Politecnico di Torino

Corso di Laurea
A.a. 2020/2021
Sessione di Laurea Luglio 2021

Web crawling e analisi semantica per il supporto all'innovazione del business

Relatori:
Gatteschi Valentina
Lamberti Fabrizio
Demartini Claudio Giovanni

Candidati:
François Matilde

SOMMARIO

INDICE DELLE FIGURE

1. Introduzione

Lo scopo principale degli algoritmi di web crawling e web scraping, protagonisti del presente lavoro di tesi, è quello di raccogliere i dati non strutturati all'interno del web e gestirli in modo da potersi adattare al maggior numero di scenari possibili. A tal proposito, come si vede nella maggior parte dei casi proposti in letteratura, le tecniche di web scraping e web crawling sono state customizzate e rese procedure ad hoc che fossero in grado di gestire sistemi completamente automatizzati e di convertire interi siti-web in set di dati ben organizzati.

Il progetto di tesi si presenta come lo sviluppo di un software in linguaggio Python, svolto a partire da dati di input forniti dall'ente collaborante Fondazione Torino Wireless. Uno dei core business della Fondazione è quello di gestire un network di imprese mettendole in contatto tra loro per eventuali progetti, lavori o collaborazioni. Il software implementato ha lo scopo di fornire all'ente collaborante l'opportunità di valutare un sistema per scaricare un set di dati relativi al network di aziende presenti all'interno della sua rete. In questo modo sarà possibile supportare i data analyst nella ricerca delle prime informazioni riguardo la profilazione di aziende di interesse, riducendo i tempi e migliorando l'efficacia di una ricerca puntuale e manuale sui siti web delle singole imprese. Queste informazioni sono utili agli analisti non solo per mantenere aggiornato il profilo delle aziende già facenti parte del network, ma anche per fare una ricerca mirata di imprese non ancora conosciute ma con un profilo allineato con gli obiettivi della ricerca. Il secondo obbiettivo di questo progetto è quello di fornire una soluzione per un problema di machine learning relativo alla classificazione delle aziende appartenenti alla rete di Fondazione Torino Wireless in base ai settori in cui esse si collocano.

Il presente lavoro di tesi è strutturato nel seguente modo:

1. Nel secondo capitolo si descrive il contesto in cui si collocano le tecniche web crawling e web scraping nonché le relative strutture e funzionamenti.

2. Nel terzo capitolo vengono illustrate tutte le tecniche principali che vengono usate per implementare gli algoritmi di web crawling e web scraping.
3. Nel corso del quarto capitolo si entra nel cuore del progetto ed è quello dove viene illustrato e descritto lo sviluppo delle varie fasi del software sviluppato.
4. Nel quinto capitolo si illustrano le conclusioni a cui si è giunti alla fine del progetto e le possibili prospettive future che si possono implementare a partire dall'intero lavoro svolto.

2. Web scraping e web crawling

“Nel ventunesimo secolo i dati sono diventati il nuovo petrolio” [1]. Con il passare degli anni si stanno affermando molte realtà aziendali che fondano il proprio obbiettivo di business sulla conoscenza e sulla corretta acquisizione di essa. La base della conoscenza in un mondo ormai molto legato alla sfera dell’internet è l’acquisizione di dati di interesse che siano più o meno sensibili. Una delle fonti principale di raccolta dati è il web, come suggerisce la Figura **1**.

FIGURA 1-WEB SCRAPING E WEB

L’operazione di estrazione dei dati da questa fonte può però risultare spesso troppo dispendiosa in termini di tempo e di forza lavoro impiegata per la sua implementazione, si pensi semplicemente se qualcuno dovesse cercare ogni volta dei dati all'interno del web e, per acquisirli, si dovesse copiare e incollare la parte d'interesse. Le tecniche di web crawling e web scraping si collocano in questo contesto con l’obbiettivo di rendere il processo di estrazione dei dati più scalabile. Nelle sezioni che seguono, tali tecniche e le relative caratteristiche verranno descritti dettagliatamente.

2.1 Web Scraping

Il web scraping è una forma di data mining e, come suggerisce la traduzione letterale (grattar via, raschiare), è una tecnica che permetta l'estrazione dei dati non strutturati dai diversi siti web, per mezzo di programmi software, che

possono essere successivamente analizzati e salvati all'interno di un database. Tale processo è tanto più efficace quanto più è automatizzato. L'intero funzionamento di uno scraper può essere diviso in due fasi sequenziali [2].

1) Acquisizione delle risorse web:

   Un software di web scraping inizia effettuando una richiesta HTTP per raccogliere le risorse da una pagina web d'interesse. Il tipo di richiesta può variare, può essere una richiesta di tipo GET contenente l'URL del sito al quale si vuole accedere o può essere una richiesta di tipo POST, quest'ultima è usata principalmente quando si vuole inviare dei dati al server. Una volta che il sito web ha ricevuto la richiesta con successo invierà una risposta al software che, se correttamente ricevuta (indicato dal codice di stato dell'oggetto requests.Response() uguale a 200), la gestirà nel modo che ritiene più opportuno. Le risorse all'interno della risposta HTTP possono essere in formati differenti poiché le pagine web possono essere costruite con diversi linguaggi di programmazione. La struttura di questa fase è descritta in Figura **2**.

FIGURA 2 – COMUNICAZIONE CLIENT-SERVER

2) Estrazione dell'informazione:

   Dopo la fase di acquisizione il software si occupa dei dati grezzi ricevuti e di gestirli, facendo tutte le operazioni necessarie per renderli strutturati e pronti per raggiungere lo scopo di business prefissato. Per portare un esempio, si potrebbe essere interessati esclusivamente ad ottenere delle tabelle presenti su uno specifico sito, quindi il codice dovrà essere in grado di escludere tutto il resto delle informazioni presenti sulla pagina designata. La struttura di questa fase è descritta in Figura **3**, reperita grazie a [1].

FIGURA 3- ARCHITETTURA WEB SCRAPER

Il web scraping viene utilizzato per una grande varietà di scenari. Per esempio, un campo di applicazione in cui viene spesso utilizzato è quello del monitoraggio dell'andamento dei prezzi presenti su qualsiasi sito web di e-commerce oppure quello per cui si raccolgono i commenti sui social media riguardo un determinato argomento per avere delle informazioni riguardanti l'opinione pubblica rispetto ad un determinato evento. Ad un livello più alto sono raccolti i metadati di ogni sito per costruire un sempre più efficiente motore di ricerca.

2.1.2 Questioni legali

Nonostante la frequenza con cui ormai si usino le varie tecniche di web scraping e web crawling per raccogliere informazioni sul web ancora non vi è una legislazione o un codice etico ben definito per regolare l'uso di tali algoritmi. Questa ignoranza potrebbe risultare molto scomoda poiché risulta impossibile per un programmatore costruire un programma di web scraping volto alla ricerca di informazioni utili al suo obbiettivo senza essere sicuro di poter incorrere in controversie, etiche o legali, legate al suo lavoro, che potrebbero, oltre a comportare dispendiose cause legali, compromettere la sua reputazione e rendere inutile tutto il lavoro svolto. Da una ricerca effettuata prendendo dati pubblici dal web potrebbero scaturire questioni etiche legate alla compromissione della privacy degli individui a cui si riferiscono quei dati o della riservatezza legata all'uso di informazioni confidenziali legate alla partecipazione di organizzazioni in attività descritte in un sito web. A tal proposito il sito web potrebbe essere compromesso e non visto più "sicuro" per chi accetta di condividere al suo interno le proprie informazioni. Come ci è reso noto dallo studio [3] molti quadri giuridici sono stati applicati in casi di controversie legate all'uso dei dati presi dal web portati in tribunale. Tra questi vi sono:

- Accesso e uso illegale dei dati

Il CFAA (Computer Fraud and Abuse Act) è un insieme di leggi che compongono la legislazione che regola il campo della sicurezza informatica. Tale legge vieta l'accesso ad un computer senza autorizzazione o in eccesso di essa e prevede sanzioni sia civili che penali in base alla gravità dell'illecito. Fino ad oggi le disposizioni legali riguardo l'applicazione del CFFA alle tecniche di web scraping non erano chiare. Adesso, vi sono casi recenti da cui è possibile dedurre informazioni più chiare. Si è stabilito che l'attività di web scraping è possibile che violi i "Termini d'uso" o "Termini di servizio" di un sito web. Tuttavia, è stato altresì stabilito che questa violazione da sola non comporti la base per una controversia per il CFAA. Tuttavia, da casi successivi in America, si è stabilito che il web scraping non fosse punibile ai sensi del CFAA neanche nelle situazioni in cui l'accesso al sito web venga revocato a chi sta eseguendo il web scraping su dati accessibili pubblicamente tramite una lettera di diffida e un blocco IP.

- Violazione del contratto

  Anche questo tipo di contenzioso si basa sul contratto dei "Termini di servizi" o "Termini di utilizzo" in cui il proprietario del sito web può specificare il divieto di accesso automatizzato al suo sito. La violazione di questi termini può portare appunto ad una violazione contrattuale ai danni di chi sta estraendo dati dal sito. Tuttavia, la responsabilità del programmatore sussiste solo nel momento in cui egli accetta esplicitamente i "Termini d'uso" quindi se questo procedimento non viene posto in essere, dal punto di vista legale, l'attività di web scraping non è passibile di denuncia. A questo si aggiunge il fatto per cui insieme alla violazione dei termini di servizio il proprietario del sito web deve anche dimostrare che egli ha subito dei danni a seguito della attività di estrazione dei dati violando il contratto stipulato attraverso l'accettazione dei termini di servizio.

- Uso di materiale protetto da diritto d'autore

  Il diritto d'autore protegge giuridicamente la rappresentazione e la forma specifica di una idea. Nel caso in cui essa venga copiata e replicata allora potremmo essere passibili di denuncia. Da ciò ne deduciamo che la sola attività di scraping su dei dati coperti da copyright non può cadere vittima di un caso di violazione del diritto d'autore. Il primo fatto che deve

sussistere è quello per cui tali dati devono essere replicati e riutilizzati allo stesso modo e inoltre, da questo uso, ne deve scaturire un guadagno finanziario per l'utilizzatore dello scraper. A questo si aggiunge il principio del "fair use" per il quale è possibile riutilizzare dei dati coperti da copyright in scala limitata per un uso originale di essi.

- Appropriazione indebita

  L'appropriazione indebita si manifesta nel momento in cui dall'attività di web scraping scaturisce un danneggiamento o sovraccaricamento del sito bersaglio dello scraper. La violazione del web scraper diventa passibile di denuncia con un conseguente risarcimento del danno causato quando il danno è facilmente monetizzabile. La materialità del danno, tuttavia, non si riesce a dimostrare in molti casi quindi questo tipo di violazione non è molto frequente nei casi giuridici inerenti al web scraping.

- Violazione del segreto commerciale

  Un segreto commerciale viene trattato dal tribunale come se fosse un'immobilizzazione immateriale e quindi è protetto dalla stessa legislazione che si applica per i brevetti e i marchi di una impresa. Le attività protette da segreto industriale sono tutte "*le informazioni aziendali e le esperienze tecnico-industriali, comprese quelle commerciali, che siano "segrete" (nel senso che non siano nel loro insieme o nella precisa configurazione e combinazione dei loro elementi generalmente note o facilmente accessibili agli esperti ed agli operatori del settore), che abbiano "un valore economico" in quanto segrete e che "siano sottoposte a misure da ritenersi ragionevolmente adeguate a mantenerle segrete*". Da questa definizione se ne deduce che se l'oggetto dell'attività di scraping non soddisfa questa definizione allora l'attività stessa non potrà essere passibile di denuncia.

2.2 Web crawling

Il web crawler è un software anche conosciuto come robot o spider con lo scopo di navigare il web in modo automatico e ricorsivo. Il crawler spesso, nella sua accezione e nel suo utilizzo più generale è la base della costruzione di un motore di ricerca che generalmente si compone di tre parti: un crawler, un indicizzatore e un algoritmo di classificazione della ricerca per riuscire a rispondere in modo

sempre più ottimale alle richieste dei suoi utilizzatori. L'architettura base di un algoritmo di web crawling prevede, come indicato in Figura **4**, che a partire da uno o più URL di partenza, chiamati semi, il crawler scarichi le pagine associate e le salvi in locale, prenda tutti gli hyperlink di quelle pagine e li inserisca in una coda, controllando che essi non siano già presenti al suo interno e scegliendo un ordine preciso in modo da poter allargare il più possibile il campo di ricerca delle pagine web. Gli hyperlink presenti all'interno della coda serviranno al crawler per continuare ricorsivamente il processo inziale fino a che la coda non sarà vuota.

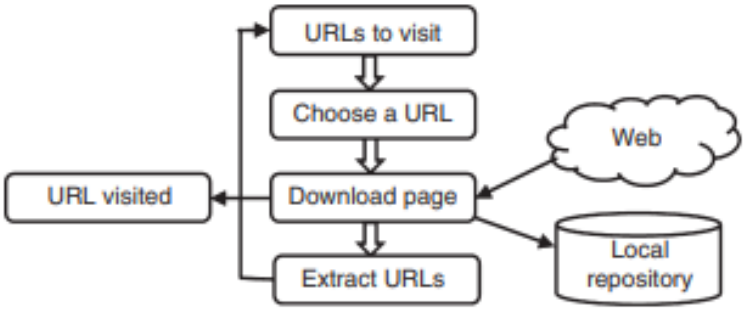

FIGURA 4 – ARCHITETTURA WEB SCRAPING

Questa semplice architettura può essere integrata in base al funzionamento che il crawler deve avere per uno specifico obbiettivo.

2.2.1 Tipologie di crawler

All'interno della letteratura non vi è una precisa suddivisione di tutti i tipi di crawler ma lo studio riportato in [4] ci fornisce, tramite la Figura **5**, una tassonomia importante per avere una visione più chiara dei vari tipo di crawler esistenti.

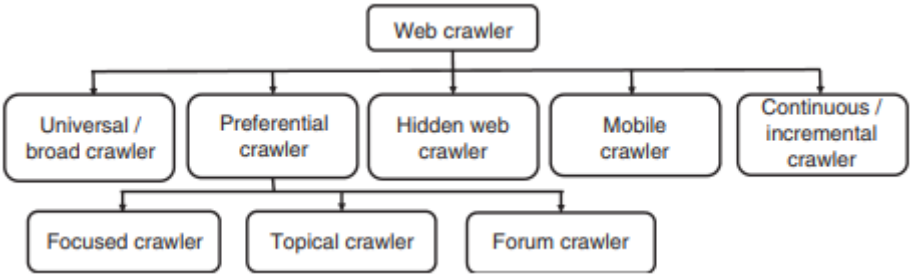

FIGURA 5 – TASSONOMIA ALGORITMI DI WEB CRAWLING

- Universal/broad crawler: Questo tipo di crawler naviga salva e mette in coda tutti i link presenti nelle pagine per una ricerca più ampia possibile e poco controllata, scaricando tutte le pagine web a cui riesce ad arrivare.

- Preferential crawler: La sua caratteristica è quella di essere guidato nella sua navigazione da un argomento preciso specificato dall'utente che caratterizzerà le pagine che esso dovrà scaricare ed i relativi link che dovrà salvare nella sua coda. Questo tipo di crawler può essere suddiviso a sua volta in:
  - Focused crawler: esso cerca in maniera molto orientata tramite l'etichettatura di pagine web che vengono classificate come rilevanti o meno le informazioni relative ad un argomento di interesse.
  - Topical crawler: tramite questo algoritmo vengono ricercate pagine web specificando solo l'argomento d'interesse relativo.
  - Forum crawler: ricerca le pagine web specifiche relative ad un forum
- Hidden web crawler: è usato per il crawling del così detto deep web ovvero quella parte di internet a cui non si riesce ad accedere direttamente tramite gli hyperlinks estraibili dalle pagine web.
- Mobile crawler: è implementato in modo da non sovraccaricare spostando la parte di selezione e filtraggio delle pagine web lato server invece che sul motore di ricerca
- Continuos/Incremental crawler: il funzionamento principale di questo tipo di crawler è quello di mantenere aggiornato il database delle pagine indicizzate poiché i dati sul web sono dinamici e subiscono cambiamenti molto spesso.

2.3 Combinazione crawling e scraping

Il modo di sviluppare un algoritmo che unisca le funzionalità del crawling a quello dello scraping dipende molto dallo scopo che si vuole raggiungere e da quanto sono complicate da implementare le due parti. Per i casi in cui la ricerca da parte del crawling è molto ampia, come per le tipologie di crawler descritte nella sezione 2.2.1 è buona norma separare la parte di codice riguardante l'algoritmo di crawling da quella che riguarda quella di scraping in modo che l'intero programma risulti più robusto. In questo modo la parte di navigazione dei siti, l'estrazione degli hyperlinks e l'inserimento degli stessi nella coda sarà separata da quella in cui si estraggono le informazioni dalle pagine web. Il nome web crawler è però spesso usato anche in contesti più semplici in cui si propone di ampliare il concetto di scraping, indicando la volontà di voler scaricare dati di molteplici pagine web in modo automatico. In questa luce il crawler diventa un

software un po' più elaborato dello scraper ma con lo stesso obbiettivo di fondo. In molti casi, relativamente più facili da gestire, la fase di crawling è ristretta a un gruppo di pagine web con confini bene stabiliti.

2.4. Vantaggi dell'applicazione realizzata

Le tecniche di web crawling e web scraping raggiugono la loro massima utilità nel momento in cui vengono sviluppate per un preciso scopo di business. A tal proposito, dopo un'ampia ricerca all'interno della letteratura, si è constatata la mancanza di un algoritmo di web crawling che potesse soddisfare le esigenze dell'ente collaborante "Fondazione Torino Wireless".

