e il programma entra in modalità di ricerca *(il ripristino precedente sessione **non** viene effettuato)*. La funzione `initialize` viene invocata, seguita dalla funzione `test`, che ridimensiona la variabile `lett` a 15 elementi. Il programma continua come prima, ma con un campione solitamente di 300 elementi.

Ripristino precedente sessione

Che cosa fa?

Il ripristino della precedente sessione è la *prima* parte del programma `WebScraping.py`. Qui si controlla se ci sono sessioni precedenti incomplete controllando il contenuto dei file `lasrecord.txt` e `lastsearch.txt`. Questa fase permette di riprendere il lavoro dal punto in cui era stato interrotto in caso di arresto del programma, evitando di perdere l'intera sessione precedente.

Come funziona?

Per ripristinare la precedente sessione del programma, vengono eseguiti i seguenti passaggi:

1. Si estrae l'elenco dei link dei vocaboli già presenti nel database dal file `data.xlsx`, se presente, e li salva nella variabile `old_link`.
2. Si controlla il contenuto di `lastrecord.txt`. Se il file contiene un link e l'utente desidera continuare la sessione precedente, il programma procede all'estrazione dei dati e continua con il salvataggio dei record a partire dal successivo a quello indicato dal file. Quindi apre il file `links.txt`, cerca l'elemento contenuto in `lastrecord.txt` e lo salva nella variabile `lastrecord`. Successivamente, copia nella variabile `link_risultati` i valori di `links.txt` dall'indice di `lastrecord` fino alla fine attraverso un ciclo `for`. Infine, salta direttamente alla fase di estrazione dei dati. In caso contrario, il programma cancella il contenuto di `lastrecord.txt`, elimina tutti i link contenuti in `links.txt` che non sono stati salvati e procede normalmente.
3. Se `lastrecord.txt` è vuoto, il programma controlla il contenuto di `lastsearch.txt`. Se contiene un link e l'utente desidera continuare con la sessione precedente, il programma recupera i risultati della precedente ricerca e imposta la nuova in modo che cominci dal punto in cui era stata interrotta. Quindi salva nella variabile

