ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE
LAUREA IN INFORMATICA

Generazione automatica di una knowledge base con applicazione ai Sustainable Development Goals

Relatore:
Chiar.mo Prof.
Maurizio Gabbrielli

Presentata da:
Stefano Colamonaco

Sessione I
Anno accademico 2021/2022

«*The question of whether a computer can think is no more interesting than the question of whether a submarine can swim.*»

-Edsger W. Dijkstra

Abstract

Questa tesi di laurea compie uno studio sull' utilizzo di tecniche di web crawling, web scraping e Natural Language Processing per costruire automaticamente un dataset di documenti e una knowledge base di coppie verbo-oggetto utilizzabile per la classificazione di testi.
Dopo una breve introduzione sulle tecniche utilizzate verrà presentato il metodo di generazione, prima in forma teorica e generalizzabile a qualunque classificazione basata su un insieme di argomenti, e poi in modo specifico attraverso un caso di studio: il software SDG Detector. In particolare quest ultimo riguarda l'applicazione pratica del metodo esposto per costruire una raccolta di informazioni utili alla classificazione di documenti in base alla presenza di uno o più Sustainable Development Goals.
La parte relativa alla classificazione è curata dal co-autore di questa applicazione, la presente invece si concentra su un'analisi di correttezza e performance basata sull'espansione del dataset e della derivante base di conoscenza.

Indice

1 Introduzione

Con l'aumentare della quantità di dati testuali digitalizzati presenti nel Web e non solo, aumenta sempre di più anche il bisogno di creare sistemi di Information Retrieval, capaci di rappresentare, memorizzare e soprattutto recuperare le informazioni non strutturate, sulla base di una determinata richiesta. Dai primi sistemi più elementari, basati su indicizzazione di documenti, si sono sviluppate recentemente tecnologie considerate 'intelligenti'. Queste fanno uso di tecniche di Machine Learning e Deep Learning, le quali hanno preso piede negli ultimi decenni grazie alla grande quantità di dati di cui possiamo disporre attraverso la rete o dataset già noti e alla potenza di calcolo dei nuovi elaboratori [6].

Un dataset è un insieme di dati strutturati in forma relazionale creati per essere letti ed elaborati da appositi algoritmi. La loro costruzione richiede solitamente molto lavoro sia dal punto di vista computazionale, per via dell'enorme mole di dati da gestire, che manuale in quanto non esistono metodi standard di costruzione automatica che facciano a meno della supervisione umana [4]. Ancora più complicata è la costruzione di una knowledge base (KB), la differenza sostanziale è che mentre i primi sono considerabili insiemi di dati utili ad ottenere risposte, una knowledge base costituisce un ambiente volto a facilitare la raccolta, l'organizzazione e la distribuzione della conoscenza fornendo alle applicazioni direttamente le risposte di cui hanno bisogno [11].

La presente tesi si propone prima di tutto di esporre un metodo sperimentale per la costruzione automatizzata di un dataset di coppie verbo-oggetto utili ad effettuare classificazioni, e in seguito di mostrare un'applicazione di questo metodo con un caso di studio. L'idea per la generazione della collezione è quella di sfruttare un principio di località piuttosto astratto, ma potenzalmente funzionante: *"Se mi aspetto che un testo parli di uno specifico argomento perchè ho un riscontro con ciò che è attualmente contenuto nel mio dataset, è molto probabile che le informazioni extra presenti nel testo in*

questione possano essermi di aiuto a riconoscere nuovamente questo argomento in futuro". Al termine di questa costruzione, se il dataset avrà raggiunto una dimensione e una accuratezza adeguate e possederà un alto livello di affinità con l'argomento di studio, avremo a disposizione una base di conoscenza in grado di aiutarci a classificare un testo in esame. Lo scopo dell'applicazione SDG Detector, sviluppata con il tesista Filip Radovic, è infatti quello di raccogliere dati testuali presenti su specifiche pagine web o documenti in rete ed analizzarli per verificare se vi sono associazioni con uno o più Sustainable Development Goals (SDG). I SDG sono una serie di 17 obiettivi interconnessi, definiti dall'Organizzazione delle Nazioni Unite come strategia "per ottenere un futuro migliore e più sostenibile per tutti" [20]. Riuscire ad individuare riferimenti a questi obiettivi potrebbe risultare particolarmente utile nell'analisi di documentazioni aziendali, con lo scopo di verificarne la dedizione alla sostenibilità. Il documento è strutturato nel modo seguente:

- Il capitolo corrente si occupa semplicemente di fornire un quadro generale alla tesi;
- Il secondo e il terzo capitolo forniscono i fondamenti teorici necessari a comprendere la natura e l'implementazione del metodo. In particolare vengono fornite nozioni di base in ambito di Natural Language Processing, Text mining, Text classification, Web crawling e Web scraping e vengono mostrati alcuni tool utilizzati all'interno dell'applicazione realizzata, quali NLTK, BeautifulSoup, Stanford NLP e Stanza;
- Nel quarto capitolo vengono esposte le principali tematiche riguardanti la costruzione di un dataset e le operazioni necessarie. Inoltre viene esposto in maniera generale il metodo di generazione incrementale alla base di questo documento;
- Nel quinto capitolo vengono introdotti brevemente i Sustainable Development Goals con particolare attenzione sul loro ruolo e su cosa afferma ognuno di essi. Dopo di che viene mostrata una implementazione del metodo esposto nel capitolo 4 all'interno di un caso di studio: l'applicazione SDG Detector, completa di una valutazione delle performance di generazione;

- Il sesto e ultimo capitolo conclude l'elaborato facendo un'analisi degli obiettivi raggiunti, delle limitazioni attualmente presenti e degli studi futuri.

2 Natural Language Processing

2.1 Cos'è il Natural Language Processing

Il Natural Language Processing (NLP) si riferisce al ramo dell'informatica - e più specificamente, il ramo dell'intelligenza artificiale - che si occupa di fornire ai computer la capacità di comprendere testo e parole pronunciate più o meno allo stesso modo degli esseri umani [2].
Il NLP combina la linguistica computazionale e la modellazione del linguaggio umano basata su regole con modelli statistici di apprendimento automatico e di deep learning. Insieme, queste tecnologie consentono ai computer di elaborare il linguaggio umano sotto forma di testo o dati vocali e di "capire" il suo pieno significato, comprensivo dell'intento e del sentimento di chi parla o scrive.

2.2 Studio del linguaggio naturale

Il linguaggio umano è pieno di ambiguità che rendono incredibilmente difficile scrivere software che determini con precisione il significato previsto del testo o dei dati vocali. Omonimi, omofoni, sarcasmo, modi di dire, metafore, grammatica, eccezioni d'uso e variazioni nella struttura delle frasi: queste solo alcune delle irregolarità del linguaggio umano che necessitano di anni per essere imparate, mentre i programmatori devono insegnare alle applicazioni natural language-driven a riconoscerle e capirle accuratamente fin dall'inizio. Diverse attività NLP scompongono il testo umano e i dati vocali in modi che aiutano il computer a dare un senso a ciò che sta analizzando [3]. Alcune di queste attività includono quanto segue:

- Il **riconoscimento vocale**, chiamato anche sintesi vocale, è il compito di convertire in modo affidabile i dati vocali in dati di testo. Il riconoscimento vocale è necessario per qualsiasi applicazione che segua i comandi vocali o risponda

a domande pronunciate. Ciò che rende particolarmente difficile il riconoscimento vocale è il modo in cui le persone parlano: velocemente, confondendo le parole insieme, con enfasi e intonazione diverse, con accenti diversi e spesso utilizzando una grammatica errata.

- Il **part of speech tagging**, chiamato anche tagging grammaticale, è il processo per determinare la parte del discorso di una particolare parola o pezzo di testo in base al suo uso e contesto. Part of speech identifica "sperare" come un verbo in "Possiamo solo sperare che non sia così" e come sostantivo in "Lo sperare dà conforto".

- La **disambiguazione** del senso delle parole è la selezione adatta del concetto espresso da una parola con più significati, attraverso un processo di analisi semantica che determina la parola che ha più senso nel contesto dato. Ad esempio, la disambiguazione del senso delle parole aiuta a distinguere il significato di "rombo" in "Ho acquistato un rombo fresco al mercato" (pesce) rispetto a "Il rombo ha quattro lati della stessa lunghezza" (forma geometrica).

- Il **Named Entity Recognition** (NEM) identifica parole o frasi come entità utili. NEM identifica "Kentucky" come luogo o "Fred" come nome di un uomo.

- La **risoluzione di co-riferimento** è il compito di individuare se e quando due parole si riferiscono alla stessa entità. L'esempio più comune è determinare la persona o l'oggetto a cui si riferisce un certo pronome (ad esempio, 'lei' = 'Martina'), ma può anche implicare l'identificazione di una metafora o di un modo di dire nel testo.

- La **sentiment analysis** tenta di estrarre qualità soggettive - atteggiamenti, emozioni, sarcasmo, confusione, sospetto - dal testo.