Torino Wireless necessita di uno strumento che, dato un insieme di URL di aziende ad essa collegate, scarichi informazioni dai loro siti web in base a euristiche precise e definite dagli analisti della Fondazione. Tali euristiche servono a raccogliere informazioni riguardo le aziende, i cui URL sono presenti nel database inziale. In particolare, per ogni azienda, le informazioni di maggiore interesse in una ricerca preliminare sono:

- Una sua descrizione generale
- Le sue competenze specifiche
- I casi applicativi in cui è coinvolta
- I progetti che ha intrapreso
- La sua rete di collegamenti, quindi i suoi partner e clienti
- La sua posizione geografica
- I suoi contatti
- La sua partita iva
- La sua presenza sui social

Tali informazioni sono volte a dare una prima profilazione dell'azienda in modo che il data analyst incaricato potesse ricostruire un profilo aziendale senza consultare direttamente il sito web, operazione che avrebbe comportato un sostanzioso impiego di tempo, soprattutto se ripetuta per un numero maggiore di aziende. Un altro punto importante per Fondazione Torino Wireless era quello relativo alla creazione di una sintesi dei testi che descrivono le varie imprese. A questo proposito è stato implementato un algoritmo volto ad arricchire il carico informativo tramite l'estrazione di parole chiave in grado di descrivere l'azienda target. Successivamente è stato ritenuto interessante provare a costruire un

modello di machine learning per classificare le aziende secondo il relativo settore per ottenere informazioni ancora più generali su di esse.

A valle di queste considerazioni si è pensato allora, nel presente progetto di tesi, di implementare un algoritmo che avesse i requisiti necessari per soddisfare gli obbiettivi operativi di Torino Wireless. Tale algoritmo, oltre che utilizzare l'intelligenza artificiale, combina le caratteristiche del web crawling a quelle del web scraping tramite un'implementazione congiunta delle due parti. Il motivo di questa scelta deriva dal fatto per cui le informazioni che si vogliono ricavare provengono da un insieme prestabilito di pagine web i cui URL sono contenuti in un database di input iniziale.

3 Strumenti, tecniche e librerie per il web scraping

A causa della forma non strutturata del web non è facile reperire ed esportare i dati in modo semplice. Alcuni siti come Facebook, Google o Twitter mettono a disposizione delle API (Application Programming Interface) che permettono all'utente di accedere ai dati più facilmente. Ovviamente questo non basta perché non tutti i siti forniscono servizi simili oppure ciò che è di nostro interesse non è accessibile solo tramite l'uso delle API. Per questo diventa necessario l'uso di strumenti più complessi per accedere ai dati, salvarli, analizzarli e usarli nel modo che ci serve per raggiungere il nostro obbiettivo. Gli strumenti e le tecniche che si possono utilizzare sono molti. Nel presente capitolo ne analizzeremo alcuni dei più celebri nello stato dell'arte e spiegheremo la motivazione che ci ha portato a sceglierne uno in particolare.

3.1 Selenium

Come si apprende dalla letteratura e in particolare da [5] e [6], Selenium è un tool e una libreria di Python usata principalmente per l'automazione del browser web. Il suo funzionamento si basa sulla emulazione di azioni reali di un utente fisico su un browser, come un click o una compilazione di campi testuali. È importante notare che l'uso di Selenium non basta a garantirne il corretto funzionamento ma è necessaria l'integrazione con un secondo attore, il Web Driver, di cui esistono diverse versioni specifiche per il browser che andremmo ad utilizzare. Grazie all'interazione tra queste componenti sarà possibile avere un riscontro visivo e reale, infatti, all'esecuzione del codice, il browser si aprirà e si vedranno le azioni simulate dal browser che seguiranno ovviamente il codice scritto dal programmatore. Un'altra caratteristica molto importante di Selenium e che rende possibile in modo molto più semplice di altri tool di gestire appropriatamente anche i siti web che fanno un uso massivo di JavaScript, lasciando, di fatto, la pagina HTML vuota di qualsiasi cosa da estrarre.

3.2 Requests e BeautifulSoup

Requests è la libreria di Python che permette di effettuare richieste ed elaborare risposte HTTP in modo da stabilire una comunicazione tra il software e il web. BeautifulSoup è la libreria, con la documentazione di riferimento disponibile in [7] che permette di parsificare una pagina HTML in modo che da essa si possano ricavare delle informazioni utili, in particolare si occupa di trasformare la pagina

HTML in una struttura ad albero le cui parti possano essere facilmente identificate grazie ai numerosi metodi messi a disposizione dalla libreria. Il suo funzionamento prevede che, come prima cosa, venga creato un oggetto BeautifulSoup al quale deve essere passato il contenuto della pagina web HTML. Questo parametro da solo non basta, infatti, l'oggetto BeautifulSoup ha bisogno che venga specificato un parsificatore specifico che si occupi di rendere leggibili i dati ottenuti tramite la risposta HTTP. I possibili parsificatori, come indicato in [5] sono:

- html.parser è il parsificatore di default ovvero quello con cui viene fatta la parsificazione nel caso in cui nessun parametro venisse specificato. In questo caso il codice restituisce un "warnings". Questo parsificatore funziona correttamente e non richiede installazioni aggiuntive.
- lxml è molto veloce ma richiesta una installazione aggiuntiva
- html5lib mira ad analizzare la pagina web esattamente nello stesso modo in cui lo fa il browser web ma è un po' più lento nell'esecuzione.

La combinazione di queste due librerie è perfetta per lo scraping di dati dalle pagine web.

3.3 Scrapy

Come si legge dalla documentazione di riferimento Scrapy è un framework open source scritto in linguaggio Python che viene utilizzato per navigare il web seguendo lo stesso funzionamento di un web crawler e per estrarre dati strutturati impiegabili in una moltitudine di applicazioni. Come si evince dallo studio [5] Scrapy è uno strumento molto potente con una interfaccia di programmazione un po' diversa da quelle di Selenium o di BeautifulSoup o Requests. La sua utilità spicca soprattutto nel caso in cui si debba scrivere un crawler robusto. Scrapy non fornisce la possibilità di gestire i siti scritti in JavaScript, al contrario di Selenium.

3.4 Rvest

Rvest è un package per lo scraping del web scritto in linguaggio R ovvero un linguaggio di programmazione utilizzato principalmente per l'analisi statistica dei dati. È molto simile alla libreria BeautifulSoup.

3.5 Scelta della tecnica per il caso studio in esame

Dopo lo studio delle varie tecniche di scraping e crawling ed un approfondito studio del caso di business posto dall'azienda collaborante si è constatato che il l'area più importante del progetto era quella relativa al processo di web scraper e che i siti scritti prevalentemente con codice JavaScript rappresentavano la minoranza. A tal proposito si è scelta la strada più congeniale allo scopo. Infatti, ricordiamo che Selenium, come spiegato precedentemente, è un tool che automatizza il processo di navigazione di un sito, i tools come Scrapy sono molto utili nel caso in cui la parte più importante del codice fosse quella relativa al crawling, rvest è usato nel linguaggio di programmazione R mentre la libreria BeautifulSoup è stata realizzata per il preciso scopo di raccogliere le informazioni necessaria da una pagina HTML o XML ed è disponibile in linguaggio Python. Essa, infatti, tramite delle specifiche funzioni in linguaggio Python concede molta libertà allo sviluppatore garantendogli la possibilità di navigare e modificare in modo efficiente l'albero parsificato della pagina HTML di origine.

4. Sviluppo dell'applicazione

Dopo un primo colloquio con l'azienda di riferimento che ha collaborato nello sviluppo dell'applicazione fornendo i dati di input, si è deciso di scegliere come linguaggio di programmazione il linguaggio Python. Tale decisione scaturisce da due motivazioni particolari, la prima è quella per cui l'applicativo usato fino ad ora dall'azienda è scritto in Python, quindi sarebbe stato più facile per quest'ultima capire il funzionamento di quello descritto e implementato nel lavoro di tesi. La seconda motivazione è che tale linguaggio, oltre a prestarsi molto bene all'obbiettivo del progetto, non si discosta molto dalle logiche degli altri linguaggi imparati durante il corso di studi. Esso, infatti fa parte di quei linguaggi di programmazione così detti "objected-oriented", studiati ampiamenti nel corso della formazione universitaria. Di seguito è riportata la descrizione completa dell'architettura del software.

4.1 Architettura del Sistema

Con il termine architettura applicativa si intendono i modelli e le tecniche utilizzate per realizzare una applicazione, questo procedimento fa in modo che essa risulti ben strutturata in tutte le sue componenti. La presente sezione si propone di descrivere l'architettura dell'applicativo creato, visibile in Figura **6**, nella sua interezza e rendere chiaro il suo funzionamento.

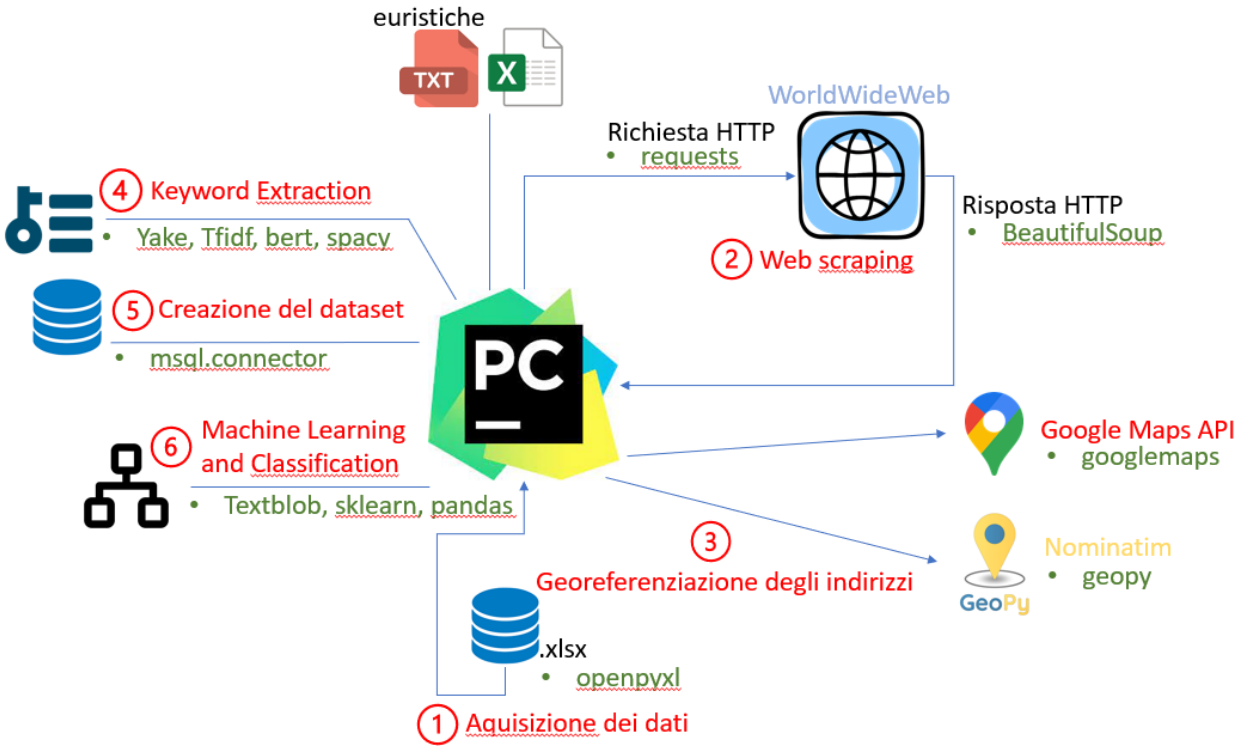

FIGURA 6-ARCHITETTURA DEL SOFTWARE

Il fulcro di tutto il sistema è PyCharm, un ambiente di sviluppo integrato (IDE) usato per la programmazione in linguaggio Python. La scelta è stata guidata dalle molteplici agevolazioni che la piattaforma fornisce durante la programmazione come l'interfaccia intuitiva e user-friendly, la gestione degli errori, la facilità del processo di importazioni delle librerie e tanto altro. All'interno dei file .py è scritto infatti tutto il codice necessario al corretto funzionamento del software. L'architettura del software può essere descritta attraverso le fasi di sviluppo del software:

1. La prima fase dello sviluppo è quella dell'*acquisizione dei dati*. In questa fase vengono scaricati e acquisiti i dati di input forniti dall'azienda collaborante. Si tratta di un file in formato xlsx contenete 300 URL di siti web aziendali importato tramite la libreria openpyxl.
2. La seconda fase, denominata *web scraping,* compone il primo macro-blocco che contraddistingue l'intero lavoro. Questa contiene al suo interno le due micro-fasi. La prima delle due è quella in cui ci si occupa della effettuazione di richieste HTTP dal web browser al server sul World Wide Web mentre durante la seconda micro-fase si riceve una risposta HTTP. La richiesta e la risposta vengono gestite dal modulo requests che il

linguaggio Python mette a disposizione. Nella risposta otteniamo una pagina HTML che verrà interpretata, in modo che diventi gestibile, dalla libreria BeautifulSoup. Infine, il software è in collegamento con un file txt in cui vi sono tutte le euristiche utili al corretto funzionamento della parte di scraping e con diversi file xlsx contenenti dati utili allo sviluppo delle funzioni dell'applicativo.

3. La terza fase si chiama *georeferenziazione degli indirizzi.* L'architettura prevede che il software, una volta acquisiti i dati di interesse dai siti web target, proceda alla georeferenziazione di indirizzi fisici tramite delle tecniche di geocoding. Per far questo sono previste due possibili strade. La prima è quella di servirsi delle API messe a disposizione da Google Cloud Platform, servizio a pagamento a fronte di un numero massivo di richieste, mentre la seconda è quella che prevede l'utilizzo di un servizio simile, meno performante ma completamente gratuito, messo a disposizione da Geopy, libreria messa a disposizione da Python.
4. La quarta fase del progetto è quella di *Keyword Extraction*. Per questa fase ci si è avvalsi dell'uso delle librerie Yake, Tf-idf e Bert al fine di implementare tre algoritmi diversi per il nostro scopo e valutarne, in un secondo momento, le relative performance.
5. La quinta fase descritta dall'architettura si chiama *Classification.* Per quanto riguarda questa fase sono state usate le librerie textblob per la traduzione del testo, sklearn per implementare gli algoritmi di classificazione e di pandas per la gestione dei dati.

4.2 Le fasi dello sviluppo del software

Possiamo descrivere il funzionamento del software attraverso la narrazione dettagliata dei vari step che costituiscono la sua composizione. Le diverse fasi che andremmo a descrivere sono state implementate in ordine temporale. Si sottolinea che le sole informazioni condivise come input da Torino Wireless sono informazioni pubbliche ricavabili dai siti web delle aziende e che da questo momento in poi alcuni nomi delle aziende saranno oscurati.

4.2.1 Acquisizione dei dati di input

L'inizio del lavoro ha avuto luogo nel momento in cui l'ente collaborante ha fornito i dati di input. In particolare, l'input è rappresentato da un set di 300 record che rappresentano una lista di aziende presenti nel network di Fondazione Torino

Wireless. Come si evince dall'estratto del database in Figura **7**Figura **7** ogni record è descritto da tre campi: la ragione sociale dell'azienda, la partita iva dell'azienda e il campo URL che contiene il collegamento al sito web aziendale.

Ragione sociale	PIVA	Sito WEB
@CULT S.R.L.	06851241007	http://www.atcult.it
3A - SOCIETA' DI SVILUPPO PER L'AMBIENTE E L'AGRO-ALIMENTARE A RESPONSABILITA' LIMITATA SIGLABILE "3A S.R.L."	07366180011	https://www.green-planet.it/

FIGURA 7-STRUTTURA DATABASE DI INPUT

Una volta effettuata l'analisi dei dati si è provveduto a selezionare quelli di nostro interesse ovvero i siti web aziendali ed a importarli all'interno dell'applicativo grazie alla libreria openpyxl che permette di gestire correttamente file di questo tipo. Così si è creata una lista di siti web, necessaria per la fase successiva.

4.2.2 Web scraping

La fase di web scraping è uno dei centri focali dell'intero lavoro, tanto che è possibile dividerla in due sottofasi che andremmo a descrivere nelle sezioni seguenti.

4.2.2.1 Richiesta HTTP, risposta HTTP e BeautifulSoup Object

Per ogni sito web aziendale presente nella lista creata nella fase precedente il software esegue una richiesta http di tipo GET. Nel metodo, oltre al necessario URL di destinazione della richiesta si è deciso di impostare anche un parametro time-out di 30 secondi per consentire il caricamento delle pagine web più lente. Come si evince dalla Figura **8**, è presente, inoltre, un'accurata gestione degli errori tramite l'uso del blocco try-except.

```python
def richiesta_bs(url_):
    try:
        resp = requests.get(url_, timeout=30)
        url_response = resp.url
        if url_ != url_response:
            resp = requests.get(url_response, timeout=30)
    except requests.exceptions.SSLError:
        print("Errore SSL, impostiamo verify=False")
        resp = requests.get(url_, verify=False, timeout=30)
    except requests.exceptions.ConnectionError:
        resp = "<body>Errore di connessione, non è possibile effettuare la richiesta</body>"
        return BeautifulSoup(resp, 'html.parser'), "errore di connessione, no url"
    except requests.exceptions.TooManyRedirects:
        resp = requests.get(url_, verify=False, timeout=30, allow_redirects=False)
    except requests.exceptions.InvalidURL:
        resp = "<body>Errore nella realizzazione dell'url che non viene raggiunto</body>"
        return BeautifulSoup(resp, 'html.parser'), "il software non riesce a comporre l'url" \
                                                   " di destinazione"
    except requests.exceptions.ReadTimeout:
        resp = "<body>TIMEOUT OCCURRED</body>"
        return BeautifulSoup(resp, 'html.parser'), "Timeout Occurred"

    return BeautifulSoup(resp.content, 'html.parser'), resp.url
```

FIGURA 8-CODICE RICHIESTA HTTP E CREAZIONE OGGETTO BEAUTIFOULSOUP

In particolare, si gestiscono i seguenti errori:

1) SSLError: se incorriamo in questo errore vuol dire che il sito a cui stiamo cercando di accedere non sta fornendo i giusti certificati che dovrebbero stabilire un collegamento crittografato tra un browser e un server o una rete. Dato che in questo caso non stiamo fornendo dati sensibili impostiamo un campo verify=False così che non si debba verificare la presenza di tali certificati. In questo caso la richiesta va a buon fine e si riesce ad ottenere la risposta desiderata.
2) ConnectionError: questo errore viene scatenato nel momento in cui si presente un errore di connessione con il sito destinazione, in questo caso non si può fare niente e ci si deve accontentare di segnalare l'errore tramite il corrispettivo messaggio "Errore di connessione, non è possibile effettuare la richiesta" passandolo come fosse una pagina HTML all'oggetto BeautifulSoup, non otterremo infatti nessuna risposta utile.
3) TooManyRedirects: indica il fatto per cui la richiesta non venga mai completata perché il sito web continua ad essere reindirizzato. Come il caso 1), è possibile gestire questo errore in modo da non rinunciare

all'informazione che si sta cercando impostando il campo allow_redirects=False.