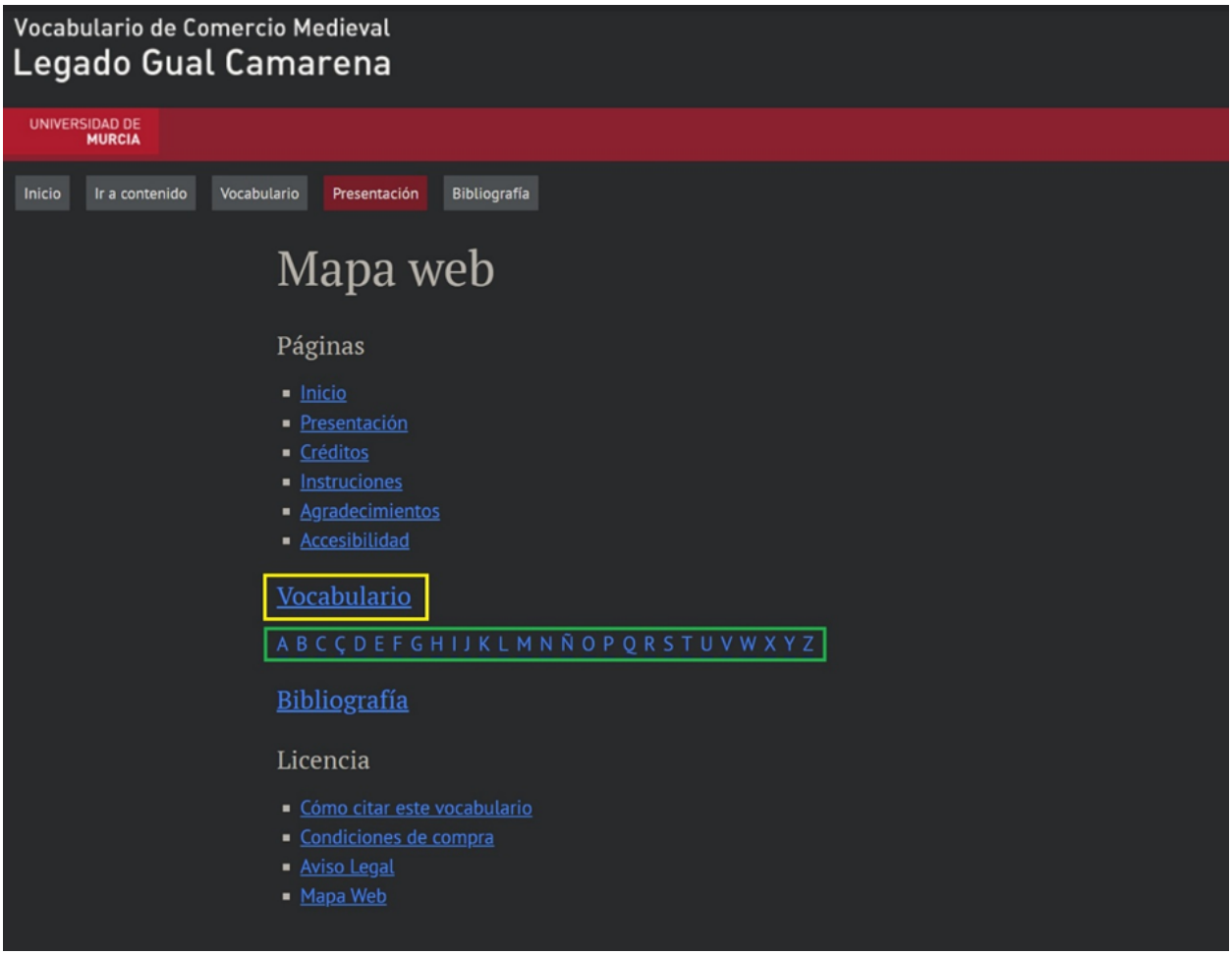
`lastsearch` il contenuto del file `lastsearch.txt` e salva i nuovi vocaboli che erano già stati trovati con la precedente ricerca in `link_risultati`. Dopo la chiamata alla funzione `initialize`, rimuove le pagine che sono già state analizzate attraverso l'esecuzione di un ciclo `for` che cancella gli elementi precedenti a `lastsearch` contenuti nella variabile `lett`.
In caso contrario, il programma cancella il contenuto di `lastsearch.txt` e cancella da `links.txt` tutti le voci che non sono state salvate. Infine, il programma procede normalmente.