- La **Natural Language Generation** è talvolta descritta come l'opposto del riconoscimento vocale o della sintesi vocale; è il compito di inserire informazioni strutturate nel linguaggio umano.

2.3 Text mining

Il text mining è una branca del Natural Language Processing che si occupa in particolare dell'estrazione di informazioni da input di tipo testuale. Esso è definito come il processo di esplorazione e analisi di grandi quantità di dati testuali non strutturati grazie all'aiuto di software specifici che riescano ad identificare concetti, patterns, topics, parole chiave e altre peculiarità dei dati. L'obiettivo della ricerca riguardante il text mining è quindi studiare nuovi metodi e algoritmi per estrarre automaticamente conoscenza dal testo, al fine ad esempio di classificare o raggruppare documenti in base ai contenuti. Per natura il text mining è simile al data mining, anche se si concentra su testi che in genere non sono strutturati [18].

2.4 Text classification

La classificazione del testo è una tecnica di machine learning che assegna tag o etichette ad un testo libero in analisi a seconda del suo contenuto [19]. I classificatori di testo possono essere utilizzati per organizzare, strutturare e classificare praticamente qualsiasi tipo di testo, da documenti, studi medici e file e in tutto il Web.

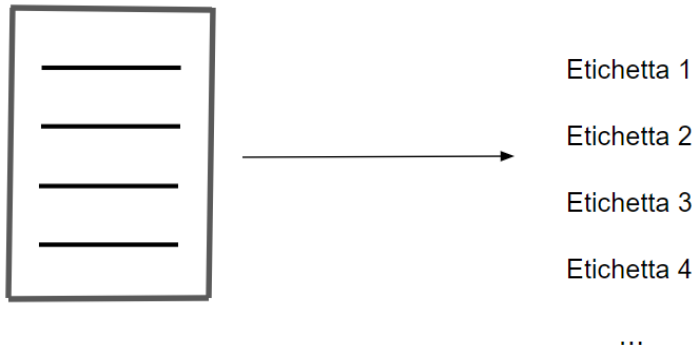

Figura 2.1 Text classification

Si stima che circa l'80% di tutte le informazioni presenti in rete non sia strutturato. Tra queste, i testi rappresentano uno dei tipi di dato più comune, ma anche i più difficili da analizzare a causa della loro natura disordinata. L'utilizzo della text classification viene incontro a questi problemi e porta notevoli vantaggi rispetto ad un'analisi manuale:

- **Scalabilità:** l'analisi e l'organizzazione manuali sono lente e molto meno accurate. L'apprendimento automatico può analizzare automaticamente milioni di sondaggi, commenti, e-mail, ecc., a una frazione del costo, spesso in pochi minuti. Gli strumenti di classificazione del testo sono scalabili per qualsiasi esigenza aziendale, che sia essa grande o piccola.

- **Analisi real-time:** ci sono situazioni critiche che le aziende devono identificare il prima possibile e agire immediatamente (ad esempio, crisi di pubbliche relazioni sui social media). La classificazione del testo attraverso il machine learning può operare costantemente su dati sempre aggiornati così da permettere un'identificazione in tempo reale delle informazioni e una risposta istantanea.

- **Consistenza dei criteri:** gli umani commettono errori quando classificano i dati di testo a causa di distrazioni, stanchezza o fattori esterni e la soggettività umana crea criteri incoerenti. I classificatori automatici invece applicano la stessa lente e criteri a tutti i dati e risultati. Una volta che un modello di classificazione del testo è stato adeguatamente addestrato, funziona con una precisione difficile da superare.

2.4.1 Approcci possibili

Esistono molti approcci alla classificazione automatica del testo, ma rientrano tutti in tre tipi di approcci: [19]

- **Sistemi rule-based:** Gli approcci basati su regole effettuano la classificazione utilizzando una serie di regole linguistiche. Queste regole istruiscono il sistema a utilizzare elementi semanticamente rilevanti di un testo per identificare le etichette possibili in base al suo contenuto.

  Ad esempio supponiamo di voler classificare gli articoli di alcune notizie in due gruppi: Sport e Politica. Innanzitutto, dovremo definire due elenchi di parole che caratterizzano ciascun gruppo (es. parole legate a sport come football, basket, Cristiano Ronaldo, LeBron James, ecc., e parole legate alla politica, come Donald Trump, Hillary Clinton, Putin , eccetera.). Successivamente, quando vogliamo classificare un nuovo testo in arrivo, dovremo contare il numero

di parole relative allo sport che appaiono nel testo e fare lo stesso per le parole relative alla politica. Se il numero di apparizioni di parole relative ad una categoria è superiore rispetto al conteggio fatto per l'altra, il testo viene classificato come Sport o viceversa.