4) InvalidURL: questa eccezione indica che l'URL che cerchiamo di raggiungere non è valido. Gestire questo errore significa contenere tutte le situazioni in cui la pagina che stiamo cercando non esiste più oppure che l'URL dato in input alla richiesta è stato originariamente scritto in maniera sbagliata o sia incompleto. Anche in questo caso viene comunque creato un oggetto BeautifulSoup e viene passato come parametro al suo interno una pagina HTML creata manualmente contenente il messaggio "Errore nella realizzazione dell'url che non viene raggiunto".
5) ReadTimeout: nel caso in cui si generi questo tipo di errore vuol dire che la pagina, dopo il tempo di timeout di 30 secondi impostato precedentemente, non è stata comunque caricata. In questo caso l'errore non dipende dal software implementato quindi si può solo gestire attraverso la visualizzazione del messaggio "TIMEOUT OCCURRED" passato, anche in questa situazione, come pagina HTML, nell'oggetto BeautifulSoup.

Dall'analisi dei risultati ottenuti dopo l'invio delle richieste HTTP si è riscontrato che su 50 siti web totali, sono andate a buon fine 46 richieste. Si sono registrate allora una percentuale di siti web non raggiunti dell'8%.

Una volta effettuata la richiesta HTTP si riceve come risposta una pagina HTML grezza. Per gestire la pagina in modo adeguato e poter accedere a tutto il suo contenuto abbiamo bisogno di parsificare la pagina ovvero analizzare il flusso di dati che essa contiene. Per farlo abbiamo bisogno dell'ausilio di una libreria di Python chiamata BeautifulSoup. In particolare, come si vede in Figura **8**, dobbiamo creare un BeautifulSoup Object che prende come parametri in ingresso la risposta ed un parsificatore di pagine HTML. In Python esistono più parsificatori come html.parser, lxml e html5lib, si è scelto il primo perché funziona correttamente, è veloce e non richiede installazioni aggiuntive.

4.2.2.2 Estrazione dell'informazione

Come introdotto precedentemente il funzionamento di un software di web scraping prevede che ad ogni richiesta HTTP segua la fase di estrazione delle informazioni dal sito web designato. L'obbiettivo in questa fase è quello di creare

un database con dei campi testuali che caratterizzino un'azienda in base alle informazioni riportate sul suo sito web. I campi testuali sono valorizzati e separati tra loro in base a delle euristiche precise fornite in un primo momento come dati di input dall'azienda collaborante. Tali euristiche sono poi state ampliate sotto l'ipotesi per cui altre informazioni che sono state riscontrate all'interno dei siti web e poi giudicate di molta importanza non erano estraibili con le sole euristiche di partenza. L'implementazione del codice di programmazione, che verrà spiegata nel particolare nella sezione che segue, è stata fatta ad hoc per ognuno dei campi testuali. Si riporta inoltre in Figura **9** la corrispondenza tra campi testuali ed euristiche.

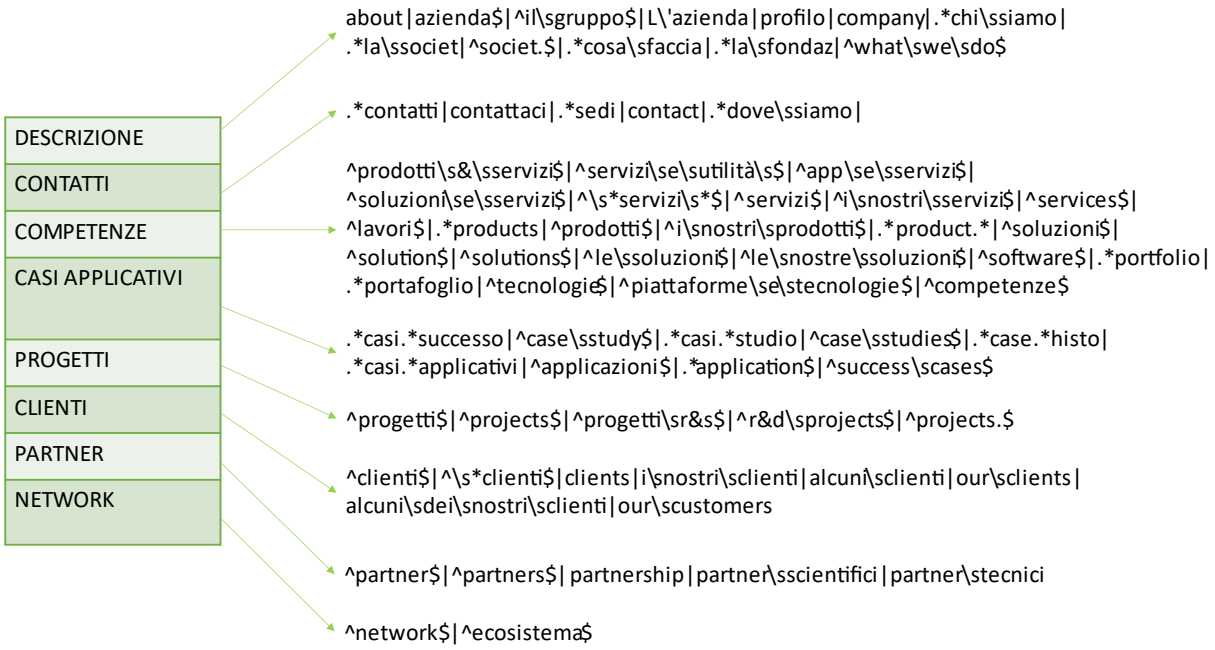

FIGURA 9-CORRISPONDENZA TRA EURISTICHE E CAMPI TESTUALI

Le euristiche, sotto forma di espressioni regolari, sono contenute in un file di testo in formato .txt all'interno del progetto così che l'operazione di lettura risulti facile e snella. In particolare, come si evince dalla Figura **10**, si leggono le righe del file e si memorizzano in una lista di stringhe così da rendere agile l'accesso. Basterà infatti invocare la stringa di interesse posta in una posizione nota all'interno della lista.

```
with open("euristiche.txt") as f:
    lines = f.readlines()
```

FIGURA 10-APERTURA, LETTURA E SALVATAGGIO FILE TXT

Di seguito vengono esposti i ragionamenti che sono stati sostenuti per implementare la logica del codice di programmazione relativo allo scraping delle varie sezioni delle pagine web dei siti aziendali.

1) DESCRIZIONE:

   Dopo alcuni controlli sulla correttezza di scrittura dell'URL di partenza e dopo aver richiesto la pagina web di riferimento si ricerca all'interno della stessa, già sottoforma di oggetto BeautifulSoup un link, indicato dal tag <a>, che abbia all'interno del testo l'euristica corrispondente. Questo risulta possibile grazie al metodo di ricerca invocabile sull'oggetto BeautifulSoup, tale metodo è uno dei metodi principali della libreria ed è quello che permette di trovare frazioni della pagina HTML indicando il tag nel quale sono contenuti. La sua struttura è la seguente:

   - **find(name, attrs, recursive, string, **keywords)**

   mentre il modo in cui è stato utilizzato all'interno del codice è riportato in Figura **11** dal quale si vede che risulta possibile specificare all'interno del campo "attrs" sia un parametro text che accetta delle espressioni regolari come input sia un parametro href che indica che il tag <a> trovato debba avere un attributo valido di href in cui è specificato l'URL a cui si accede attraverso il click su tale link.

```
verticale_link = soup1.find("a", text=re.compile(euristiche,
                                                 re.IGNORECASE), href=True)
```

FIGURA 11-RICERCA TAG <A> CON ATTRIBUTO HREF VALORIZZATO

Questo URL sarà quello a cui vogliamo arrivare, ovvero la pagina in cui troveremo la descrizione cercata. Nel caso in cui l'URL cercato venga trovato correttamente lo useremo per creare un secondo oggetto BeautifulSoup dal quale scaricare il testo. Dopo alcuni tentativi si è deciso di scaricare il testo dell'intero body della pagina e, successivamente, di pulirlo da eventuali informazioni superflue. Nel caso in cui si arrivi a questo punto ma non si riesca a trovare il tag <body> il campo descrizione verrà valorizzato come "All'interno del sito il tag body = None". Si riporta in Figura **12** un esempio di output.

URL	DESCRIZIONE
url azienda	Chi siamo Un Team dinamico di Consulenti e Formatori al servizio delle imprese *nome azineda* nasce nel 1991 dallo spirito imprenditoriale di professionisti della consulenza aziendale e di docenti universitari. I soci sono tutti impegnati come consulenti di direzione, in ambiti diversi ma strettamente correlati, guidano un solido TEAM di giovani e dinamici collaboratori specializzati nelle diverse aree aziendali . I professionisti *nome azienda*, oltre a possedere competenze tecniche costantemente aggiornate, manifestano capacità di ascolto, flessibilità ed empatia, importanti per entrare rapidamente in sintonia con i clienti I nostri valori Impegno Flessibilità

FIGURA 12-OUTPUT DESCRIZIONE

2) CONTATTI:

I contatti riguardano quella sezione del sito in cui si trovano le informazioni aziendali riferite principalmente al luogo dove sono presenti le sedi dell'azienda target e di conseguenza tutti i relativi indirizzi fisici, i vari numeri di telefono, cellulare o fax, informazioni riguardanti il capitale sociale o la partita iva, sono esclusi ovviamente tutti i form sovente presenti nel sito per contattare l'azienda tramite mail. La procedura della ricerca di questa sezione è, per una prima ricerca, la medesima di quella implementata per la parte della descrizione. Vi è un ulteriore passaggio nel caso in cui la ricerca non produca risultati. Si è deciso di salvare la parte del sito web relativa al footer che spesso si trova riferendosi a tag caratteristici, quindi non è specifico per ogni sito web, e di riferirsi a quest'ultima per reperire le informazioni sui contatti che sono risultati spesso presenti in questa sezione. Dai contatti, su richiesta dell'azienda sono stati estratti e salvati a parte i dati che si riferiscono all'indirizzo, poi utilizzato per la parte di geocoding per ottenere latitudine e longitudine precisa delle varie sedi aziendali, e della partita iva, codice identificativo dell'impresa target. In Figura **13** si riporta un esempio di output del testo relativo alla sezione contatti.

URL	CONTATTI
url azienda	| ©2020 3a Srl - Via Le Chiuse, 68 - 10144 Torino - Tel +39.011.43.78.552 Fax +39.011.43.78.221 - 3a@green-planet.it - P.IVA 07366180011

FIGURA 13-OUTPUT CONTATTI

Una volta effettuato lo scraping del sito per ottenere la sezione dei contatti, questa è stata utilizzata, congiuntamente al testo del footer della pagina web aziendale isolato precedentemente, per estrarre e isolare informazioni più specifiche dell'azienda bersaglio del software. In particolare, si è ritenuto molto utile rilevare il testo che identifica la partita iva, ovvero il numero di 11 cifre necessario ad identificare la società titolare della partita iva stessa. A tal proposito ci siamo serviti dello strumento delle espressioni regolari, gestite in Python tramite la libreria "re", spesso usate per identificare stringhe caratterizzate dalla stessa struttura. Nel caso in cui la partita iva non venga trovata si visualizzerà un messaggio "NO PT_IVA". Si riporta in Figura **14** la sezione di codice relativo a tale procedimento.

```
def get_ptIVA(contatti_, footer_):
    pt_IVA = "NO PT_IVA"
    lista_cf = re.split(r'\W+', contatti_) + re.split(r'\W+', footer_)
    for elemento_ in lista_cf:
        if re.match(r"^[0-9]{11}$", elemento_, re.IGNORECASE) is not None:
            pt_IVA = elemento
            break
    return pt_IVA
```

FIGURA 14-ESTRAZIONE PARTITA IVA

3) COMPETENZE, PROGETTI E CASI APPLICATIVI:

Dopo un'attenta analisi sul campo si è notato che il caso di questo tipo di informazione differisce un po' da quello del campo descrizione poiché in quel caso il testo del link ricercato in base all'euristica era unico dato che all'interno del sito web si riscontra solo una sezione in cui è presente una descrizione generale dell'azienda. Lo stesso ragionamento non si può applicare nel caso delle competenze, dei progetti e dei casi applicativi

poiché è stato appurato, come è dimostrato in Figura **15**, che ci possono essere più link con il testo ricercato in base alla relativa euristica.

FIGURA 15-CORRISPONDENZA TRA MOLTEPLICI TAG <A> E STESSA EURISTICA

Per includere tutti i casi e non avere così delle informazioni parziali si è utilizzato il metodo seguente:

- **findAll (name, attrs, recursive, string, **keywords)**

Come si intuisce, il metodo segue lo stesso funzionamento del caso precedente ed è usato anche allo stesso modo all'interno del programma con l'unica differenza per cui invece di ritornare un unico oggetto ritorna una lista di tutti gli oggetti trovati in base agli attributi specificati. In questo modo, prendendo come esempio il caso in figura, si trovano sia i link corrispondenti al testo "servizi" che quelli corrispondenti al testo "soluzioni". Tutti i link corrispondenti alla stessa euristica sono stati inseriti in una lista che il programma, in un secondo momento, dovrà essere in grado di processare uno ad uno, prendendo così l'accezione di un vero e proprio crawler, il cui funzionamento è stato ampiamente descritto nei capitoli precedenti. Infatti, il software, seguendo la coda, entrerà in tutte le diverse pagine web appartenenti allo stesso sito e ne raschierà il contenuto, salvando le informazioni di cui ha bisogno.

Si è altresì notato, come sottolineato in Figura **16**, che all'interno dei siti web il link corrispondente all'euristica non ha un attributo "href", oppure

l'"href" risulta uguale a "#" o viene valorizzato in modo che non porti a nessuna pagina in particolare. In tutti questi casi l'oggetto associato al tag <a> trovato fa aprire un menu a tendina.

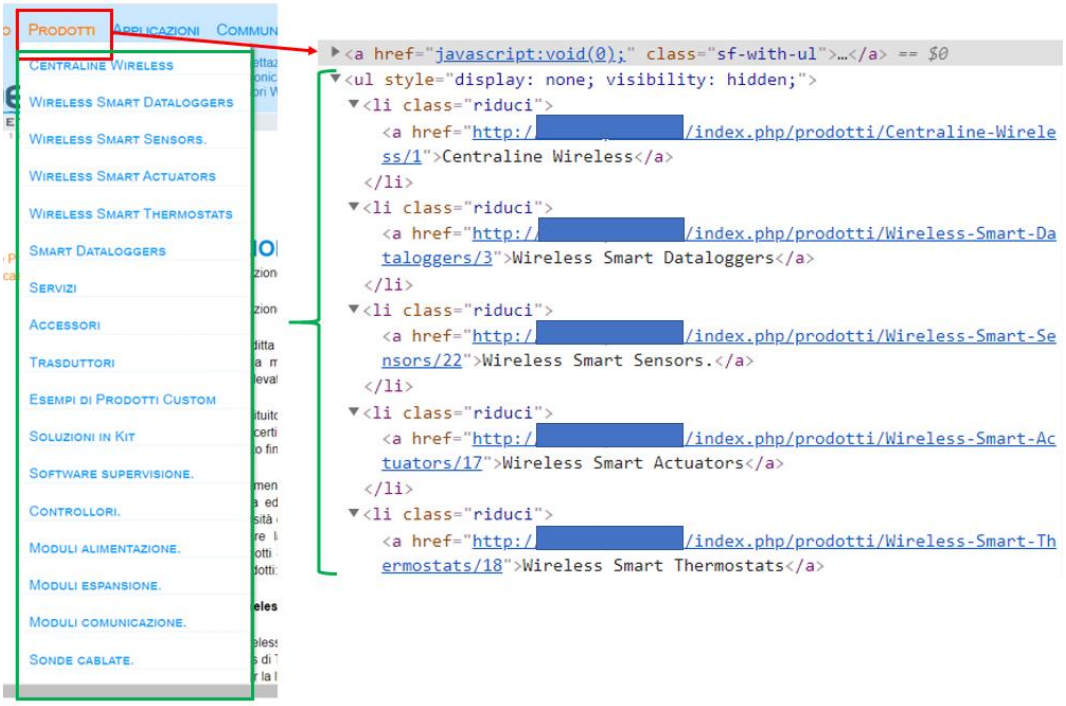

FIGURA 16-CORRISPONDEZA HFREF NON VALORIZZATO CON MENU A TENDINA

È stato pensato quindi di sfruttare questa peculiarità di molti siti creando una lista riassuntiva delle competenze, dei casi applicativi e dei progetti. Anche in questo caso abbiamo sfruttato la tipica struttura di una pagina HTML e l'enorme forza della libreria BeautifulSoup che dato l'oggetto principale ci permette di trovare tutti gli elementi del menu a tendina a lui associato grazie al metodo seguente:

- **findNextSibling ()**

Tutti gli elementi trovati, compreso quello principale, andranno a riempire una lista che andrà a valorizzare un altro campo del database finale. Inoltre, anche gli elementi del menu a tendina verranno inseriti nella coda a cui il crawler dovrà accedere. In questo modo si avrà una massiccia mole di informazioni che comunque sarà anche riassunta nella maggior parte dei casi. È importante mettere in risalto che anche se l'attributo href del tag <a> è valorizzato in modo che cliccando sul rispettivo link si arrivi ad una destinazione è possibile che comunque, passandoci sopra con il

mouse, si apra lo stesso un menu a tendina che sfrutteremo nello stesso modo descritto precedentemente. Questo caso si vede dalla Figura **17**.

FIGURA 17-CORRISPONDENZA HREF VLORIZZATO CON PRESENZA MENU A TENDINA

Si riporta per completezza, in Figura **18** e in Figura **19**, l'output relativo ai due casi.