4. Se entrambi i file sono vuoti, il programma procede normalmente.

Ricerca

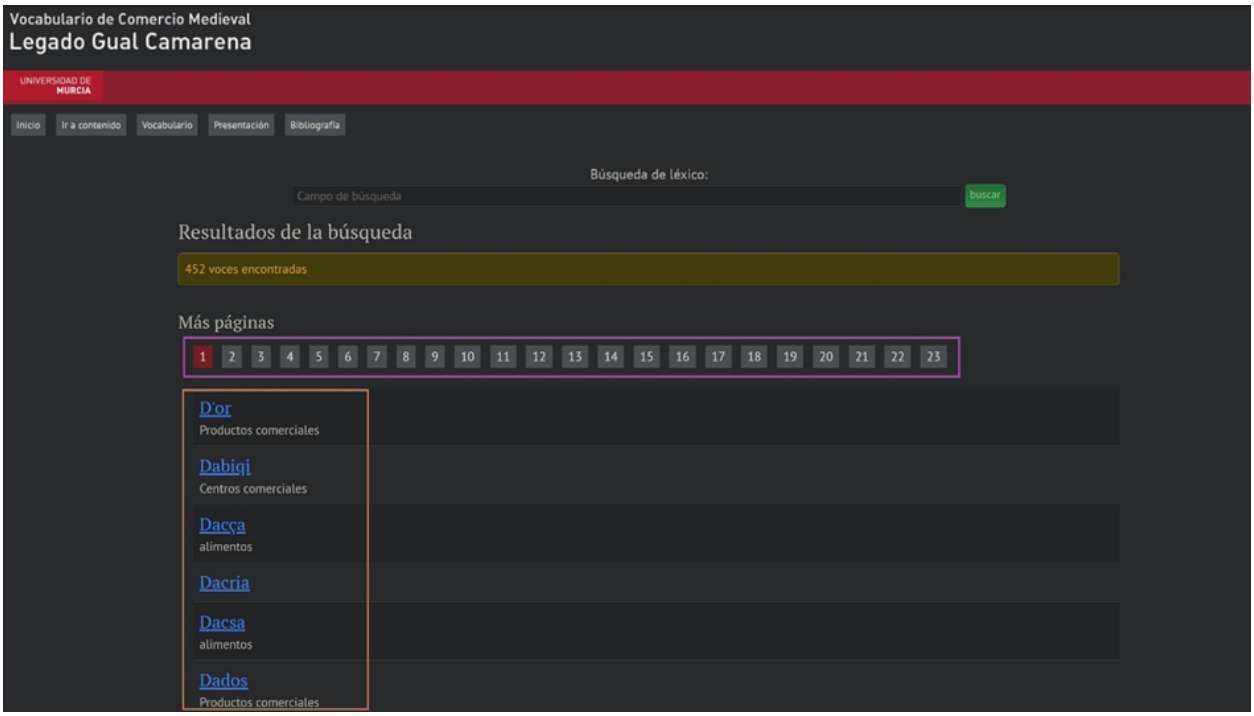
Che cosa fa?

La ricerca è la *seconda* parte del programma. Qui vengono recuperati i link alle pagine sui vocaboli del commercio medievale. Vengono utilizzate le funzioni `initialize` e `searching`. I link ai vocaboli vengono raccolti dalla mappa del sito, disponibile all'indirizzo *https://www.um.es/lexico-comercio-medieval/index.php/mapaweb*. Sotto la sezione "*Vocabulario*" (in giallo nella figura 3.1) ci sono le lettere dell'alfabeto che separano i vocaboli trattati dal sito (in verde nella figura 3.1). Il programma esamina ogni lettera e recupera l'elenco delle pagine (in viola nella figura 3.2).



[figura 3.1 - Mapa Web]

Per ogni pagina vengono estratti i link relativi ad un dato vocabolo (evidenziato in marrone in figura 3.2) che vengono poi salvati nella variabile `link_risultati` e nel file `links.txt`.



[figura 3.2 - Vocaboli divisi per lettera]

Come funziona?

La ricerca inizia con la chiamata alla funzione `initialize` che recupera le pagine del vocabolario e li salva nella variabile `lett`. Successivamente, viene invocata la funzione `searching` per recuperare i vocaboli presenti in una pagina (è un elemento di `lett`) e li salva nella variabile `res`. Infine, viene controllato se ogni elemento in `res` è presente in `old_link` o in `link_risultati`. Se non è presente in nessuna delle due, il vocabolo viene aggiunto al file `links.txt` e a `link_risultati`. La fase di ricerca può essere interrotta premendo "***q***". In caso di interruzione, il programma salva l'ultima ricerca in `lastsearch.txt` e termina. Se il server non risponde correttamente, il programma salva l'ultima ricerca in `lastsearch.txt` e termina. Durante il processo, viene mostrata la percentuale di progressione calcolata sulla dimensione della variabile `lett`.

Estrapolazione dati

Che cosa fa?

La fase di estrapolazione dati è la *terza* parte del programma `WebScraping.py`. Qui, vengono estratti i dati linguistici dei vocaboli trovati durante la ricerca. Le funzioni `scraping` e `table_write` vengono utilizzate in questa sezione. I dati estratti per ciascun

vocabolo includono il **vocabolo** stesso, la sua **definizione**, la **tipologia**, i **documenti del professor Gual** e della **fondazione J. March** che sono stati referenziati dal vocabolo, il **numero** totale di **documenti referenziati**, l'**URL** della pagina e la **data di salvataggio** del vocabolo nel foglio elettronico. Questi dati vengono salvati nel foglio elettronico `data.xlsx` nella cartella `result`.