I sistemi basati su regole sono comprensibili dall'uomo e possono essere migliorati nel tempo. Ma questo approccio presenta alcuni svantaggi. Per cominciare, questi sistemi richiedono una profonda conoscenza del dominio. Inoltre, richiedono molto tempo, poiché la generazione di regole per un sistema complesso può essere piuttosto impegnativa e di solito richiede molte analisi e test. In aggiunta, i sistemi basati su regole sono difficili da mantenere e non si adattano bene, poiché l'aggiunta di nuove regole può influire sui risultati delle regole preesistenti.

- **Sistemi machine learning based:** Invece di fare affidamento su regole create manualmente, questo tipo di classificazione impara ad operare basandosi su osservazioni passate. Utilizzando esempi pre-etichettati come dati di addestramento, gli algoritmi di machine learning possono apprendere le diverse associazioni tra parti di testo e la relazione tra un input particolare e un output particolare. Il primo passo verso la formazione di questo tipo di classificatore è l'estrazione di funzionalità (feature extraction): viene utilizzato un metodo per trasformare ogni testo in una rappresentazione numerica sotto forma di vettore. Uno degli approcci più utilizzati è il "bag of words", in cui un vettore rappresenta la frequenza di una parola in un dizionario predefinito.

  Ad esempio, supponiamo di aver definito il nostro dizionario in modo che abbia le seguenti parole: *Lorem*, *ipsum*, *dolor*, *sit*, *amet*. se volessimo vettorializzare il frammento "Lorem sit", avremmo la seguente rappresentazione vettoriale: (1,0,0,1,0).

  Quindi, l'algoritmo viene alimentato con dati di allenamento che consistono in coppie di tag (ad es. sport, politica) e risultati della feature extraction (con vettori per ogni esempio di testo) per produrre un modello di classificazione:

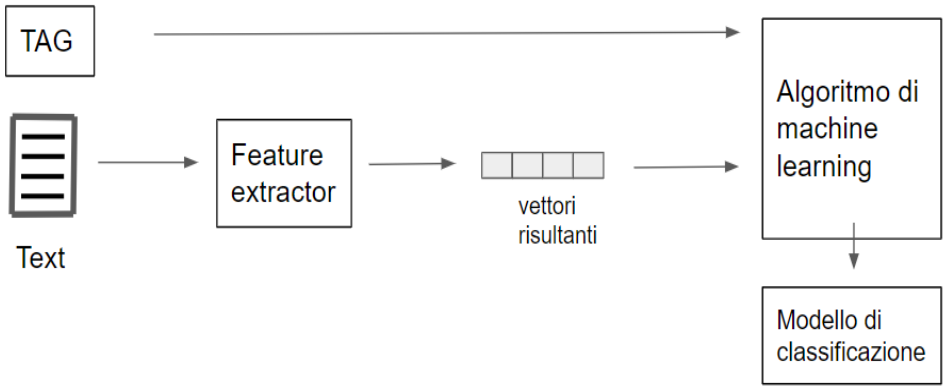

Figura 2.2 Costruzione del modello di classificazione

Una volta addestrato con un numero sufficiente di campioni di addestramento, il modello di apprendimento automatico può iniziare a fare previsioni accurate. L' estrattore di funzionalità viene utilizzato per trasformare testi mai visti in set di funzionalità, che possono essere inseriti nel modello di classificazione per ottenere previsioni sui tag:

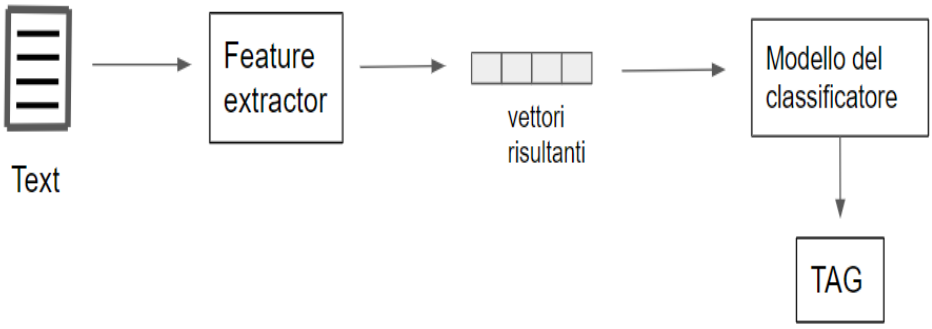

Figura 2.3 Utilizzo del modello del classificatore

Questo tipo di classificazione è generalmente molto più accurata dei sistemi basati su regole, in particolare su attività di classificazione complesse. Inoltre, sia il mantenimento che l'aggiunta di nuovi esempi sono molto più semplici.