Servizi, Servizi su misura, Internationalization e finanza agevolata, Consultancy per le aziende, Outsourcing processo produttivo, Facility Management servizi integrati, Help Desk reperibilità 24/7, Internet Services web & mail, Education corsi & training, Soluzioni per il business, Monitoring e Alerting, Consolidation & Virtualization, Cloud Computing, Custom Application Development, Enterprise Printing, HCL Domino (IBM/Lotus Notes), IBM Tivoli Storage Manager, Servizi Voip, Telemetria e Geoposizionamento, Soft CRM, Circuit by Unify Unified Communication, Web Digital Signage Controllo dei contenuti audio/video

FIGURA 18-OUTPUT LISTA RIASSUNTIVA COMPETENZE(1)

Centraline Wireless, Wireless Smart Dataloggers, Wireless Smart Sensors., Wireless Smart Actuators, Wireless Smart Thermostats, Smart Dataloggers, Servizi, Accessori, Trasduttori, Esempi di Prodotti Custom, Soluzioni in Kit, Software supervisione., Controllori., Moduli alimentazione., Moduli espansione., Moduli comunicazione., Sonde cablate., Accessori BEMS.

FIGURA 19-OUTPUT LISTA RIASSUNTIVA COMPETENZE(2)

4) CLIENTI, PARTNER E NETWORK:

Il caso dei clienti, partner e network è ancora differente rispetto agli altri due. Nella maggior parte dei siti, come è visibile nella Figura 18, si è visto che le aziende tendono a mettere, al posto del campo testuale contenente le informazioni della propria rete di connessione con le altre aziende delle immagini dei loghi delle stesse. Questa impostazione è risultata congeniale al nostro obbiettivo poiché abbiamo ottenuto un risultato più schematizzato e quindi meno confusionale riuscendo ad avere una lista sintetica di solo i nomi delle aziende collegate.

Per far ciò si è sfruttato anche qua la struttura della pagina HTML e i principi della buona programmazione riguardo l'inserimento delle immagini all'interno di un sito. Come si nota in Figura **20** questa prevede che all'interno del tag <img> usato per inserire una immagine si valorizzino due importanti attributi, l'attributo "src" e l'attributo alt.

FIGURA 20-INFO NOME AZIENDA IN ATTRIBUTO SRC DEL TAG <IMG>

Nell'algoritmo oggetto del progetto sono stati proprio presi i valori di questi due attributi, puliti di eventuali parole superflue e poi inseriti in una lista che è andata a valorizzare i campi clienti, partner e network del database di cui si riporta un esempio di output in Figura **21**.

URL	CLIENTI	PARTNER	NETWORK
[redacted]	nw, pastorfrigor, poderi gianni gagliardo, sebaste, sant'anna, tecno 3, telerobot, abc international pharma, alstom, anfia, ambro food, api formazione, arione, arteni group, ascensori rossini, asl cn1, bertero mario, bragapan, bongiovanni, borgwarner, bovone, caffarel, celi, ceretto, deagostini, diageo, diasorin, doyouwine, finpiemonte, ferrero, flenco, fornace calandra, giacomini, gioielli di valenza, giraudi, green technik, ideazione, maestri, maina, massucco t, metlac, mista, chiudi, loader, poltronafrau, syntomer, e-lungo, ap-nice, centraledellatte, confindustria, confartigianato, inntecck, lacave, life, miroglio, senza-titolo-1, riso-vignola, saced, bsa, traspeed, waterteck, well-work	sap silver partner, amazon web services partner, microsoft partner, netcomm services, netcomm suisse, polo agrifood, sme, unito, confindustria cuneo, uito, per.form, confartigianato cuneo, apito, 4planning, polo-ict_member, digix	

FIGURA 21-OUTPUT CLIENTI, PARTNER E NETWORK

Vi è stata una ulteriore implementazione dell'algoritmo per valorizzare questa sezione del database. Si è notato infatti che molti siti, come quello appena portato in esempio, presentavano i loghi dei clienti e partner all'interno della home page del sito. La caratteristica che fortunatamente accomuna tutti questi siti è che le immagini di solito sono introdotte da un titolo, riconoscibile nella struttura HTML grazie al tag <h*> dove l'asterisco indica un qualsiasi numero naturale. L'algoritmo qualora la ricerca del tag <a> iniziale non avesse portato a nessun risultato utile sfrutta la peculiarità appena descritta cercando il testo dell'euristica all'interno dei titoli della pagina web. Nel caso in cui la ricerca porti un risultato positivo l'algoritmo cerca tutti i tag <img> al di sotto del tag <h*> così da ridurre al minimo la possibilità di includere informazioni superflue. La sezione partner del testo relativo all'esempio riportato in Figura **21** è stata valorizzata nel modo appena descritto. Si riporta invece la corrispondenza tra questa ed il sito web in Figura **22**.

FIGURA 22-CORRELAZIONE TAG <H*> CON TASSONOMIA PARTER

In questa fase si è notato che i siti web aziendali presentano strutture della pagina HTML molto particolari, diverse tra loro e difficili da gestire in modo generale. In molte occasioni si è dovuto gestire i singoli tag HTML che corrispondevano a specifiche sezioni del sito di una singola azienda per non mostrare informazioni inutili allo scopo di business. Ovviamente è impensabile ripetere questa azione per ogni sito web anche perché verrebbe meno lo scopo principale del web scraper, ovvero l'automazione. Fortunatamente, alcuni pattern di impostazione della pagina HTML sono validi per molti siti. Alcuni di questi sono la formattazione del footer del sito che spesso è indentificata dal tag <footer> o da tag con attributo "id" o "class" con la parola "footer" nel nome. Si riporta un esempio con la Figura **23** che rappresenta il codice con cui si riesce ad eliminare tutti i footer dei siti aziendali. Lo stesso procedimento si può effettuare per la parte relativa alla gestione dei cookie e per l'header.

```python
def elimina_footer(bs2):
    csystem_footer = bs2.find("td", {"class": "testoboxbianco"})
    if csystem_footer is not None:
        csystem_footer.decompose()
    bylogix_footer = bs2.find("div", {"id": "bylogix_footer"})
    if bylogix_footer is not None:
        bylogix_footer.decompose()
    footer_bepseng_ = bs2.find("div", {"class": "footer-information"})
    footer_bepseng = bs2.findAll("div", {"class": "footer-content"})
    if footer_bepseng_ is not None:
        footer_bepseng_.decompose()
        for f in footer_bepseng:
            if f is not None:
                f.decompose()
footer_almaviva = bs2.find("div", {'class': 'footer-wrapper'})
if footer_almaviva is not None:
    footer_almaviva.decompose()
footer = bs2.select_one("footer")
div_footer = bs2.find("div", {'id': 'footer'})
footerend = bs2.find("div", {'id': 'footerend'})
footersecondrow = bs2.find("div", {'class': 'footer-second-row'})
footerlastrow = bs2.find("div", {'class': 'footer-last-row'})
footerhome = bs2.find("div", {'id': 'footer-home'})
if footer is not None: footer.decompose()
if div_footer is not None: div_footer.decompose()
if footerend is not None: footerend.decompose()
if footersecondrow is not None: footersecondrow.decompose()
if footerlastrow is not None: footerlastrow.decompose()
if footerhome is not None: footerhome.decompose()
btrees_footer = bs2.find("div", {"class": "elementor-location-footer"})
if btrees_footer is not None:
    btrees_footer.decompose()
```

FIGURA 23-CODICE ELIMINAZIONE FOOTER SITI WEB AZIENDALI

4.2.3 Georeferenziazione degli indirizzi

Dopo la fase di estrazione dei dati dai siti web si è manifestata la necessità di estrarre alcune informazioni particolari delle diverse sezioni appena descritte e di isolarle dal resto. A tal proposito, si è implementato un algoritmo che, una volta ricevuto in input il testo della sezione contatti, riesce a separare dal resto le stringhe contenenti i potenziali indirizzi delle varie sedi dell'azienda di cui si stanno reperendo le informazioni. Sono stati trattati diversamente il caso delle sedi aziendali presenti nelle città italiane e quello delle sedi presenti in stati

esteri. Per il primo caso abbiamo sfruttato la lista di tutte le città d'Italia presente sul sito dell'ISTAT. Questa è stata unita al programma e ci ha permesso di trovare una corrispondenza tra le parole del testo dei contatti e le città, fondamentale per l'applicazione dell'algoritmo descritto in questa sezione. Una volta trovato l'indirizzo in questo caso è stata fatta l'operazione di geocoding ovvero quel processo che prende una descrizione testuale di una posizione e ne restituisce le coordinate geografiche, latitudine e longitudine. Si è deciso di implementare il codice relativo a questa fase in due modi, così che l'azienda collaborante avesse più alternative tra cui scegliere. Il primo modo è stato quello di sfruttare le potenti API messe a disposizione da Google Platform e il secondo è stato quello di appoggiarsi ad un'altra nota piattaforma collaborativa finalizzata a creare mappe del mondo a contenuto libero, chiamata OpenStreetMap, tramite la libreria geopy per Python. Entrambi i metodi presentano dei lati positivi e dei lati negativi. Analizzando i risultati su 50 aziende dell'intero database si è visto che il primo, quello che sfrutta l'API di Google ha una percentuale d'errore dello 0% ovvero riesce a geolocalizzare tutti gli indirizzi scritti in modo corretto mentre il secondo presenta una percentuale più alta d'errore, nello specifico non riesce a trovare circa il 20% degli indirizzi anche se scritti in maniera corretta. D'altra parte, il primo metodo prevede un pagamento nel caso in cui se ne voglia fare un uso massivo mentre il secondo è totalmente gratuito. Si riporta l'output ottenuto in questa fase del lavoro in Figura **24**.

LAT_LONG	LAT_LONG_GOOGLE
via statuto 8 12100 cuneo (44.3867353, 7.54694) corso trapani 16 10139 torino (45.0740805, 7.641591)	via statuto 8 12100 cuneo (44.3867029, 7.5468375) corso trapani 16 10139 torino (45.0740122, 7.641501)
	via l b alberti 34 48124 ravenna (44.40061439999999, 12.1861906)

FIGURA 24-OUTPUT LATITUDINE LONGITUDINE

Il caso delle aziende con sedi dislocate al difuori dello stato italiano è stato più difficile da gestire durante l'implementazione poiché non è stato possibile reperire online una lista completa di tutte le città presenti all'estero. Si è deciso allora di scaricare una lista di tutti gli stati esteri e delle relative capitali poiché all'interno dei vari siti si è notato che in alcuni casi veniva specificato solo lo stato e in altri solo la città. L'algoritmo, come prima, esegue un controllo per verificare la corrispondenza tra le parole del testo dei contatti e quelle presenti nella lista delle

città e degli stati. In caso di riscontro positivo oltre ad incrementare un contatore si valorizza un campo chiamato “sedi estere” all’interno del database con il nome della città o dello stato in cui effettivamente è presente una sede della azienda in questione. In caso di riscontro negativo il campo testuale verrà valorizzato come “Non sono state trovate sedi estere dell’azienda”. Anche in questo caso si riporta in Figura **25** un esempio di output corrispondente a questa fase.

COUNT_SEDI_ESTERE	SEDI_ESTERE
0	Non sono state trovate sedi estere dell'azienda
1	russia

FIGURA 25-OUPUT INFORMAZIONI SEDI ESTERE AZIENDA

L’operazione di geocoding è stata ritenuta importante perché oltre a fornire istantaneamente una panoramica efficiente sulla dislocazione delle sedi delle diverse aziende, anche grazie all’implementazione della sezione di codice relativa alla estrazione delle sedi estere, potrebbe consentire in una fase successiva di creare una mappa rappresentabile tramite strumenti di visual analytics, utile per fornire una visione più generale al data analyst che valuterà per effettuare efficientemente le relative decisioni di business.

4.2.4 Preprocessing e keyword Extraction

Dopo la fase di geocoding descritta nella sezione 4.2.3 è stata implementata la porzione di codice relativa ad un’ulteriore fase di estrazione di conoscenza. Tale implementazione è stata possibile grazie all’avvenuta acquisizione dei dati tramite la fase di web scraping descritta nella sezione 4.2.2.2. L’estrazione della conoscenza di cui si parla avviene nel momento in cui riusciamo a descrivere un testo articolato e complesso da una serie di parole chiave. Nelle sezioni seguenti ci occuperemo di spiegare come questa sia stata eseguita e di quali processi e strumenti ci si è serviti.

4.2.4.1 Preprocessing

Nell'ambito data science un documento di testo è un insieme di parole che vengono usate per descrivere qualcosa. Si tratta di un dato non strutturato che per poter essere processato ha bisogno di essere preparato in modo opportuno dotandolo così di una struttura. Con il termine data preprocessing si intende quel processo composto da una serie di tecniche di data mining che consistono e hanno lo scopo di trasformare i dati in un formato adatto alle nostre necessità. Nel nostro caso risulta una fase fondamentale che deve precedere sia quella relativa all'estrazione delle parole chiave dal testo sia quella relativa alla classificazione. Vi sono molte funzioni precompilate disponibili tra le librerie di Python ma per questo caso si è scelto di customizzare il processo che per semplicità di esposizione divideremo in fasi:

1) Document splitting

   È la fase in cui si deve definire il concetto di record ovvero dividere il documento di testo in parti a seconda dell'obbiettivo che abbiamo. Si tratta di decidere quale deve essere il l'elemento base da analizzare che può essere un paragrafo, una frase una sezione o tutto il documento. Più è piccolo questo elemento, maggiore sarà l'impatto delle parole che lo compongono. Nel nostro caso, dato che l'input di testo è relativamente corto si è deciso di prendere come elemento di analisi tutto il testo che abbiamo a disposizione.

2) Tokenizzazione

   Con tokenizzazione si intende la divisione del testo grezzo ovvero del record definito nella fase precedente, in frasi o parti di esse, nel primo caso parleremo di tokenizzazione delle frasi mentre nel secondo caso, prendendo il più estremo, parleremo di tokenizzazione di parole. Per il nostro obbiettivo operiamo la seconda delle due tecniche appena esposte, dividendo il testo in singole parole che chiameremo token.

Dopo aver convertito il testo di input in caratteri minuscoli è possibile scegliere tra le tecniche note di tokenizzazione, identificate grazie a [8] e riportate in Figura **26**.

FIGURA 26-ALGORITMI DI TOKENIZZAZIONE

Nel nostro caso si è scelta la tecnica basata su regole, in particolare sulle espressioni regolari dato che esse risultano congeniali per uno scopo custom. A tal fine si è importata una libreria specifica in Python dal package nltk.tokenize chiamata RegexpTokenizer. Questa ci ha permesso di creare un tokenizer customizzato che dividesse tutte le parole in base alla regola "\w+" che indica che il testo deve essere diviso in token che risultano sequenze formate da ogni carattere alfabetico o numerico, equivalente a [a-zA-Z0-9_].

3) Rimozione delle stopwords

La fase in questione è quella centrale della parte di preprocessing. Le stopwords sono tutte quelle parole che non hanno un significato semantico all'interno di un testo e sono utilizzate molto nella scrittura di esso. Non esiste una lista prestabilita di stopwords e inoltre, esse variano da lingua a lingua, per esempio, in italiano esse possono essere preposizioni, articoli, congiunzioni, preposizioni, avverbi ecc... Dato che da questo tipo di parola non scaturisce nessuna informazione è sempre buona norma che esse vengano rimosse prima di una qualsiasi operazione sui dati testuali. Nel nostro caso abbiamo importato una lista

di stopwords, sia inglesi che italiane data la presenza di testi in entrambe le lingue, grazie alla libreria stopwords importata dal package nltk.corpus. Una volta implementato l'algoritmo di keywords extraction dopo la rimozione delle stopwords si è notato che comunque all'interno dei risultati ottenuti vi erano parole che non portavano alcuna informazione importante per la nostra analisi. È stato deciso allora di creare una lista aggiuntiva di parole customizzate trovate dall'osservazione iterativa del risultato dell'algoritmo di estrazione di keywords. Tali parole sono state prima suddivise in categorie come quelle associate ad informazioni temporali, oppure parole riferite all'ambito aziendale in modo troppo generale o altre parole non contenute nella lista di stopwords iniziali, dopo inserite all'interno di liste e infine rimosse dal testo. Si riportano le liste appena descritte in Figura **27**.

```
lista_tempo = ["mese", "year", "years", "presto", "oggi", "day", "gennaio", "febbraio", "marzo", "aprile", "maggio",
               "giugno", "luglio", "agosto", "settembre",
               "ottobre", "novembre", "dicembre"]
lista_geo = ["mondo", "puglia", "napoli", "basilicata", "città", "regione", "genova", "aosta", "italia", "italy",
             "torino", "turin", "torinese", "italiana", "piemonte", "anni", "milano"]
lista_parole_aziendali = ["www", "pdf", "riservati", "decisioni", "obiettivi", "aziendale", "aziendali", "clienti",
                          "informazioni", "prodotti", "prodotto", "cliente", "clients", "customer", "customers",
                          "attività", "srl", "spa", "società", "company", "compani", "cookie", "policy", "iva",
                          "copyright", "sede", "pec", "azienda", "aziende", "gruppo", "made", "imprese"]
custom_stopwords = ["great", "piu", "oltre", "due", "necessario", "migliorare", "nuovo", "ogni", "fra", "nata",
                    "gennaio", "proprio",
                    "insieme", "scopri", "guarda", "segui", "causa", "annullato", "leggi", "continuamente",
                    "propri", "qualcosa", "qualunque", "prima", "dopo", "ognuno", "attenzione", "senza",
                    "contattaci", "info", "grazie", "tutte", "rendere", "sempre", "vuoi", "quando", "quel", "cosa",
                    "persone", "troppo"]
```

FIGURA 27-LISTE CUSTOMIZZATE PER RIMOZIONE STOPWORDS

Uno step successivo è stato quello di cercare di eliminare il nome dell'azienda in oggetto dalle parole del testo di input. Questo è stato possibile grazie alle buone norme di programmazione web per cui, come si nota in Figura **28**, all'interno del titolo della pagina vi è scritto, nella maggior parte dei casi, il nome dell'azienda all'interno del tag <title>, facilmente estraibile tramite i metodi di BeautifulSoup descritti precedentemente. Il nome è stato poi confrontato con la creazione on time di una stringa data dalla parsificazione dell'URL di input relativo alla società e una volta trovata una corrispondenza cercato all'interno del testo in input e poi eliminato.