Come funziona?

La fase di estrapolazione dati comincia con il controllo della presenza del file `data.xlsx` nella cartella `result`. Se il file non è presente, il programma ne creerà uno nuovo e lo caricherà. Dopo aver recuperato l'indice di riga ed aver rinominato il foglio di lavoro con la data corrente, il programma estrae i dati dei vocaboli contenuti nella variabile `link_risultati`. Quindi, invoca la funzione `scraping` per estrarre i dati dalla pagina web e la funzione `table_write` per effettuarne il salvataggio. Infine, viene salvato il link appena esaminato sul file `lastrecord.txt` tramite la funzione `save` per poter ripristinare la sessione in caso di interruzione.
Si può terminare anticipatamente la fase di estrapolazione dati inserendo da tastiera il carattere "***q***". L'interruzione viene gestita attraverso due `if` che controllano ad ogni iterazione del ciclo se è stato inserito il carattere "***q***". In caso di arresto anticipato, il programma salva il file `data.xlsx` e chiude il programma.
Durante la fase possono verificarsi **tre** diverse eccezioni ed in base all'eccezione lanciata, il programma si comporta in modo diverso:

- `requests.exceptions.ConnectTimeout`: il link interessato viene salvato nella variabile `link_error` per essere processato successivamente.
- `BeautifulSoupException`: l'errore viene registrato nel file di *log* e il link interessato viene rimosso tramite la funzione `remove_link`.
- `requests.exceptions.RequestException`: il link interessato viene salvato nella variabile `link_error` per essere processato successivamente.

Durante il processo viene stampata a video la progressione, calcolata sulla dimensione della variabile `cont_res`.

Apertura link che hanno avuto un errore di connessione

Che cosa fa?

L’apertura dei link che hanno avuto un errore di connessione costituisce la *quarta* parte del programma `WebScraping.py`. Essa consiste nell'apertura dei link che avevano avuto problemi di connessione nella fase precedente. Questa fase non viene sempre eseguita e comporta l'estrapolazione dei dati dei link interessati in modo analogo alla fase precedente.

Come funziona?

Inizialmente, i link contenuti nella variabile `link_error` vengono rimossi dal file `links.txt`. Successivamente, vengono recuperati i dati linguistici in modo analogo alla fase di estrapolazione dei dati, con l'eccezione che **non** viene più salvato in `lastrecord.txt` l’ultimo elemento processato. Se si riscontra un `Connection Timeout` o un `RequestException`, il link interessato viene spostato in coda a `link_error` per essere riprocessato successivamente. Se l'estrapolazione dei dati va a buon fine, i dati estratti vengono salvati e il link viene aggiunto a `links.txt`. Infine, il link appena salvato viene rimosso da `link_error`. Il numero di errori ammessi è limitato dal valore della variabile `maxfail` (che è uguale a due volte e mezzo la lunghezza di `link_error`). Il ciclo continua fino a quando tutti i link in `link_error` vengono estrapolati correttamente o il numero di tentativi falliti supera quello di `maxfail`. Alla fine, eventuali link ancora presenti in `link_error` vengono persi. Durante il processo, viene visualizzata a video la progressione, calcolata sulla dimensione della variabile `c`.

Aggiornamento dei vocaboli

Che cosa fa?

L'aggiornamento dei vocaboli costituisce l'*ultima* fase del programma `WebScraping.py`. Per controllare se un vocabolo debba essere aggiornato si confronta la data di salvataggio con la data di modifica riportata nel sito. Così facendo è possibile determinare se una voce necessiti di un aggiornamento o meno.

Come funziona?