- **Implementazioni ibride:** i sistemi ibridi combinano un classificatore di base addestrato attraverso machine learning con un sistema basato su regole, utilizzato per migliorare ulteriormente i risultati. Questi sistemi ibridi possono

essere facilmente ottimizzati aggiungendo regole specifiche per quei tag generatori di conflitto che non sono stati modellati correttamente dal classificatore di base.

Naive Bayes

La famiglia di algoritmi statistici Naive Bayes comprende alcuni degli algoritmi più utilizzati nella classificazione e nell'analisi del testo [16] e sono stati una base fondamentale per lo sviluppo del metodo di generazione e dell'applicazione trattati in questo documento.
Per comprendere come questi classificatori funzionano è prima necessario introdurre alcune basi teoriche di probabilità.
Il teorema di Bayes, enunciato da Thomas Bayes (1702-1761), discende da due risultati fondamentali della teoria della probabilità: il teorema della probabilità composta e il teorema della probabilità assoluta. Esso ci fornisce una formula utile al calcolo di una probabilità condizionata, ovvero la probabilità che una cosa accada dato qualcosa che è già successo.
Teorema di Bayes: Siano A e B due eventi tali che P(A) > 0 e P(B) > 0, allora vale la formula

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

In particolare con il Naive Bayes calcoliamo la probabilità di selezione di ciascun tag per un determinato testo e viene dato in output il tag con la probabilità più alta. Per fare ciò i passi da seguire sono i seguenti:

- Si definisce un vettore di attributi

  $$A = (A_1, A_2, ...A_n)$$

  dove qualsiasi vettore che rappresenti un testo dovrà contenere informazioni sulle probabilità di apparizione di specifiche parole all'interno dei testi di una determinata categoria, in modo che l'algoritmo possa calcolare la probabilità che il testo in esame appartenga alla stessa;

- Si definisce una variabile di classe C;

- Possiamo a questo punto trattare le due variabili come variabili casuali e catturare le loro relazioni probabilistiche utilizzando *P(C | A)*;
- Durante la fase di allenamento si imparano i legami probabilistici *P(C | A)* per ogni combinazione di valori assunti da A e C;
- Conoscendo queste probabilità un test record x può essere classificato trovando la label di classe c che massimizza la probabilità *P(c | x)*.

Quasi tutte le librerie di NLP forniscono un'interfaccia per utilizzare il metodo di classificazione bayesiano, visto il suo ottimo rapporto tra efficienza e affidabilità dei risultati.

2.4.2 Metriche e valutazione dei risultati

La convalida incrociata è un metodo comune per valutare le prestazioni di un classificatore di testo. Funziona suddividendo il set di dati di addestramento in gruppi di esempio casuali di uguale lunghezza (ad esempio, 4 set con il 25% dei dati ciascuno). Per ogni set, viene addestrato un classificatore di testo utilizzando i campioni rimanenti (in questo caso di esempio, il 75% del set). Successivamente, i classificatori fanno previsioni sui rispettivi insiemi e i risultati vengono confrontati con i tag rilevati attraverso analisi umane. Questo determinerà quanto una previsione era effettivamente corretta e quando invece sono stati commessi degli errori (falsi positivi, falsi negativi).
Con questi risultati si possono costruire metriche delle prestazioni, utili per una rapida valutazione del funzionamento di un classificatore:

- **Accuratezza:** la percentuale di testi classificati con il tag corretto.
- **Precisione:** la percentuale di esiti positivi ottenuta dal classificatore rispetto al numero di quelli previsti per un determinato tag.
- **Richiamo (recall):** la percentuale di testi classificati con un dato tag sul numero effettivo che avrebbe dovuto prevedere per esso.
- **Punteggio F1:** media armonica tra precisione e recall.

2.5 Strumenti e risorse

Il linguaggio di programmazione Python fornisce un'ampia gamma di strumenti e librerie per approcciare attività NLP specifiche. Esistono ovviamente moltissime librerie famose che offrono strumenti per la realizzazione di applicazioni che sfruttano NLP (ad esempio Scikit-learn, SpaCy, Keras, TensorFlow ...), ma sono state citate nello specifico solo quelle effettivamente utilizzate all'interno del progetto trattato in questa tesi.