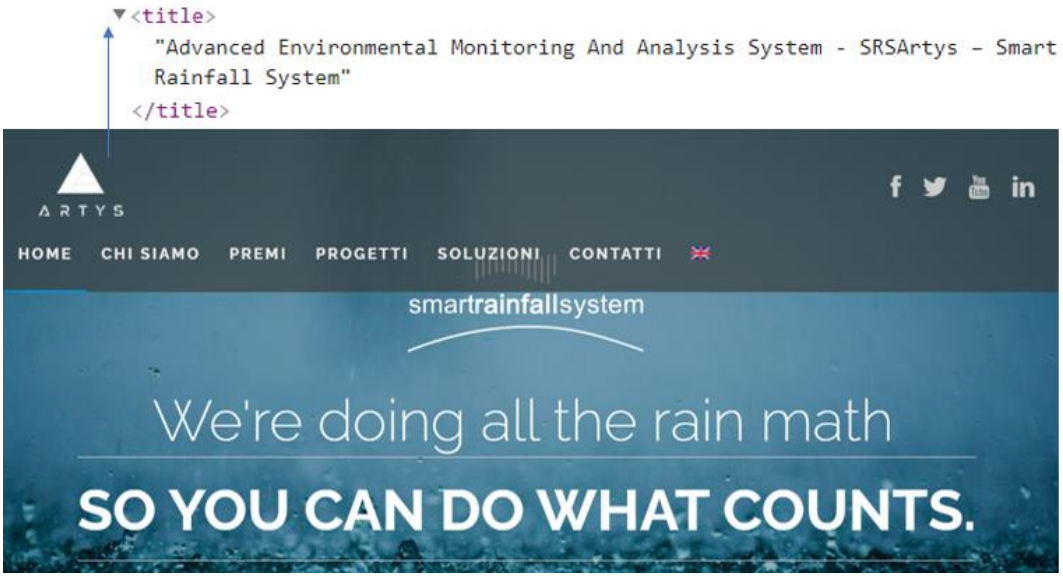

FIGURA 28-CORRELAZIONE TRA TAG <TITLE> E NOME AZIENDA

4) Stemming

L'ultima fase implementata per il completamento del preprocessing del testo è stata quella dello stemming. Come si vede dall'esempio in Figura **29**, identificata grazie a [9], lo stemming è il processo di riduzione della forma flessa della parola alla sua forma radice detta tema che non corrisponde necessariamente alla radice morfologica che invece è chiamata lemma.

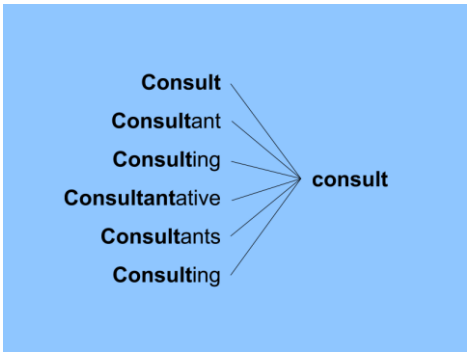

FIGURA 29-ESEMPIO STEMMING

Questo processo è stato implementato in modo che le parole estratte dall'algoritmo di keywords extraction non fossero simili tra loro nel significato e quindi per evitare che si ottenessero parole che apportano lo stesso tipo di informazioni che, se estratte, non avrebbero lasciato la possibilità ad altre, con significato diverso da loro ma comunque presumibilmente ricche di informazioni, di apparire. Per questa analisi ci siamo affidati ancora una volta al package nltk importando uno degli Stemmer più noti in letteratura ovvero lo SnowballStemmer, utilizzabile su testi in più lingue. Tuttavia, dopo le opportune analisi di controllo, si è visto che lo stemming ha portato a risultati tutt'altro che utili, non eliminando il problema delle parole con lo stesso significato ripetute tra le parole chiave estratte e limitando la conoscenza fruibile dalle altre poiché lo stemmming risultava troppo invasivo, non consentendo di capire il significato di alcune parole. Si è quindi deciso infine di non implementare questa operazione.

Alla fine delle fasi appena descritte e riassunte tramite l'esempio in Figura **30** otterremo un elenco di parole necessario per la fase successiva. Questo elenco prende il nome di bag of words.

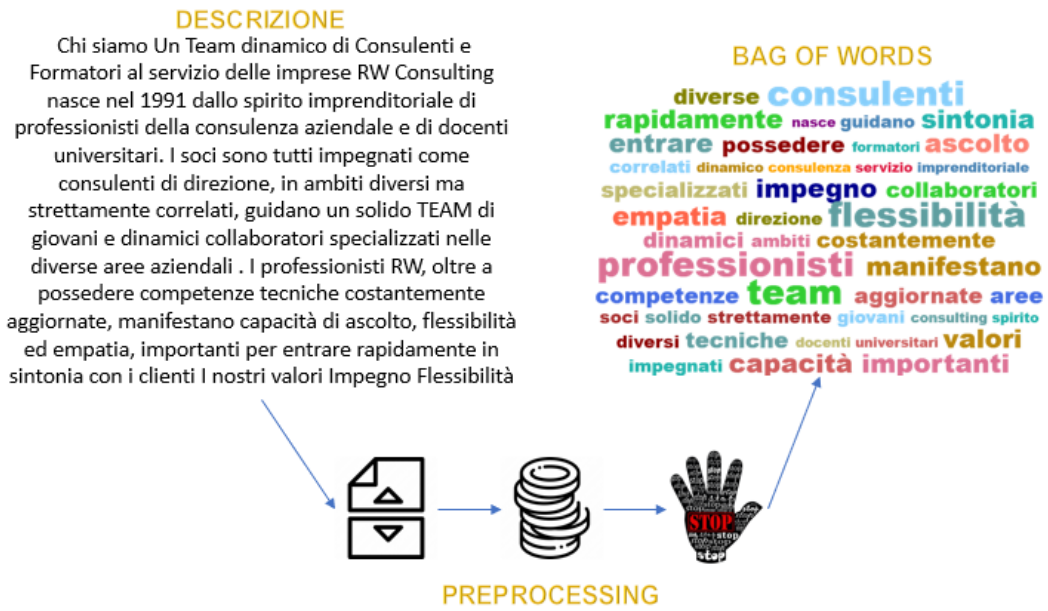

FIGURA 30-CREAZIONE BAG OF WORDS DA TESTO

4.2.4.2 Metodi utilizzati per l'estrazione delle parole chiave

Una volta ultimata la parte relativa allo scraping dei dati dai siti web delle aziende si è potuta iniziare la fase di keyword extraction. Questa prevede che per ogni campo di testo valorizzato durante le operazioni svolte precedentemente vengano estratte parole chiave che possano descrivere tale testo nel modo più accurato possibile e possano catturare informazioni riguardanti l'azienda e renderle visibili istantaneamente. In una prima fase di implementazione sono stati testati diversi algoritmi, disponibili in [10] ma dopo una prima analisi dei risultati si è constatato che solo tre di questi portavano a risultati accettabili e meritavano una seconda analisi più approfondita. I tre algoritmi emersi di cui descriveremo brevemente il funzionamento e come essi sono stati utilizzati all'interno del software sono:

1) TF-IDF

   L'acronimo sta per "term frequency-inverse document frequency". L'algoritmo calcola la frequenza di ogni termine ed associa un peso a ogni parola in base appunto alla sua frequenza all'interno del testo. Ovviamente più la parola sarà presente all'interno de testo più il suo peso sarà alto. Il valore del peso verrà poi normalizzato ovvero diviso per il numero di parole presenti nell'intero documento. L'algoritmo tiene altresì conto del fatto per cui una parola che appare troppo spesso in una collezione di documenti non è caratteristica di un documento quindi il peso cresce in maniera inversamente proporzionale con la frequenza del termine nella collezione. Nel nostro caso non abbiamo una collezione di documenti quindi si riconosce che tale algoritmo non viene sfruttato come potrebbe. La libreria di Python utilizzata per implementare il metodo è sklearn da cui importiamo TfidfVectorizer. L'utilizzo della libreria è spiegato in [11].

2) BERT

   Mentre gli altri algoritmi di keyword extraction lavorano principalmente basandosi su informazioni statistiche delle parole all'interno del testo, BERT si basa più sulla similarità semantica delle parole. BERT è un modello di trasformatore bi-direzionale che ci permette di mutare frasi e documenti in un vettore e di catturare il loro significato. L'algoritmo che è stato implementato, come si vede nello studio [12], prevede che,

inizialmente siano state selezionate le parole candidate ad essere keywords tramite la semplice funzione CountVectorizer() che converte un documento di testo in una matrice di numeri che rappresentano il numero di volte che la parola appare nel testo. Ovviamente la posizione del numero nella matrice corrisponde a quella della relativa parola all'interno del testo. All'interno del metodo, tramite l'attributo n_gram_range è stato possibile specificare la lunghezza delle frasi che volevamo estrarre, trattandosi di keywords si è specificato n_gram_range=(1,1). Il passaggio successivo è quello per cui si trasforma sia il documento che le parole candidate trovate in dati numeri è quello di creare un word embeddings con BERT. Un word embeddings è un tipo di rappresentazione delle parole che accumuna le parole con significati simili tra loro. Tale rappresentazione è stata fatta tramite la libreria di Python chiamata sentence-transformer a cui passeremo come parametro un modello BERT pre-addestrato chiamato distilbert-base-nli-stsb-mean-tokens. Nella fase finale seleziamo quei candidati che sono più simili al documento assumendo che i candidati più simili siano le keywords che stiamo cercando. Per calcolare tale somiglianza usiamo CosineSimilarity.

3) YAKE

   YAKE [13] è un algoritmo di estrazione di keywords non supervisionato e rapidamente applicabile, grazie alla sua natura plug and play, su un testo scritto in diverse lingue. L'algoritmo funziona basandosi sulle caratteristiche locali del testo e su informazioni statistiche, come la co-occorrenza e la frequenza dei termini. Uno dei vantaggi dell'algoritmo è che non ha bisogno di un corpus (collezione di testi selezionati e organizzati per facilitare le analisi linguistiche) di training. In secondo luogo, l'algoritmo è studiato per lavorare su un singolo documento, come il nostro caso, e non su una collezione di essi su cui di solito lavorano tanti altri metodi. Si ricordi infatti che tf-idf è pensato per lavorare su una collezione di documenti e noi lo abbiamo adattato ad un singolo documento. L'implementazione di questo metodo era già presente all'interno delle librerie di Python quindi per arrivare al risultato è bastanto importare KeywordExtractor dal package yake.

4.2.4.3 Analisi dei risultati

Per valutare quale dei metodi utilizzati è stato più performante sono stati analizzati i risultati che ne sono scaturiti. Il metodo di valutazione si è basato su degli indicatori noti in letteratura ovvero le metriche Precision, Recall e F-score che definiamo brevemente:

Siano:

- x = parole identificate sia dall'algoritmo che manualmente
- y = parole estratte dall'algoritmo ma non manualmente
- z= parole estratte manualmente ma non dall'algoritmo

Definiamo:

$Precision = \frac{x}{x+y}$, $Recall = \frac{x}{x+z}$ e $Fscore = \frac{2*Precision*Recall}{Precision+Recall}$

L'analisi è stata condotta prendendo come riferimento il testo della descrizione di cinque aziende, dal quale sono state estratte manualmente le parole chiave che meglio lo descrivevano, e le rispettive keywords estratte con i tre metodi sopradescritti. Si riporta il risultato in Tabella **1**.

TABELLA 1-OUTPUT KEYWORDS EXTRACTION

AZIENDA	TF-IDF	BERT	YAKE	BENCHMARK
AZIENDA 1	tecnologie unix competenze fornire integrator verso skill sistema servizi ricerca quali prodotti network vision mission soluzioni innovazione innovators energy conoscenza	cultura sperimentiamo soluzioni informatiche realizzati consapevolezza innovazione innovativa sperimentazione conoscenza innovators universitari specializzazioni industria eccellenza tecnologie software tecnologica tecnologici unix	integrator fornire unix tecnologie competenze software innovators system skill conoscenza sistema ricerca stato mondo prodotti cliente soluzioni network verso quali	Software Innovators tecnologica innovativa unix System Integrator Skill ict Informatiche industria
AZIENDA 2	produzione progettazione wireless sistemi hw fw test	industrializzazione telematiche collaborazione progettuali dinamiche apparecchiatura migliorare	produzione progettazione wireless sistemi elettronica prodotto apparecchiature	Progettazione Produzione apparecchiature elettroniche custom Monitoraggio Sensori

	apparecchiature elettronica prodotto temperatura ricerca prodotti custom grado azienda tutte network sensori assistenza	progettazione pluridecennali monitoraggio politecnico organizzazione specializzate microprocessori specializzato ferroviarioprogettazione ottimizzazione wireless ledprogettazione velocitàprogettazione	test custom ricerca azienda grado prodotti temperatura sensori innovazione tecnologica network sviluppo servizio	Wireless manifatturiera tecnologia hw fw
AZIENDA 3	web app software native applications mobile net development android apps custom asp ecommerce market ios different engineers eg windows erp	billion engineers web usnilosexperts websites swift plugins innovative php profitable app java ecommerce californian html5 innovation ios apps javascript android	web native app software applications mobile ios android custom development market asp net ecommerce usnilosexperts languages javascript apps trends real	Custom Software development innovation native iOS Android web applications coding app mobile connection integration
AZIENDA 4	team creiamo vuoi creare lavorando italia formano facilita digitali designer dentro dedicato condividendo marchi cliente chiamiamo chiamalo atmosfera analogici amore	ambizione analogici creiamo dentro multiculturale agenzia miglior sviluppatori sviluppo condividendo squadra principi ambienti atmosfera chiamiamo chiamalo personalizzate torino soluzioni italia	agenzia chiamiamo team creiamo software sviluppo uffici torino italia dedicato sviluppatori designer narratori problem solver soluzioni personalizzate marchi ambienti digitali	Software Sviluppo soluzioni digitali analogici team
AZIENDA 5	believe customers value team business customer main management services	integration automotive humanity contacts customer consulting customers partner telecommunications	technology main team business services staff added passionate creative	technology data telecommunications finance automotive Activa team business ICT

	activa technology national culture bpo one center companies contact creative crm	finance innovative technologies companies competitive business microsoft global professionality proactive innovation	proactive firmly preserving humanity culture positively impacts products deliver national players	BPO ITO IT

Dato che in ciascuno dei tre metodi era possibile specificare il numero di keywords che essi avrebbero dovuto restituire si è ritenuto opportuno verificare, oltre a quale fosse l'algoritmo che riportava risultati migliori, anche il numero di keywords che esso doveva generare per raggiungere l'ottimo. Tale analisi è stata valutata secondo la metrica F-score, combinazione delle altre due. A tal proposito, per ogni algoritmo e separatamente per ogni azienda, si è calcolato l'F-score rispettivamente per 3, 5, 7, 8, 9, 10 e 20 keywords. Si è fatta successivamente la media prendendo in considerazione ogni azienda e si sono ottenuti i risultati presenti in Figura **31** per i tre algoritmi.

TF-IDF (n° parole)	AVG FSCORE %
3	25,66
5	36,28
7	38,65
8	40,47
9	44,81
10	44,52
NO VINCOLI	40,55

YAKE (n° parole)	AVG FSCORE %
3	30,99
5	42,27
7	51,17
8	52,45
9	55,53
10	54,53
NO VINCOLI	44,08

BERT (n° parole)	AVG FSCORE %
3	9,46
5	10,76
7	13,90
8	15,12
9	19,14
10	22,02
NO VINCOLI	29,75

FIGURA 31-RISULTATI BERT, YAKE E TF-IDF PER NUMERO DI KEYWORDS

Come si evince dai risultati ottenuti l'algoritmo che presenta un risultato migliore è Yake con un numero di parole da estrarre pari a nove, si riporta, oltre all'andamento del punteggio al variare del numero di parole, visibile in Figura **32**, anche il singolo punteggio per ogni azienda, visibile in Figura **33**, da cui si evince

che le aziende responsabili del sensibile abbassamento della media generale sono quelle denominate Azienda 4 e Azienda 5.

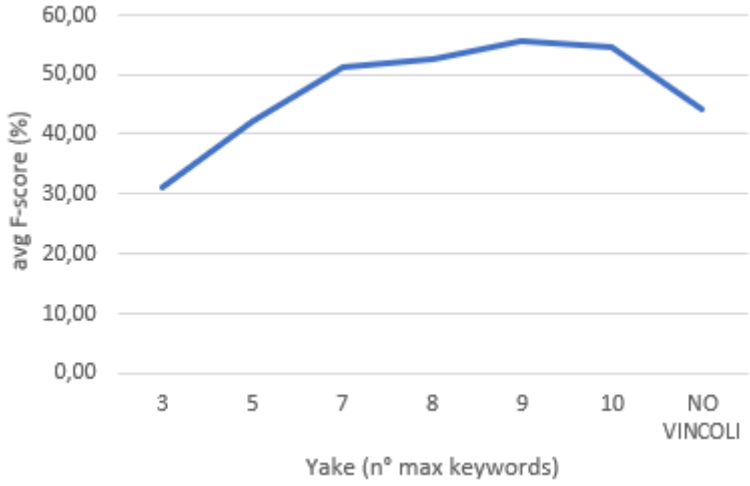

FIGURA 32-RELAZIONE TRA F-SCORE E NUMERO DI KEYWORDS

YAKE RESULT	
AZIENDA	F-SCORE %
AZIENDA 1	73,6842105
AZIENDA 2	57,1428571
AZIENDA 3	78,2608696
AZIENDA 4	40
AZIENDA 5	28,5714286

FIGURA 33-RISULTATI YAKE PER AZIENDA

Si è a questo punto considerato che fosse opportuno cercare di capire la motivazione quantitativa dei bassi risultati conseguiti per queste specifiche due aziende, per questo sono state fatte ulteriori valutazioni. Una di queste per mezzo della analisi del sentimento e della oggettività sul testo descrizione. L'analisi del sentimento è un campo dell'analisi del linguaggio naturale che valuta l'accezione di un testo, in particolare il livello di neutralità, positività o negatività. Si è pensato infatti che una descrizione poco tecnica dell'azienda, poco oggettiva, e quindi con una positività spiccata limitasse la bontà della estrazione delle parole chiave dal testo. Un'altra correlazione cercata è stata che coinvolgeva la lunghezza del testo con la poca bontà dell'algoritmo ma come si

evince dalla Tabella **2** non si sono riscontrati risultati che dimostrassero la sua esistenza.