Il programma inizia leggendo il file `data.xlsx` e recuperando per ogni vocabolo la coppia di valori **URL** e la **data di salvataggio**. Questi valori vengono salvati in una `tupla` e aggiunti a `links`. In questa fase il programma controlla anche se sono presenti righe vuote ed eventualmente le segnala all’utente riportando l’errore nel file `logerror.txt`. Successivamente, per ogni elemento della lista `links`, il programma chiama la funzione

`update` per aggiornare i record, escludendo quelli appena salvati. Al termine dell'aggiornamento, il programma salva il file `data.xlsx` e termina l'esecuzione. Si può interrompere la fase di aggiornamento premendo il tasto "***q***". In caso di interruzione anticipata, il programma salva il file `data.xlsx` con i record aggiornati e chiude il programma. Durante il processo di aggiornamento, il programma mostra la percentuale di avanzamento che viene calcolata sulla lunghezza della variabile `links`.

Funzioni

Funzione *test*

Che cosa fa?

Questa funzione effettua il ridimensionamento della variabile `lett` che contiene l'elenco delle pagine del vocabolario dove effettuare la ricerca dei vocaboli.

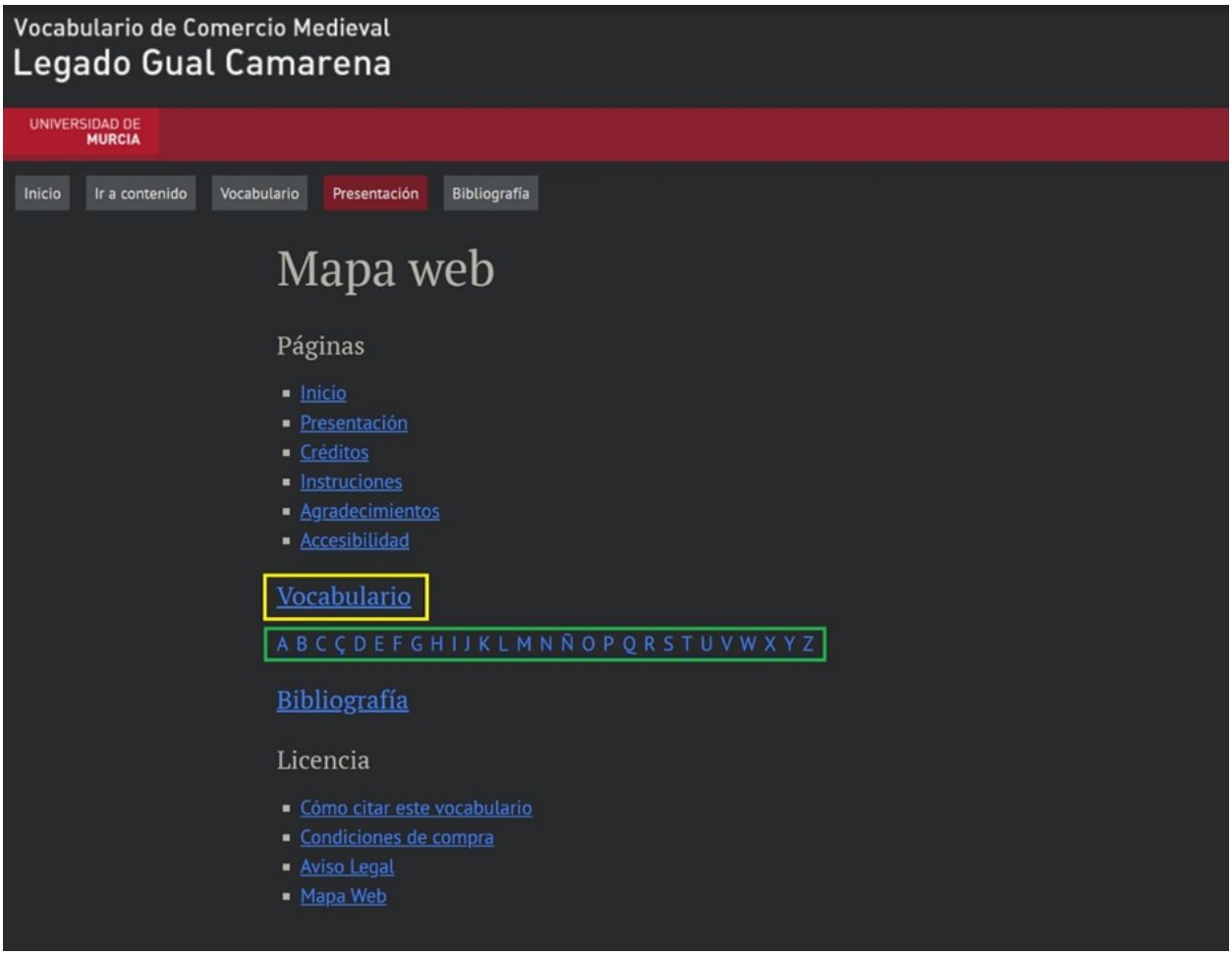
Come funziona?