TABELLA 2-RELAZIONE TRA F-SCORE E LUNGHEZZA TESTO

F-SCORE	PAROLE TESTO
73,68%	218
57,14%	498
78,26%	105
40,00%	48
28,57%	168

Dai grafici in Figura **34** e in Figura **35** si può notare facilmente che non possiamo stabilire una correlazione certa tra le due variabili. Quello che si può dire certamente è che tutte le aziende presentano un testo descrizione positivo e questo è comprensibile dato che risulta impensabile che una azienda scriva una descrizione negativa nei suoi stessi confronti. Ulteriormente possiamo affermare che l'azienda (Azienda 3) che presenta il testo meno positivo riporta il punteggio maggiore di F-score. Per quanto riguarda la soggettività del testo i risultati, presenti in Figura **35**, sono ancora meno marcati di quelli dell'analisi del sentimento. Si può solamente notare che il punteggio più basso di F-score, associato all'Azienda 5, presenta una soggettività più alta, anche se in misura molto ridotta, delle altre.

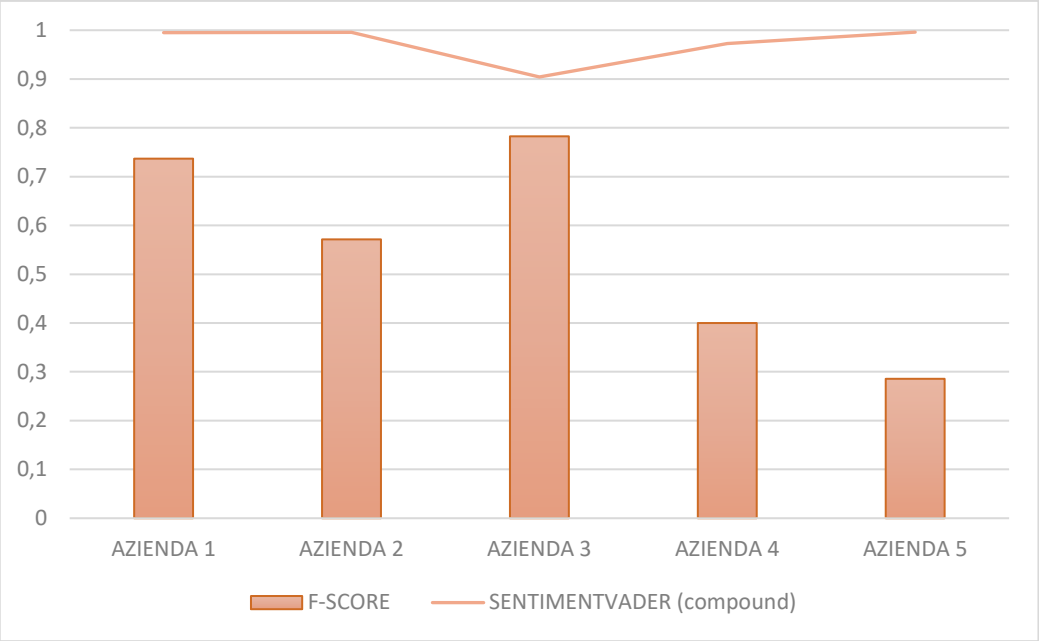

FIGURA 34- RELAZIONE TRA F-SCORE E POSITIVITÀ DEL TESTO

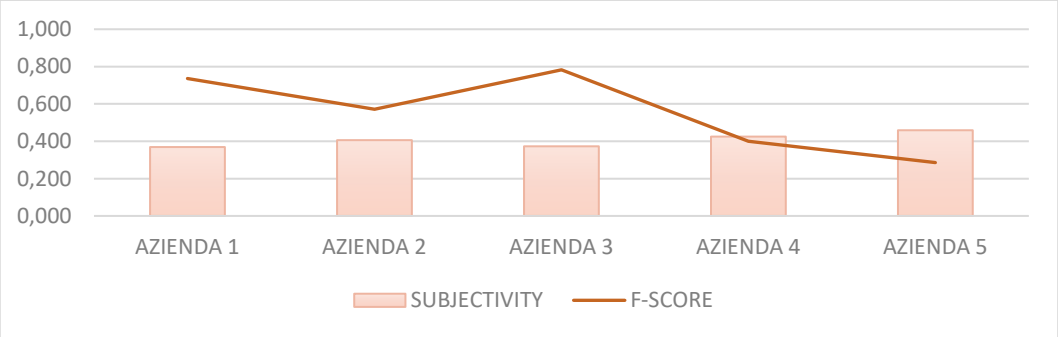

FIGURA 35-RELAZIONE TRA F-SCORE E SOGGETTIVITÀ

Una volta estratte le keywords è stata implementata una operazione di POS tagging. L'acronimo sta per Part Of Speech tagging ed è definito come il processo per cui si riesce ad etichettare una parola del testo con il suo significato grammaticale corrispondente all'interno della frase. L'etichettatura viene fatta considerando sia la definizione della parola, sia il suo contesto all'interno della frase in cui si trova. Per questo tale operazione è stata fatta passando come input direttamente l'interno testo, prima che venisse implementata la parte di preprocessing. La libreria protagonista di questa parte di implementazione, come si vede dalla Figura **36** è spacy. Dato che i testi passati come input potevano essere sia in lingua inglese che in lingua italiana abbiamo utilizzato un'altra libreria, chiamata langdetect, in grado di capire appunto la lingua del testo sorgente. Una volta appurata quest'ultima è stato caricato il modello con il

metodo spacy.load("model name"). In questo caso i modelli utilizzati sono stati, come suggerisce la libreria di riferimento reperibile in [14], "en_core_web_sm" per i testi inglesi e "it_core_news_sm" per i testi in italiano. Entrambi i modelli presentano all'interno della documentazione ufficiale dati relativi ad una accuratezza di novantasette punti percentuali.

```
def do_spacy(text_):
    try:
        lang = detect(text_)
        lista_verbi = []
        if lang == 'it':
            sp = spacy.load('it_core_news_sm')
        else:
            sp = spacy.load("en_core_web_sm")
        doc = sp(text_)
        for token in doc:
            if token.pos_ == 'AUX' or token.pos_ == 'VERB':
                lista_verbi.append(str(token))
        return lista_verbi
    except:
        return []
```

FIGURA 36-CREAZIONE LISTA VERBI DA TESTO

Questa operazione ha permesso di aumentare il valore delle metriche valutative dell'algoritmo di keyword extraction poiché è stata fatta una sottrazione tra due insiemi, quello di tutte le parole chiave e quello dei verbi delle parole chiave, che ha permesso in particolare di aumentare il valore della variabile indicata precedentemente come y, che ricordiamo essere quella che esprime il numero di parole estratte dall'algoritmo ma non manualmente, con il conseguente aumento del valore di precision e quindi di quello dell'F-score. Si riportano i risultati ottenuti in Figura **37**-risultati aggiornati f-score con yake e 9 keywordsFigura **37**.

YAKE- 9 PAROLE - VERBI							
AZIENDA	x	y	z	RECALL %	PRECISION %	FSCORE%	MEDIA F SCORE
AZIENDA 1	7,00	1,00	3,00	70,00	87,50	77,77777778	
AZIENDA 2	6,00	3,00	6,00	50,00	66,67	57,14285714	
AZIENDA 3	9,00	0,00	5,00	64,29	100,00	78,26086957	57,86707013
AZIENDA 4	3,00	4,00	3,00	50,00	42,86	46,15384615	
AZIENDA 5	3,00	5,00	9,00	25,00	37,50	30	

FIGURA 37-RISULTATI AGGIORNATI F-SCORE CON YAKE E 9 KEYWORDS

4.2.5 Creazione del dataset

Una volta estratte tutte le informazioni di nostro interesse storicizziamo tutte le informazioni all'interno di un database. Tra le varie possibilità si è deciso di salvare i dati raccolti in un database locale di tipo MySQL in modo che questo possa essere esportato e condiviso facilmente. A tal proposito importiamo la libreria python necessaria alla connessione con il database che si chiama mysql.connector. Questa libreria mette a disposizione la funzione mysql.connector.connect() che ci consente di connetterci con il database indicando i campi host, user, password e nome del database. Una volta stabilita la connessione inseriamo un record all'interno del database. Il metodo verrà ovviamente ripetuto per ogni azienda presente nel database iniziale di input. Alla fine dell'inserimento è doveroso chiudere la connessione al database. L'implementazione del codice relativo a questa fase è visibile nella Figura **38**.

```
import mysql.connector
db = mysql.connector.connect(host="127.0.0.1", user="root",
                             password="", database="outputscraper")
def insert_record(db_, url_, descrizione_, kw_descrizione_, competenze_, lista_competenze_,
                  kw_competenze_, casi_applicativi_, lista_casi_applicativi_,
                  kw_casi_applicativi_, progetti_, lista_progetti_,
                  kw_progetti_, clienti_, partner_, network_, contatti_, lat_lng_,
                  lat_lng_google_, count_sedi_estere_,
                  sedi_estere_, facebook_, instagram_, twitter_, linkedin_):
    cursor = db_.cursor()
    query = "insert into output_scraper (url, descrizione, kw_descrizione," \
            "competenze, lista_competenze,kw_competenze," \
            "casi_applicativi,lista_casi_applicativi, kw_casi_applicativi, " \
            "progetti, lista_progetti, kw_progetti, " \
            "clienti, partner, network, contatti, lat_lng, lat_lng_google, " \
            "count_sedi_estere, sedi_estere, facebook, " \
            "instagram, twitter, linkedin ) " \
            "values(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, "\
            "%s, %s, %s, %s, %s, %s, %s) "
    record = (url_, descrizione_, kw_descrizione_, competenze_,
              lista_competenze_, kw_competenze_, casi_applicativi_,
              lista_casi_applicativi_, kw_casi_applicativi_, progetti_,
              lista_progetti_, kw_progetti_, clienti_,
              partner_, network_, contatti_, lat_lng_, lat_lng_google_,
              count_sedi_estere_, sedi_estere_, facebook_,
              instagram_, twitter_, linkedin_)
    cursor.execute(query, record)
    db_.commit()
db.close()
```

FIGURA 38-INSERIMENTO DEI UN RECORD NEL DATABASE

4.2.6 Machine Learning e Classification

In questa ultima fase di articolazione del lavoro ci si è proposti di eseguire un algoritmo di machine learning così da poter sfruttare al meglio il dataset appena creato, estraendo ancora più informazioni di quelle già ottenute dalle fasi precedente. Prima di entrare nello specifico del procedimento utilizzato si propone una panoramica importante per apprendere al meglio i concetti di cui parleremo in seguito.

4.2.6.1 Machine Learning

Il tema del machine learning si colloca nel contesto dell'intelligenza artificiale ovvero il campo in cui un sistema deve essere in grado di adattarsi ai possibili cambiamenti dell'ambiente in cui si trova, senza che ci sia un supervisore che immagini e controlli tutti i possibili scenari. Il machine learning nasce dal bisogno

e il tentativo di voler superare la visione per cui risultasse impossibile che una macchina apprendesse qualcosa basandosi su esperienze o errori commessi in passato. Se questo fosse possibile, la raccolta di una grande quantità di dati avrebbe una importanza fondamentale in ogni campo di business poiché sarebbe possibile usarli per fare una predizione ovvero raccogliere informazioni che prima non erano disponibili. A tal proposito, si tenta di riconoscere dei pattern, delle caratteristiche comuni di una collezione di dati per creare un sistema che possa creare una approssimazione di un risultato che possa essere utile per spiegare o capire una parte dei dati.

Vi sono principalmente tre tipologie di algoritmi di machine learning come si evince dallo studio effettuato in [15] e [16]:

1) Apprendimento supervisionato (supervised learning): è quello in cui gli algoritmi devono, dato in input, fare una predizione corretta dell'output. È quello che comunemente è chiamato problema di Classificazione, nonché quello che affronteremo in questo caso studio. Il caso più semplice è il problema binario ovvero la classificazione dei dati secondo un'etichetta che può avere come valore solo due risultati, uno l'opposto dell'altro.
2) Apprendimento non supervisionato (unsupervised learning): serve per avere informazioni su come sono strutturati i dati. Il problema in questione è quello relativo alla clusterizzazione. L'obbiettivo di questo processo è quello di fare data exploration e data analysis, operazioni necessarie nel momento in cui siamo dotati di un dataset molto grande ma non è ancora ben chiaro come esso sia strutturato e quale sia l'obbiettivo di business, ovvero a cosa potrebbe essere utile il dataset. Per capire questo sarebbe opportuno riuscire a trovare un partizionamento dei dati in modo che si riescano a individuare i gruppi di dati simili tra loro e divedere questi da quelli che invece non sono simili.
3) Apprendimento di rinforzo (reinforcement learning): è il tipo di machine learning che deve osservare e imparare una linea di condotta specifica per poi saperla riprodurre. Un esempio di applicazione di questo tipo di algoritmo è quello relativo alla guida autonoma.

Le categorie appena descritte sono, nei casi reali, difficili da distinguere infatti vi sono molte applicazioni che si collocano simultaneamente in più categorie come

il così detto apprendimento semi-supervisionato che prevede sia la parte di classificazione che quella di clustering.

Tutti i più noti algoritmi di machine learning sono implementati in linguaggio Python e facilmente accessibili tramite una libreria chiamata scikit-learn e spesso abbreviata con sklearn. Questa è una libreria open source di apprendimento automatico che contiene algoritmi di classificazione, regressione, clustering e macchine a vettori di supporto. Si riporta una mappa in Figura **39**, scaricata da [17] che rappresenta tutte le varie sezioni della libreria e il campo in cui esse vengono applicate.

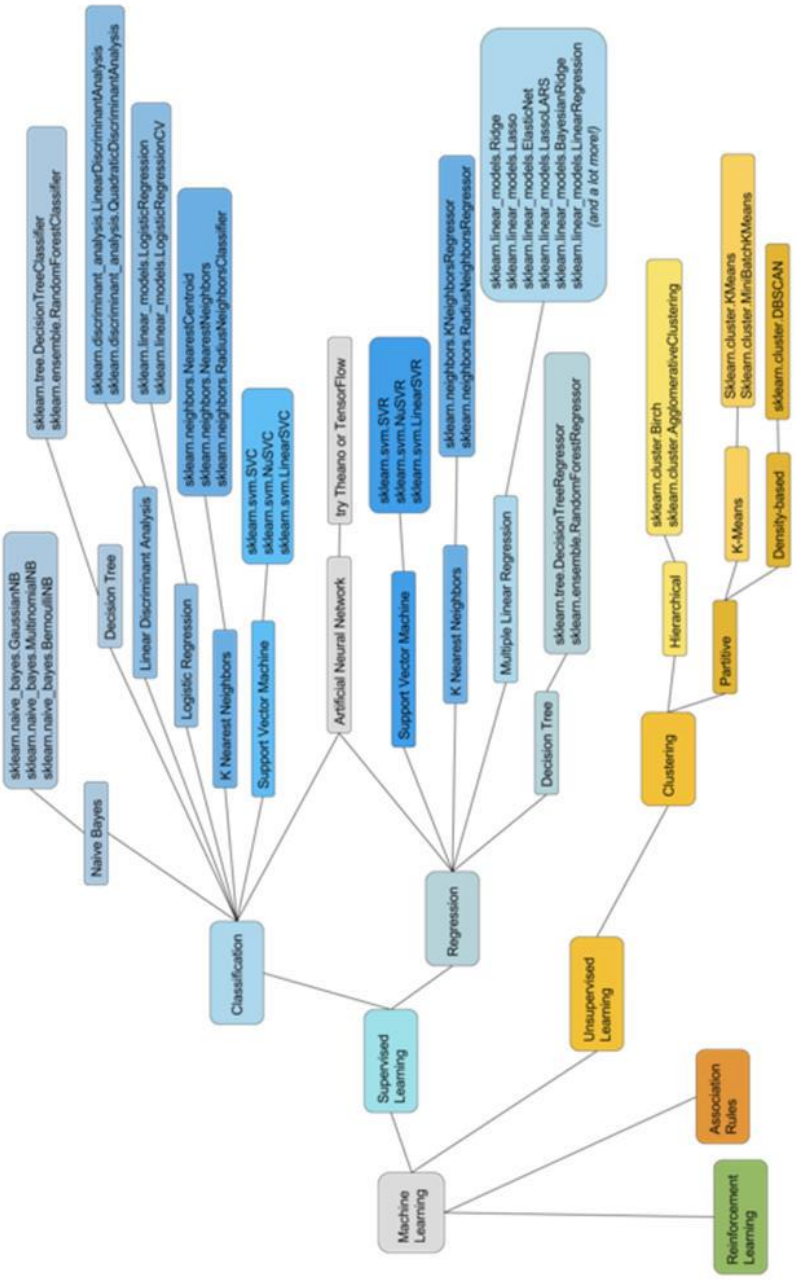

FIGURA 39 – MAPPA DI RELAZIONE TRA ALGORITMI DI MACHINE LEARNING E MODULI DELLA LIBRERIA SKLEARN

4.2.6.2 Classification

L'apprendimento supervisionato o classificazione ha come obbiettivo principale quello di insegnare alla macchina il modo in cui si è costruito un modello di classificazione con dati di input e output noti in modo che essa lo possa riproporre e implementare su nuovi dati, quest'ultima operazione si chiama previsione. Quindi, data una collezione di etichette di classe e una collezione di

dati di input già etichettati si vuole trovare un profilo descrittivo che ci permetta di fare una previsione, ovvero un etichettamento dei nuovi dati, appropriata. In questo tipo di modelli vi è sempre un training set e un test set preso dalla stessa sorgente di dati di input. Il training set è l'insieme di dati già etichettati sul quale ci si basa per creare il modello che andrà ad etichettare i nuovi dati. Il test set è quello che serve per validare tale modello.