La funzione, attraverso un ciclo `while`, elimina in modo casuale un elemento dalla variabile `lett`. Il ciclo continua finché la lunghezza di `lett` non è uguale a `num` (che nel programma è stato messo a **15**). Al termine viene restituita `lett` al chiamante.

Funzione *initialize*

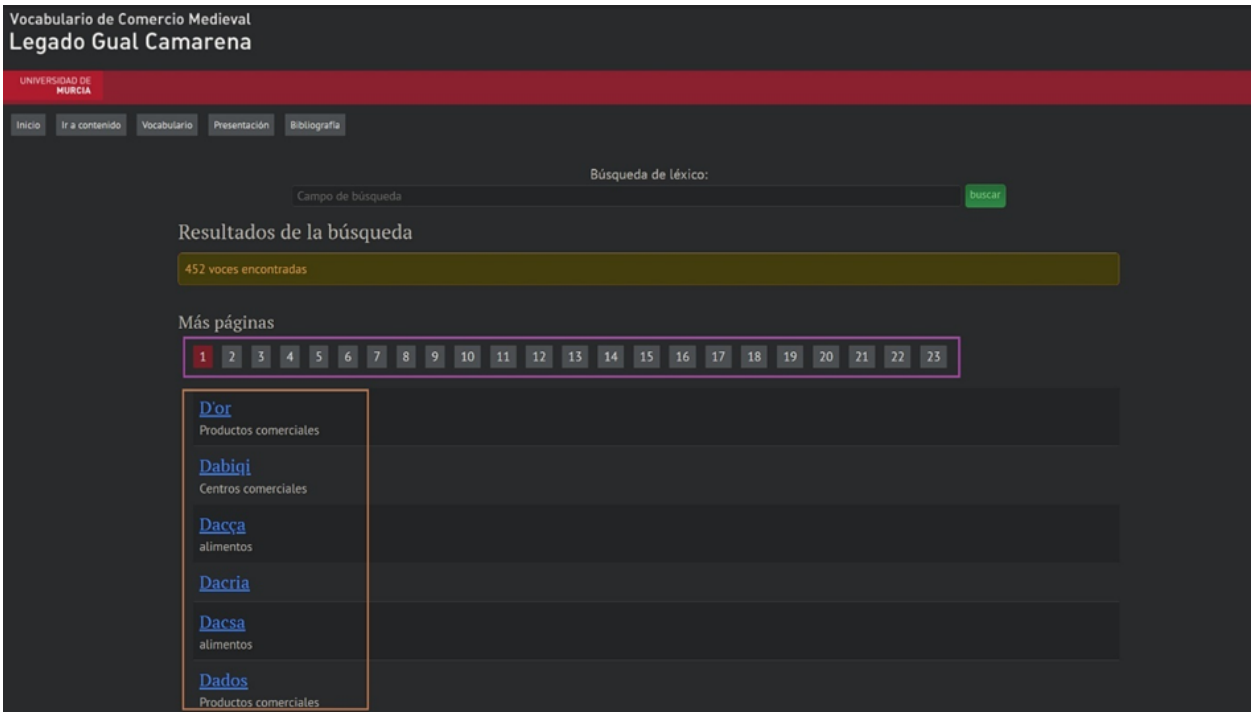
Che cosa fa?

La funzione recupera tutte le pagine che contengono i vocaboli divisi per lettera iniziale. Queste sono reperibili all'indirizzo *https://www.um.es/lexico-comercio-medieval/index.php/v/letra/LETTERA* raggiungibili dalla pagina *https://www.um.es/lexico-comercio-medieval/index.php/mapaweb* sotto la voce "*Vocabualrio*" (in figura 4.1 in verde).



[Figura 4.1 - Mapa web]

Ogni pagina contiene l’elenco di tutte le parole che sono presenti nel sito con quella lettera iniziale (in figura 4.2 in marrone).



[figura 4.2 - Vocaboli divisi per lettera]

Come funziona?

La funzione inizialmente si collega alla pagina *https://www.um.es/lexico-comercio-medieval/index.php/mapaweb* per poi estrapolarne il contenuto selezionando la porzione delimitata dal tag `<ul class="diccionario">`. Dopodiché recupera tutti i tag `href` (ovvero tutti i collegamenti ipertestuali) contenuti nella sezione evitando però di copiarne i doppioni.
Successivamente, per ogni link trovato, si collega alla pagina corrispondente e ne estrapola tutti i tag `href` delle pagine di navigazione. Queste le salva, sempre evitando di copiarne i doppioni, su una variabile denominata `res` che poi viene restituita al chiamante.

Funzione *searching*

Che cosa fa?

La funzione `searching` recupera tutti i link ai vocaboli contenuti nelle pagine recuperate attraverso la funzione `initialize`.

Come funziona?

La funzione si collega alla pagina indicata dal parametro di tipo stringa `sh_link` (questo parametro conterrà un elemento trovato dalla funzione `initialize`).
Dopo aver ottenuto il contenuto della pagina, la funzione ricerca l'elenco dei vocaboli nella porzione indicata dal tag `<article class="resultados">`. Quindi seleziona il secondo tag `ul` e ne estrae tutti i collegamenti ipertestuali che li salva, sempre omettendo i doppioni, in una variabile denominata `res` che viene poi restituita al chiamante.

Funzione *scraping*

Che cosa fa?

La funzione `scraping` serve per estrapolare i dati linguistici delle pagine web dei vocaboli.

Come funziona?

La funzione inizialmente si collega alla pagina indicata dal parametro `sr_link` bloccandone i reindirizzamenti. Poi viene controllato che lo status code della risposta del server sia 200 (`HTTP 200 OK`) e che quindi il collegamento al server sia stato eseguito correttamente. In caso di status code diverso viene lanciata l'eccezione `BeautifulSoupException`.
Dopo aver ottenuto il contenuto della pagina, la funzione procede estraendo il **vocabolo**, la **definizione** e il **tipo**. In seguito, estrapola i **documenti referenziati** tenendo conto del numero di tag `alt` che processa attraverso la variabile intera `num_doc`. Infine, salva l'**URL** della pagina (che è il parametro `sr_link`) e la data corrente (sfruttando la libreria `datetime`) in formato *dd/mm/yy*.
Per evitare di eseguire un'operazione di estrapolazione su un tag inesistente (in quanto non tutte le pagine presentano per forza tutti i parametri richiesti) viene controllato sempre che il risultato della selezione del tag sia diversa da `None`. In caso contrario quel dato viene ignorato e verrà salvata una stringa vuota al suo posto.
Tutti i dati vengono salvati nella variabile denominata `res`, di tipo lista, che viene poi restituita al chiamante.

Funzione *table_write*

Che cosa fa?

La funzione `table_write` serve per trascrivere i dati estrapolati dalla funzione `scraping` nel foglio elettronico `data.xlsx`.

Come funziona?