Vi sono numerosi algoritmi che si possono utilizzare per implementare il processo di classificazione ma non è mai noto a priori quale di esso sia il più performante perché dipende esclusivamente dal tipo di dati che abbiamo. Le metriche che vengono usate per verificare la performance di questi algoritmi sono:

- Accuratezza: valuta la qualità della predizione e come è descritto in figura si calcola con il rapporto tra il numero di record che sono stati etichettati correttamente dal modello ed il numero di record processati.
- Efficienza: indica il tempo di creazione del modello e della sua applicazione
- Scalabilità: indica la capacità di creare il modello all'aumentare del numero di dati in input e dei loro attributi
- Robustezza: è la capacità del modello di gestire i dati mancanti o che generano errori, i così detti outliers
- Interpretabilità: la caratteristica che rende il modello oltre che efficiente anche comprensibile dall'analista

Anche per quanto riguarda le metriche di valutazione del classificatore sklearn fornisce numerosi metodi per il loro calcolo, visibili in Figura **40**.

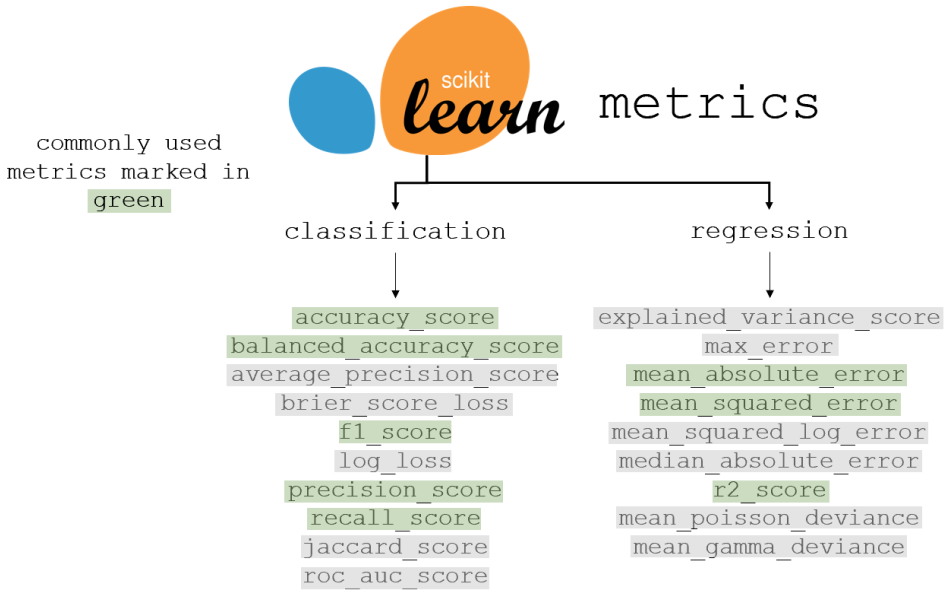

FIGURA 40- METRICHE PER VALUTARE GLI ALGORITMI DI CLASSIFICAZIONE

Allo stato dell'arte vi sono numerose famiglie di algoritmi dedicati al task della classificazione, di seguito ne elencheremo alcuni e ne definiremo brevemente il funzionamento.

- Decision tree: tramite questo algoritmo il set di dati è diviso secondo degli attributi del record detti attributi di splitting. In fase finale verranno creati dei percorsi che ci porteranno alla decisione di etichettare un record in un modo piuttosto che in un altro. Gli attributi di splitting possono essere nominale, ordinale o continuo e lo split può essere binario quando effettua due partizioni dei dati o multiway splitting quando divide i dati in più parti. I vari algoritmi appartenenti alla famiglia del decision tree si differenziano proprio in base al tipo di attributi di splitting e al tipo di splitting vero e proprio. I vantaggi principali di questo algoritmo sono che l'albero delle decisioni si costruisce in tempi molto ridotti e che il modello che si genera risulta interpretabile. Lo svantaggio è che le performance calano drasticamente se si hanno dei missing values ovvero la presenza di attributi con valore non noto.
- Random forest: rappresenta un'estensione del decision tree. Questo presenta tendenzialmente performance migliori, oltre che una maggiore robustezza in presenza di valori degli attributi mancanti, del decision tree a fronte di un tempo di esecuzione più lungo e di una minore interpretabilità. Il suo funzionamento prevede che il dataset iniziale venga

diviso in più dataset in modo randomico e che su ognuno di questi venga eseguito un algoritmo di decision tree che etichetterà il record. Dopo questa operazione si effettua un processo di majority voting per selezionare la classe finale data l'unione di quelle scaturite da tutti gli alberi di decisione creati.

- Naive Bayes: questo tipo di classificatori si basano sul concetto di calcolo delle probabilità condizionate ovvero la probabilità che si verifichi un evento particolare condizionatamente al fatto che se ne presenti un altro. La sua implementazione è fattibile solo sotto l'ipotesi di Naive per la quale gli attributi dovrebbero essere statisticamente indipendenti. Inoltre, le probabilità condizionate devono essere note o quantomeno stimabili. Sotto queste condizioni l'algoritmo risulta robusto ed efficiente.
- Reti neurali: sono algoritmi che presentano risultati di classificazione ottimi in dataset molto complessi. Sono usati prevalentemente nella classificazione di file di testo, immagini, audio o video. Il lato negativo è che ci vuole molto tempo per creare il modello sul training set e che i risultati non sono facilmente interpretabili.
- Support Vector Machine: è una tecnica, come viene affermato in [18] che viene usata molto nell'ambito della classificazione dei dati testuali. Gli algoritmi di questo tipo ottengono la loro massima efficacia in problemi di classificazione binari ma sono usati spesso anche per risolvere i problemi di classificazione multiclasse. L'algoritmo è basato sull'idea di dover trovare un iperpiano che sia in grado di dividere un set di dati in due classi distinte. Un iperpiano per un problema di classificazione binario è raffigurato come una linea che divide i dati in due gruppi corrispondenti alle classi di appartenenza. I vettori di supporto sono i record, appartenenti alle due diverse classi, che si trovano più in prossimità dell'iperpiano e il margine è definito come la distanza tra i due vettori di supporto, a metà di questa distanza si traccia l'iperpiano. Gli elementi appena descritti si possono trovare in Figura **41**.

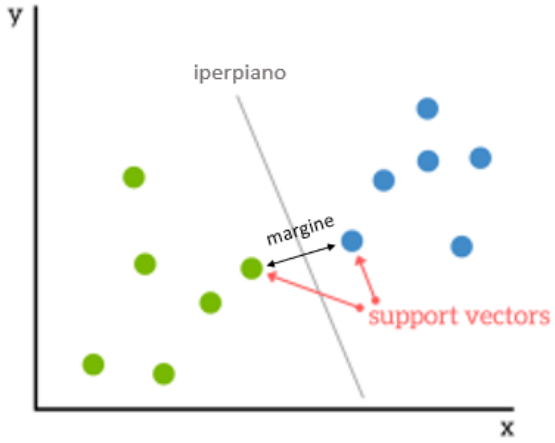

FIGURA 41 – IPERPIANO, MARGINE E VETTORI DI SUPPORTO

Il support vector machine ha come obbiettivo quello di trovare l'iperpiano che meglio divide i due vettori di supporto appartenenti alle due classi e questo si verifica nel momento in cui il margine viene massimizzato. Se l'iperpiano non esiste l'algoritmo trasforma i dati di training in una dimensione superiore così che i dati possano sempre essere separati da un iperpiano. L'algoritmo trova le sue fondamenta nelle operazioni di algebra lineare.

4.2.5.3 Applicazione al caso studio

L'obbiettivo del lavoro è quello di classificare le aziende di cui abbiamo estratto le informazioni tramite il web scraping, che successivamente sono state arricchite tramite l'operazione di keyword extraction, in base al loro settore industriale di appartenenza. Ricordiamo che per creare un modello di machine learning si ha bisogno di un training set ovvero un insieme di dati in cui il risultato dell'etichetta di classe risulti già noto. Per questo motivo è stata effettuata una prima ricerca sul web per verificare se fossero presenti dei dataset di training per allenare e poi testare il modello, adattandolo al nostro caso studio.

In una fase successiva è stato possibile ottenere un training set dei dati più pertinente poiché fornito direttamente dall'azienda. Il dataset contiene 300 aziende opportunamente classificate secondo i criteri dell'azienda stessa.

4.2.5.3.1 Dataset "Company Name Classification"

In una prima fase dello sviluppo non si era in possesso di un dataset che potesse costituire il training set quindi si è cercato in rete un dataset che potesse essere

in linea con il nostro obbiettivo di business. A tal proposito abbiamo sfruttato Kaggle (https://www.kaggle.com/). Kaggle è una community online di data scientists che, tra le sue altre funzionalità, mette a disposizione datasets pubblici tipicamente volti a risolvere problemi di machine learning. Al suo interno (https://www.kaggle.com/thecobbler/classifying-company-names-as-per-their-industries#Table-of-Contents:) abbiamo trovato una serie di dati ed un algoritmo di machine learning calzante con il nostro obbiettivo di business. Il dataset, in lingua inglese e contenente 2002 record, è strutturato in modo che ogni record rappresenti una azienda, il record è caratterizzato dai seguenti attributi:

- Company name: rappresenta il nome dell'azienda
- Exchange:Ticker: rappresenta il simbolo ticker dell'azienda ovvero l'abbreviazione utilizzata per identificare in modo univoco le azioni quotate in borsa di un determinato titolo su un determinato mercato azionario
- Company type: rappresenta il tipo di azienda, nel dataset i valori associati a questo attributo sono per il 100% uguali a "Public Company"
- Company status: rappresenta lo stato dell'azienda, nel dataset il 100% dei record ha come valore di questo attributo "Operating"
- Geographic locations: rappresenta lo stato geografico di appartenenza dell'azienda. Nel caso particolare tutte le aziende hanno questo attributo valorizzato come "United States of America"
- Business description: questo attributo rappresenta una descrizione dell'azienda
- Industry classification: rappresenta una lista di settori industriali in cui l'azienda si colloca

L'algoritmo di classificazione implementato su questo dataset prende come input la descrizione aziendale e come output cerca di etichettare le varie aziende secondo il loro settore industriale, sottolineiamo che tra la lista di settori industriali relativi ad una azienda presenti nel campo "Industry Classification" si è deciso di prendere in considerazione il primo della lista, essendo quello il più pertinente e gli altri superflui. Le fasi dell'implementazione del codice sono 4:

1. Preparazione del dataset: in questa fase si carica il dataset all'interno del programma usando la libreria pandas di Python e si eseguono gli step basilari di preprocessing sul testo di input in corrispondenza dell'attributo "Business Description".

2. Trasformazione dei dati: il secondo step è quello in cui si trasformano i dati grezzi in dati che possono essere usati per implementare un modello di machine learning. Questa fase include anche il processo per cui si vanno a creare nuove colonne che descrivono l'azienda a partire dai dati iniziali. In particolare, viene aggiunta una colonna denominata "Tiny_Desc" nel quale salveremo il nuovo testo, scaturito dalla precedente fase di preprocessing.
3. Creazione, addestramento e risultati del modello: è la fase in cui, una volta creato il modello, lo si allena sui dati iniziali già etichettati, si fa il test dei dati e si calcola una misura di accuratezza del modello. Il modello è stato implementato attraverso il classificatore messo a disposizione dal package di Python sklearn da cui abbiamo importato OneVsRestClassifier e LinearSVC. One vs Rest è un metodo euristico per l'utilizzo di algoritmi di classificazione binaria per la classificazione multi-classe che si collocano all'interno della famiglia dei modelli di apprendimento delle macchine a vettori di supporto (SVC – Support Vector Classifier) ampliamente trattati nella sezione 4.2.5.2. Il suo funzionamento prevede che si divida il dataset inziale di dati multi-classe in più problemi di classificazione binaria. Il classificatore viene allora addestrato su ogni problema binario e le previsioni vengono effettuate seguendo il modello più affidabile.

Dalla valutazione dei risultati, il modello presenta un'accuratezza del modello pari circa all'80% il che significa che su 100 previsioni, 80 sono state effettuate nel modo corretto. Per capire le criticità del modello esso è stato studiato minuziosamente variando le implementazioni delle diverse fasi che lo compongono per valutare al meglio i risultati ottenuti e le relative variazioni. Si è a tal proposito implementato diversi approcci per la fase di preprocessing. Originariamente essa prevedeva:

- Trasformazione del testo in lettere tutte minuscole
- Rimozione delle stopwords inglesi ricavate tramite l'apposita libreria stopwords importata dal package nltk.corpus
- Rimozione dei numeri e degli spazi bianchi superflui all'interno del testo
- Rimozione della punteggiatura, degli accenti e di altri segni diacritici
- Tokenizzazione, stemming e lemmarization

Alterando questo processo di è visto che le fasi di stemming e lemmarization non influivano sulle prestazioni finali del modello e sono state quindi rimosse. Fondamentale invece la fase di rimozione della punteggiatura, accento e altri simboli implementata attraverso la linea di codice:

```
train_ana.Tidy_Desc=train_ana.Tidy_Desc.apply(lambda x: x.translate({ord(c):'' for c in "!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"}))
```

che, se rimossa, determina in abbassamento delle performance di due punti percentuali.

I record rimasti dopo l'eliminazione dei missing values all'interno del dataset sono 1991. Le classi iniziali possibili in cui i record sono etichettati risultavano invece essere 201. Il modello originale è implementato ammettendo solo 10 classi che sono:

- Banks
- Healthcare
- Biotechnology
- Energy
- Consumer Discretionary
- Information Technology
- Capital Goods
- Commercial and Professional Services
- Application Software
- Communications Equipment

In queste classi si collocano 1155 record. Sono invece eliminati dal dataset quei record non etichettati con nessuna delle 10 classi selezionate per un numero di 836 record. Alterando il numero di classi ammesse per l'etichettamento dei record si registra una relazione inversamente proporzionale tra l'aumento di esse e le prestazioni del modello, nel caso estremo in cui si ammettono tutte le 201 classi possibili, il modello risulta molto lento e presenta delle difficoltà a convergere oltre al pessimo risultato di accuratezza che si assesta intorno al 40%. Si riporta l'andamento della relazione tra numero di classi possibili e accuratezza del modello nel grafico in Figura **42**.

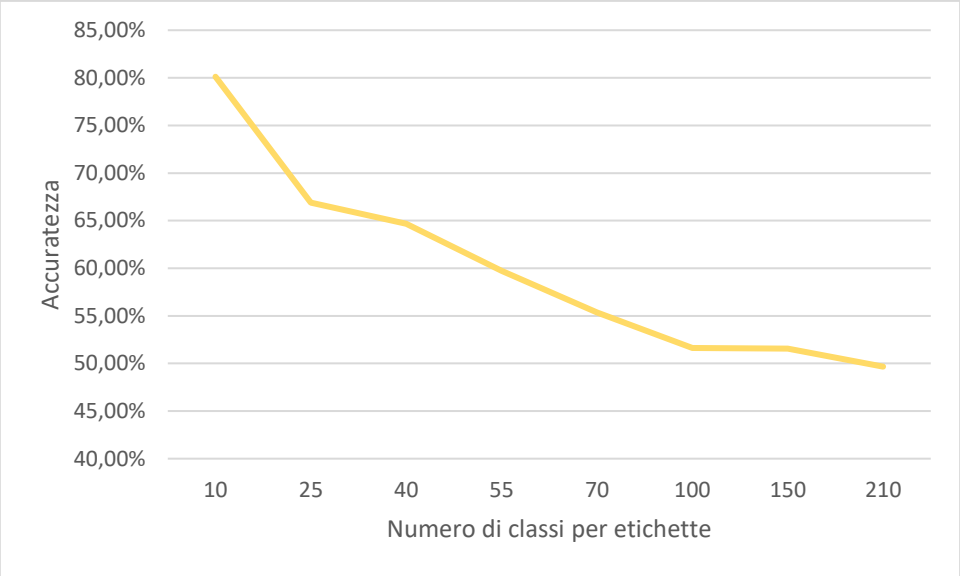

FIGURA 42-RELAZIONE TRA ACCURATEZZA DEL MODELLO E NUMERO DI CLASSI AMMESSE

4.2.5.3.2 Riadattamento dell'algoritmo al caso di business analizzato

Una volta analizzato il funzionamento dell'algoritmo appena descritto si è pensato di poterlo variare in modo da adattarlo e integrarlo con tutte le fasi di sviluppo trattate all'interno di questo capitolo relativo all'implementazione del software oggetto del progetto di tesi. A tal proposito si è pensato di cambiare i dati in input del modello aggiungendo le funzioni relative alla fase di estrazione delle keywords descritte nella sezione 4.2.4 all'algoritmo inziale e di usare le parole chiave estratte come dati di input per la classificazione. Si è poi applicato e testato il modello. I risultati delle metriche di valutazione testimoniano un abbassamento delle prestazioni intorno ai dieci punti percentuali, ovvero si è ottenuta un'accuratezza del modello di circa il 70%.

4.2.5.3.3 Previsioni sui dati non etichettati

Grazie alla descrizione dell'algoritmo di classificazione oggetto di questa sezione è facile notare che possiamo utilizzare il modello di classificazione riadattato per fare una previsione sul nostro dataset, creato nelle fasi precedenti la classificazione, la cui descrizione è presente nella sezione 4.2.5.

A tal proposito è stata necessaria l'integrazione di un'unica operazione supplementare ovvero quella di traduzione delle keywords estratte dal testo

descrizione delle aziende dalla lingua italiana a quella inglese poiché ricordiamo che il dataset scaricato da Kaggle è tutto in lingua inglese. Tale operazione è stata implementata, come si vede in Figura **43**, grazie ad una libreria Python denominata TextBlob.

```
def traduci_keywords(s_kw_no_verbi_descrizione_):
    s_kw_en = []
    for parola in s_kw_no_verbi_descrizione_.split(","):
        try:
            blob = TextBlob(parola)
            translation = str(blob.translate(to="en"))
        except textblob.exceptions.NotTranslated:
            translation = parola
        s_kw_en.append(translation)
    return " ".join(s_kw_en)
```

FIGURA 43-CODICE PER LA TRADUZIONE DELLE KEYWORDS INGLESI

La previsione è stata analizzata manualmente tramite un controllo sui siti web di 50 delle 300 aziende presenti nel dataset e si è riscontrato un risultato reale in linea con quello predetto dall'accuratezza del modello ovvero si è registrato un 30% di errore nell'etichettamento del settore delle aziende e quindi un'accuratezza delle previsioni del 70% data dal rapporto tra le 35 aziende etichettate correttamente e le 50 totali.

4.2.5.3.4 Dataset Torino Wireless – problema multi-classe

In un secondo momento, come anticipato nella sezione 4.2.5.3, sono state prese in considerazione le informazioni di classificazione che l'ente collaborante Fondazione Torino Wireless associa alle imprese del network, aggiungendo quindi al dataset usato nelle operazioni di implementazione del software fino ad ora descritte, un attributo denominato "Ambiti applicativi". L'attributo è valorizzato tramite un set di etichette che descrivono gli ambiti applicativi in cui l'azienda si colloca. La struttura del dataset è riportata in Figura **44**.

Ragione sociale	PIVA	Sito WEB	Ambiti applicativi
ragione sociale azienda 1	*p.IVA azienda 1*	http://	Automotive;Agroalimentare;Istruzione e Formazione
ragione sociale azienda 2	*p.IVA azienda 2*	http:/ .it	Chimica e Materiali

FIGURA 44-STRUTTURA DATASET TORINO WIRELESS

Tale classificazione è stata attribuita dagli analisti di Fondazione Torino Wireless alle aziende del network già note, spesso a fronte di una conoscenza puntuale delle attività della singola impresa e quindi in base ad informazioni non incluse nel relativo sito web, senza l'ausilio ed il supporto di algoritmo di intelligenza artificiale come quelli di machine learning; per questo motivo nel presente lavoro di tesi si è cercato di implementare un algoritmo di machine learning in modo che l'azienda collaborante avesse uno strumento per classificare le aziende presenti all'interno della propria rete secondo i principi che hanno guidato la classificazione manuale. Abbiamo scelto come dati di input del problema di classificazione le parole chiave estratte dal testo della descrizione di ogni azienda, ottenuto tramite l'operazione di web scraping descritta nella sezione 4.2.2. La colonna di keywords, come si vede in Figura **45** è stata aggiunta al dataset iniziale.

Ragione sociale	PIVA	Sito WEB	Ambiti applicativi	KW_DESCRIZIONE
ragione sociale azienda 1	*p.IVA azienda 1*	http://	Automotive;Agroalimentare;Istruzione e Formazione	team, consulenti, professionisti, flessibilità, dinamico, formatori, servizio, consulting
ragione sociale azienda 2	*p.IVA azienda 2*	http: it	Chimica e Materiali	rete, soluzioni, servizi, servizio, accesso, comunicazione, sistemi, costi

FIGURA 45-DATASET TORINO WIRELESS CON KEYWORDS

Come si afferma nello studio [19], il numero di record ottimale che deve essere presente nel training set non è definito a priori. Tale valore probabilmente non esiste ma sappiamo che dipende fondamentalmente da alcuni fattori quali la complessità del problema che stiamo affrontando, la funzione che deve collegare la variabile in input a quella in output e la complessità dell'algoritmo di machine learning. A valle di queste considerazioni abbiamo deciso di implementare un algoritmo di machine learning supervisionato per creare un modello e valutarne le relative performance anche se le dimensioni del database in nostro possesso risultano molto ridotte. Ricordiamo a tal proposito che il database, nelle sue dimensioni iniziali era composto da 300 record, dopo di che è stata effettuata una prima necessaria operazione di eliminazione dei record che presentavano dei missing values, ovvero valori nulli corrispondenti all'attributo "Ambiti applicativi" usato per l'applicazione dell'algoritmo e una seconda eliminazione di quei record che presentavano un valore inconsistente e povero di informazioni corrispondente all'attributo relativo alle parole chiave. L'eliminazione dei record appena descritti ha portato ad una ulteriore riduzione del numero di righe all'interno del database che è passato da 300 a 240.

Dati i confortanti risultati ottenuti e discussi nella sezione 4.2.5.3.5 si è pensato di rielaborare il dataset fornito da Torino Wireless appena descritto e trattarlo come un problema multi-classe applicando la stessa metodologia applicata per il dataset scaricato da Kaggle e descritta nella sezione 4.2.5.3.1. Il procedimento per arrivare al risultato finale è stato articolato come segue:

1. Selezione delle etichette: come detto precedentemente, ogni record è rappresentato da un set di ambiti applicativi che lo classificano. Per prima cosa, per ogni azienda è stato estratto il primo elemento del set mentre gli altri non sono stati considerati. In questo modo si ha un problema di classificazione multi-classe.
2. Creazione del modello: il classificatore scelto e la fase di training e test del modello sono stati i medesimi utilizzati per l'implementazione dell'algoritmo di classificazione effettuata con il dataset scaricato da Kaggle, ampiamente discussa nelle sezioni 4.2.5.3.2 e 4.2.5.3.3, che avevano portato, ricordiamo, ad un valore di accuratezza del modello pari al 70%.

3. Analisi dei risultati: nonostante le buone premesse l'algoritmo presenta una accuratezza nella predizione dei dati pari al 20%.

4.2.5.3.5 Dataset Torino Wireless – problema multi-label

Alla luce dei risultati poco soddisfacenti appena descritti nella sezione 4.2.5.3.6 si è considerata la possibilità di intraprendere un ulteriore approccio per cercare di risolvere il problema di classificazione con dei risultati un po' più accettabili. A tal proposito riepiloghiamo le fasi costituenti di tale approccio:

1. Gestione e analisi dei dati di input: come abbiamo già discusso nella sezione 4.2.5.3.5 analizzando il dataset è facile notare che ad ogni azienda sono associati molteplici ambiti applicativi e quindi più etichette di classi possibili assegnabili ad ogni singola azienda. Questa analisi ci sarà utile nella fase successiva che riguarda la scelta del classificatore più idoneo alla soluzione del problema. A questi dati andrà aggiunta la colonna relativa alle keywords estratte per ogni azienda che costituiranno, anche in questo caso, l'input del classificatore che quindi dovrà, date le parole chiave, restituire un set di etichette per ogni azienda.
   In questa fase si è utilizzato lo stesso dataset finale usato nella sezione precedente 4.2.5.3.5 che risulta composto da 240 record e 18 possibili etichette, le combinazioni delle quali andranno a classificare ciascun singolo record.
2. Analisi del problema e scelta del modello: A fronte dei dati di input in possesso si riconosce di essere davanti ad un problema di classificazione multi-etichetta che prevede che il set di training sia composto da un insieme di record ad ognuno dei quali possono essere associate una o più etichette con nessuna correlazione precisa tra di loro. Si spiega di seguito la differenza tra un problema multi-class e un problema multi-label. Un tipico esempio di problema multi-classe è quello relativo alla sentiment analysis il cui schema è visibile in Figura **46**.

Record	Class
1	Positivo
2	Negativo
3	Neutro

FIGURA 46-PROBLEMA DI CLASSIFICAZIONE SENTIMENT ANALYSIS

In questo caso un record può essere classificato secondo tre diverse classi che sono positivo, negativo o neutro a partire da dati di input sottoforma di testo, come lo sono le parole chiave estratte nel caso in esame. La classe associata ad ogni record è unica, un record non può essere classificato come la combinazione di due o tre di queste classi dato che un testo non può essere classificato neutro e positivo contemporaneamente ecc.…

Nel problema multi-label invece, come si vede in Figura **47**, dato un set di classi, ad esempio 3, un record può essere caratterizzato con una qualsiasi combinazione di queste tre classi quindi classificato con un set di esse.

Record	Class
1	Class 1, Class 2
2	Class1
3	Class2, Class3

Record	Class1	Class2	Class3
1	1	1	0
2	1	0	0
3	0	1	1

FIGURA 47-PROBLEMA DI CLASSIFICAZIONE MULTI-LABEL

Un modo per risolvere il problema di classificazione multi-label è quello che prevede la sua trasformazione in un problema multi-classe. Ci sono diversi metodi per fare questa operazione:

a) Binary relevance: si tratta ogni singola classe come fosse un problema di classificazione binario quindi si fanno tanti problemi di classificazione binari pari al numero di classi. Lo svantaggio di questo approccio è quello per cui nel caso in cui ci fossero delle correlazioni tra le classi, esse andrebbero perse.
b) Classifier Chains: in questo metodo si usa il risultato del training delle classificazioni precedenti a quella in atto unito alle caratteristiche dell'input per effettuare il nuovo training per le classificazioni successive. Questo metodo è molto efficiente nel

caso in cui ci sia una correlazione tra le diverse classi con cui si deve etichettare il record.

c) Label powerset: questo metodo di trasformazione prevede di creare una classe unica da associare ad ogni record che coincida con la combinazione delle corrispettive tre.

Nel nostro caso abbiamo deciso di scegliere il primo metodo dato che le possibili etichette si sono rivelate 18 e che non sussiste una correlazione logica tra di esse. Allora avremmo 18 problemi di classificazione.

3. Implementazione del codice: in questa fase si implementa il codice relativo al metodo scelto. A tal proposito ci serviamo delle seguenti librerie:
   a) skmultilearn che serve per trasformare il problema multi-label il un problema multi-classe di tipo binario tramite il metodo binary relevance descritto precedentemente. Il metodo è BinaryRelevance() che prende come parametro di input un classificatore usato per la classificazione non multi-label.
   b) sklearn che viene utlizzata per ricavare l'algoritmo per un normale problema di classificazione. Nel nostro caso utilizziamo MultinomianNB() che è un classificatore di Naive-Bayes di cui abbiamo spiegato il funzionamento nella sezione 4.2.6.2. Tale classificatore è dato come parametro di input al metodo descritto al punto a).
4. Valutazione dei risultati: dopo l'applicazione dell'algoritmo i risultati raggiunti non sono quelli che ci si attendeva di ottenere. In particolare, dalle analisi del set di test emerge che a 57 record su 60 non è stata assegnata nessuna etichetta, a 2 record su 60 è stata assegnata l'etichetta corretta ed a 1 su 60 è stata assegnata una etichetta in modo erroneo.

4.2.5.3.6 Classifications – modelli a confronto

Nella presente sezione si riporta in Tabella **3** uno schema riassuntivo di tutti i problemi di classificazione descritti dettagliatamente nelle sezioni 4.2.5.3.2, 4.2.5.3.3, 4.2.5.3.4, 4.2.5.3.5 con i rispettivi valori delle metriche ottenute.

TABELLA 3-SCHEMA CLASSIFICAZIONE

PROBLEMA DI CLASSIFICAZIONE	DATASET TRAIN	DATASET TEST/PREDICTION	RISULTATI
Multi-class	Kaggle - "Company Name Classification"	Kaggle - "Company Name Classification"	Accuratezza ≃ 70%
	Kaggle - "Company Name Classification"	TorinoWireless	Accuratezza ≃ 70%
	TorinoWireless	TorinoWireless	Accuratezza ≃ 20%
Multi-label	TorinoWireless	TorinoWireless	03 / 60 classificazione corretta 01 /60 classificazione errata 56 / 60 no classificazione

Dalla tabella si vede immediatamente che, come detto precedentemente, i risultati peggiori si ottengono nel momento in cui si utilizza sia per il training che per il test il dataset fornito come input dall'ente Fondazione Torino Wireless, soprattutto per il caso dalla risoluzione del problema di classificazione multi-label.

5. Conclusioni e prospettive future

L'ultimo capitolo si comporrà, prima, dalla descrizione della visione di insieme di tutto il progetto insieme all'analisi delle sue criticità e dopo nella descrizione di alcune prospettive future che prendano come input i risultati ottenuti dal software.

5.1 Conclusioni

Il primo obbiettivo del presente progetto di tesi è quello di riuscire ad ottenere un database che raccolga informazioni, relative ad euristiche precise, estratte dai siti web di aziende, in modo da agevolare la delineazione dei loro profili. Alla fine del lavoro, il software ha reso possibile, in primo luogo, la creazione di un preliminare database con dati di interesse riguardanti 300 aziende, utili per valutazioni di competenza di Fondazione Torino Wireless; in secondo luogo, la possibilità per l'ente collaborante di controllare uno strumento, ovvero il software sviluppato, che potrà essere replicabile in contesti simili e su larga scala.

La criticità maggiore che è stata riscontrata per il raggiungimento dell'obbiettivo appena descritto è quella relativa alla separazione dei dati sensibili da quelli ridondanti. Tale criticità si è manifestata nella fase di scraping dei dati dai siti web aziendali descritta nella sezione 4.2.2 nel momento in cui non è risultato possibile, nella maggior parte dei casi, individuare un pattern di programmazione web per la scrittura delle pagine HTML ripetuto che faciliti la selezione delle sezioni comuni dei siti web. Sicuramente, il riscontro di tale criticità potrebbe essere un buono spunto per sensibilizzare l'intero ambito che si occupa della parte di programmazione web che potrebbe diventare un po' più rigida nella sua struttura prevedendo pattern di implementazione predefiniti come quelli relativi agli esempi descritti nella sezione 4.2.2.2.

Per quanto riguarda la fase di classificazione abbiamo ottenuto dei risultati meno soddisfacenti rispetto a quelli relativi alla fase relativa allo scraping e alla estrazione delle parole chiave. Dopo una prima ondata di euforia scaturita dal fatto per cui si erano riscontrati valori di accuratezza consistenti utilizzando un algoritmo che classificava le aziende secondo 10 macrosettori prendendo in input le parole chiave estratte da una corrispettiva descrizione aziendale si è pensato che si potessero ottenere gli stessi risultati trattando il dataset fornito dall'ente

collaborante nello stesso modo e riuscire così a fornire, oltre alle informazioni aziendali, anche un solido strumento per classificare le aziende. Purtroppo, i risultati ottenuti sul dataset di Fondazione Torino Wireless non sono stati gli stessi del caso precedente. Questo, tuttavia, ci ha portato a chiederci quali fossero le problematiche legate alle scarse performance dell'algoritmo che alla fine si sono individuate in:

- dimensioni ridotte del dataset di input
- ambiti applicativi troppo specifici e spesso accumunabili sotto una stessa categoria
- etichette usate per la classificazione scaturite dopo un'intervista con le aziende

La somma di questi punti non permetterebbe al modello di allenarsi adeguatamente sul set di training e di conseguenza risulterebbe impossibile applicare il modello per predizioni future.

5.2 Prospettive future e Visual Analytics

Oltre alla possibilità di affinare l'analisi dei dati raccolti nel presente lavoro affinché essi risultino ancora più utili e precisi, tra le implementazioni future che potrebbero svilupparsi a partire proprio dai dati prodotti dal presente lavoro di tesi vi è la rappresentazione di questi tramite strumenti di Visual Analytics. La Visual Analytics è quello strumento che si occupa principalmente di unire rappresentazioni visive interattive volte a supportare le decisioni di business con processi analitici come quello di data mining effettuato nel presente lavoro. In particolare, potrebbe risultare molto interessante creare una rete in grado di identificare i vari network di aziende presenti nel sistema di Fondazione Torino Wireless. I parametri discrezionali per la costruzione di questa rete potrebbero essere molteplici come:

- I dati relativi a clienti e a partner di una azienda
- La dislocazione delle diverse sedi aziendali per verificare il territorio coperto da una impresa
- I particolari progetti seguiti da una o più aziende

Un'altra importante riflessione legata all' implementazione di progetti futuri è quella che coinvolge il problema di classificazione a partire dai dati forniti dall'azienda collaborante. I risultati ottenuti nella relativa parte del lavoro di tesi ci

forniscono infatti delle solide basi da cui partire nello studio di un modello di machine learning ad hoc per il problema descritto nel presente lavoro e per una sua successiva implementazione.

Bibliografia e sitografia

[1] K. Malkan, “Web Scraping With Python | 4 Things To Know.” https://www.techwhoop.com/web-scraping-with-python/.

[2] B. Zhao, “Web scraping,” *Encycl. big data*, pp. 1–3, 2017.

[3] V. Krotov, L. Johnson, and L. Silva, “Tutorial: Legality and Ethics of Web Scraping,” 2020.

[4] M. Kumar, R. Bhatia, and D. Rattan, “A survey of Web crawlers for information retrieval,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 7, no. 6, p. e1218, 2017.

[5] S. vanden Broucke and B. Baesens, *Practical Web Scraping for Data Science*. 2018.

[6] “selenium_documentation.” https://www.selenium.dev/.

[7] “Beautiful Soup Documentation.” https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

[8] Srinivas Chakravarthy; Chandrasehkar Nagaraj, “Tokenizzazione per l’elaborazione del linguaggio naturale.” https://ichi.pro/it/tokenizzazione-per-l-elaborazione-del-linguaggio-naturale-177543891237588 (accessed Jul. 03, 2021).

[9] “Devopedia. 2019. ‘Stemming.’ Version 3.” https://devopedia.org/stemming (accessed Jul. 03, 2021).

[10] S. Yang, “Keyword Extraction: from TF-IDF to BERT.” https://towardsdatascience.com/keyword-extraction-python-tf-idf-textrank-topicrank-yake-bert-7405d51cd839.

[11] D. Ellis, “Using TF IDF to form descriptive chapter summaries via keyword extraction.” https://towardsdatascience.com/using-tf-idf-to-form-descriptive-chapter-summaries-via-keyword-extraction-4e6fd857d190.

[12] M. Grootendorst, “Keyword Extraction with BERT.” .

[13] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt, “YAKE! Keyword extraction from single documents using multiple local features,” *Inf. Sci. (Ny).*, vol. 509, pp. 257–289, 2020.

[14] “spacy_model.” https://spacy.io/models/en.

[15] Y. Zhang, *New advances in machine learning*. BoD--Books on Demand, 2010.

[16] “Le tipologie di machine learning.” https://course.elementsofai.com/it/4/1.

[17] “MACHINE LEARNING: COME SCEGLIERE IL MODELLO MIGLIORE.” https://humanativaspa.it/machine-learning-come-scegliere-il-modello-migliore/.

[18] L. Govoni, “Algoritmo Support Vector Machine.” https://www.lorenzogovoni.com/support-vector-machine/.

[19] J. Brownlee, “How Much Training Data is Required for Machine Learning?,” 2017. https://machinelearningmastery.com/much-training-data-required-machine-learning/.