La funzione, attraverso un ciclo `for`, trascrive ogni elemento contenuto in `res` nel foglio `data.xlsx`. La posizione della scrittura viene determinata dalla coppia `col` (una variabile interna alla funzione inizializzata ad 1) e `r` (variabile passata come argomento).
Ad ogni iterazione del ciclo la variabile `col` viene incrementata di 1 permettendo così di scrivere i dati in riga.

Funzione *update*

Che cosa fa?

La funzione `update` serve per verificare che i dati salvati nel foglio elettronico `data.xlsx` siano uguali a quelli presenti nel sito. È presente la dicitura che dice se il vocabolo è stato modificato o meno e se sì in quale data. Confrontando questa data con la data di salvataggio del record, la funzione determina se un vocabolo debba esser aggiornato.

Come funziona?

La funzione inizialmente si collega alla pagina indicata dal parametro `up_link`. Avendo contenuto differente, la dicitura di modifica del vocabolo non sempre è reperibile nella stessa posizione all'interno della pagina. Di conseguenza la funzione controlla in *due* diverse posizioni:

- il primo tag `i` contenuto nella sezione `<span class='tipo'>`.
- Il primo tag `i` dopo il secondo tag `p` contenuto nella sezione `<p class='descripcion'>`.

Una volta ottenuto il contenuto della stringa procede al controllo della data. Qui viene confrontata la data estrapolata (divisa in giorno, mese ed anno e ricostruita come oggetto di classe `datetime` chiamata `dtonline`) con la data passata come parametro `rdate`. Se la data indicata da `rdate` è antecedente a quella di `dtonline`, allora si procede con una nuova estrapolazione dei dati richiamando la funzione `scraping`. Infine, si richiama la funzione `table_write` che ne sovrascriverà i dati. La sovrascrittura dei dati viene fatto grazie alla variabile `row` che contiene il numero di riga dov'è salvato il vocabolo nel foglio elettronico.

Funzione *swap*

Che cosa fa?

La funzione `swap` serve modificare l'ordine dei record contenuti in `links.txt`. Essa sposta in fondo al file di testo l'elemento passato come argomento.

Come funziona?

La funzione inizialmente recupera tutti gli elementi contenuti in `links.txt` e li mette in una variabile di tipo lista denominata `old_links`. Dopodiché inserisce in coda alla lista il link indicato da `sw_link` effettuando prima la sua rimozione e poi il suo reinserimento. Infine, sovrascrive `links.txt` con gli elementi di `old_links`.

Funzione *logerror*

Che cosa fa?

La funzione `logerror` serve a salvare nel file `logerror.txt` gli errori che sono stati riscontrati durante il funzionamento del programma. Questi vengono trascritti come:

[giorno-mese-anno]: "oggetto_interessato_dall'errore" – Tipologia errore: "errore".

Come funziona?

La funzione apre il file `logerror.txt`, salva la data corrente (attraverso la libreria `datetime`) e compone il messaggio d'errore (come descritto nell'introduzione) usando le due variabili passate per argomento. Queste due variabili, entrambe di tipo stringa, contengono reciprocamente l'oggetto dell'errore (`er_link`) e la tipologia d'errore (`exception`).

Funzione *remove_link*

Che cosa fa?

La funzione `remove_link` serve per rimuovere un record contenuto in `links.txt`.

Come funziona?

La funzione apre in lettura il file `links.txt` e ne salva il contenuto in una variabile denominata `filedata`. Poi sostituisce l'elemento contenuto nella variabile `rm_link` più il

carattere a capo `\n` con una stringa vuota. Infine, sovrascrive il contenuto di `links.txt` con quello contenuto in `filedata`.

Eccezioni

Oltre alle normali eccezioni presenti nelle librerie precedentemente elencate è stata costruita una eccezione appositamente per questo programma.

BeautifulSoupException

A cosa serve?

`BeautifulSoupException` è una eccezione che è stata creata per poter essere invocata quando un link che viene analizzato dal programma non rispetta i parametri voluti.