2.5.1 La suite NLTK

Molti degli strumenti necessari si trovano nel Natural Language Toolkit, o NLTK, una raccolta open source di librerie, programmi e risorse educative per la creazione di programmi che sfruttano NLP [9]. NLTK include librerie per molte delle attività NLP elencate precedentemente, oltre a quelle per attività secondarie, come analisi delle frasi, segmentazione delle parole, stemming e lemmatizzazione (metodi per tagliare le parole fino alle loro radici) e tokenizzazione (per spezzare frasi e paragrafi in token che aiutano l'elaboratore a comprendere meglio il testo). Include anche librerie per l'implementazione di funzionalità come il ragionamento semantico e la capacità di raggiungere conclusioni logiche basate su fatti estratti dal testo.

2.5.2 Il framework CoreNLP

Stanford CoreNLP [10] è un framework all-inclusive per l'elaborazione del linguaggio naturale in Java. Esso consente agli utenti di derivare annotazioni linguistiche per il testo, inclusi token, frasi, parti del discorso, entità denominate, valori numerici e temporali, analisi delle dipendenze, coreferenze, sentimento, attribuzioni di citazioni e relazioni. CoreNLP attualmente supporta 8 lingue: arabo, cinese, inglese, francese, tedesco, ungherese, italiano e spagnolo.

Pipeline

Il fulcro di CoreNLP è la pipeline. Le pipeline raccolgono testo grezzo, eseguono una serie di annotatori NLP sul testo e producono una serie finale di annotazioni.

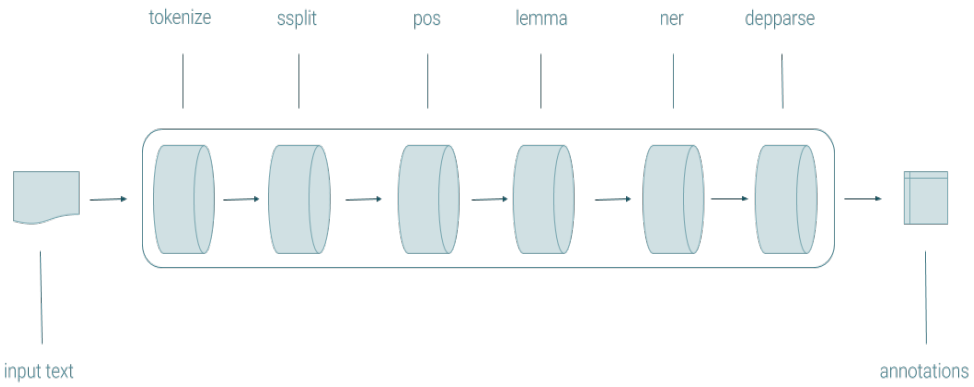

Figura 2.4 Pipeline di CoreNLP

Le pipeline producono CoreDocuments, oggetti che contengono tutte le informazioni di annotazione, accessibili con una semplice API e serializzabili su un Google Protocol Buffer.

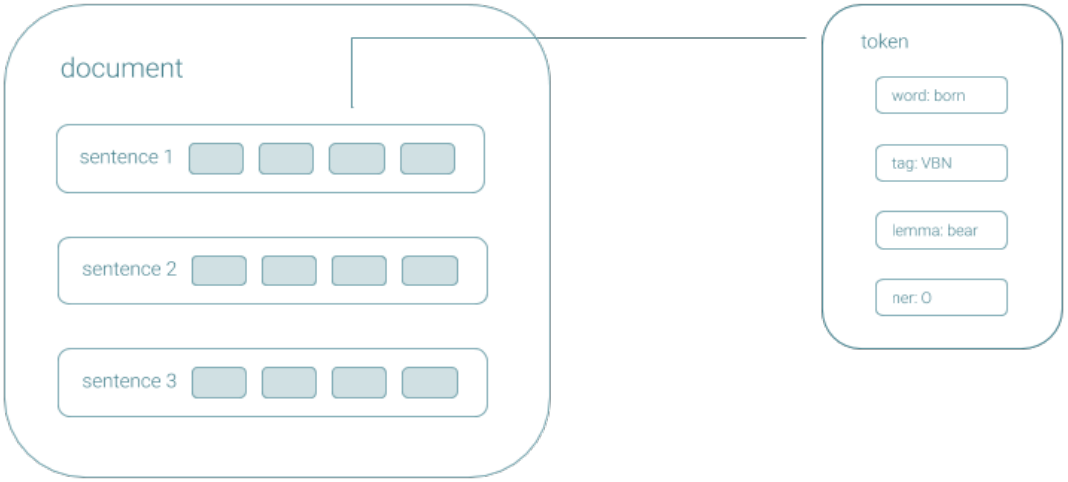

Figura 2.5 CoreDocument

CoreNLP genera una varietà di annotazioni linguistiche, tra cui:

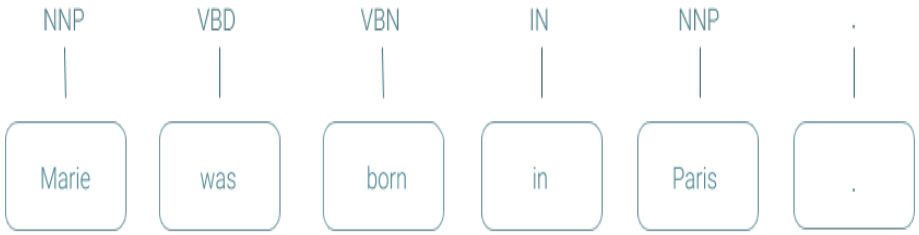

Figura 2.6 Part of speech

Figura 2.7 Named entities

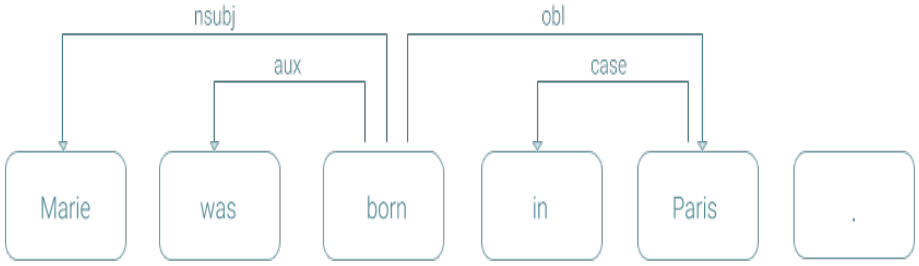

Figura 2.8 Dependency parse

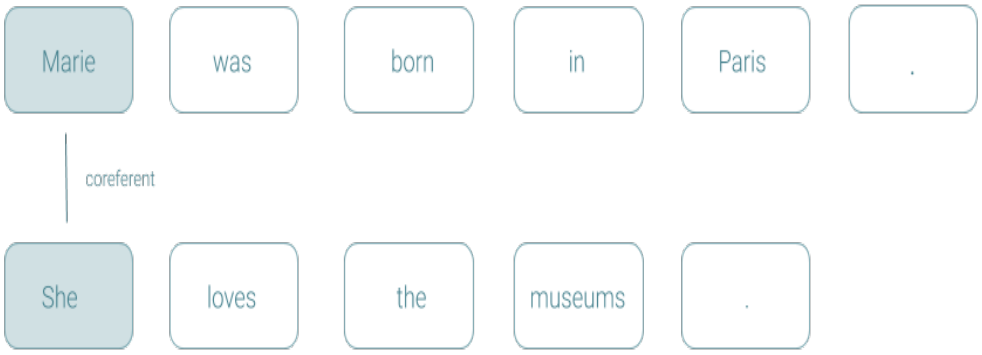

Figura 2.8 Coreference

Stanza e il POS Tagging

Stanza è un pacchetto sviluppato per l'analisi del linguaggio naturale in Python [14]. Contiene strumenti che possono essere utilizzati in una pipeline per convertire una stringa contenente testo in linguaggio umano, in elenchi di frasi e parole per generare forme di base di tali parole, le loro parti del discorso e le caratteristiche morfologiche, per fornire una struttura sintattica dell'analisi delle dipendenze e per riconoscere le entità denominate. Il toolkit è progettato per essere parallelo tra più di 70 lingue, utilizzando il formalismo delle dipendenze universali. Inoltre, Stanza include un'interfaccia Python per il pacchetto Java CoreNLP e da lì eredita funzionalità aggiuntive, come l'analisi del collegio elettorale, la risoluzione della coreferenza e la corrispondenza dei modelli linguistici. In fine Stanza può anche essere utilizzato come client per richiedere l'elaborazione remota delle informazioni, attraverso un'apposita interfaccia, ad un server su cui Stanford CoreNLP lavora come processo in background. Ecco una rappresentazione della pipeline in Stanza:

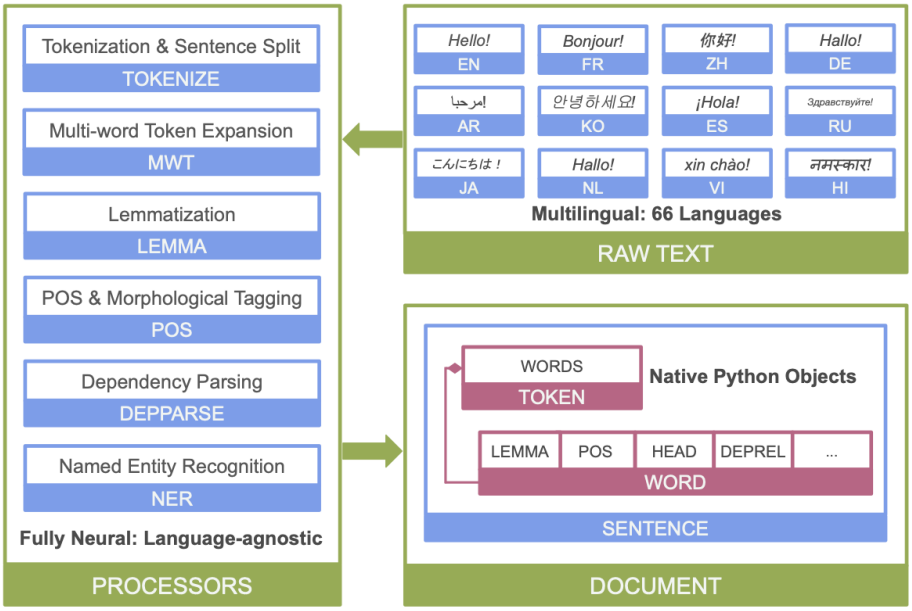

Figura 2.9 Pipeline NLP di Stanza

Tra tutti gli strumenti presenti all'interno della pipeline, quello più importante al fine di comprendere il metodo esposto in questo documento è *POS & Morphological Tagging* [21], ovvero l'assegnamento di informazioni morfologiche ai token che compongono una frase. Grazie a questi token e al loro studio nell'insieme è possibile risalire alla struttura logica della frase permettendo di identificare facilmente le relazioni che intercorrono tra soggetti, predicati verbali e oggetti. Stanza e in partcolar modo lo strumento appena descritto saranno fondamentali all'interno del caso di studio per permettere l'implementazione di un estrattore di coppie verbo-oggetto.

3 Ricerca di informazioni nel web

3.1 Web crawling e web scraping

La scansione del Web (**web crawling**) è il processo di indicizzazione dei dati nelle pagine Web utilizzando un programma o uno script automatizzato. Questi script o programmi sono conosciuti con più nomi, inclusi web crawler, spider, spider bot e spesso vengono abbreviati in crawler [13]. I web crawler copiano le pagine derivanti dall'elaborazione da parte di un motore di ricerca, che indicizza le pagine trovate in modo che gli utenti possano cercare in modo più efficiente. L'obiettivo di un crawler è scoprire di cosa trattano le pagine web e ciò consente agli utenti di recuperare qualsiasi informazione su una o più pagine quando è necessario. I crawler possono convalidare collegamenti ipertestuali e codice il HTML e possono essere utilizzati anche per il web scraping e la programmazione basata sui dati.

Il **web scraping** è il processo di raccolta di dati web strutturati in modo automatizzato [22]. Si chiama anche estrazione di dati web. Alcuni dei principali casi d'uso del web scraping includono il monitoraggio dei prezzi, l'intelligence sui prezzi , il monitoraggio delle notizie, la lead generation e le ricerche di mercato. In generale, l'estrazione di dati Web viene utilizzata da persone e aziende che desiderano utilizzare la grande quantità di dati Web disponibili pubblicamente per prendere decisioni più intelligenti. Copiando e incollando informazioni da un sito Web eseguiamo la stessa funzione di qualsiasi web scraper, solo su una scala manuale microscopica. A differenza del banale processo di estrazione manuale dei dati, il web scraping utilizza meccanismi automatici ed "intelligenti" per recuperare centinaia, milioni o addirittura miliardi di informazioni date dalla fonte apparentemente infinita di Internet. Ecco come generalmente vengono divise le operazioni svolte da uno script o un programma che si occupa di web scraping:

- Identificazione il sito web di destinazione;
- Raccolta degli URL delle pagine da cui desideriamo estrarre i dati;
- Si effettua una richiesta a questi URL per ottenere il contenuto della pagina in formato HTML;
- Si usano metodi di localizzazione per estrarre i dati nel documento HTML;
- Si salvano i dati in un file JSON o CSV o in un altro formato strutturato.

Notiamo come le prime due operazioni siano attinenti al web crawling, infatti solitamente i due meccanismi vengono utilizzati assieme.

3.1.1 Confronto

Spesso queste due operazioni vengono confuse, ma come abbiamo visto possiedono scopi e metodi operativi molto differenti. L'immagine seguente può aiutare a comprendere queste differenze:

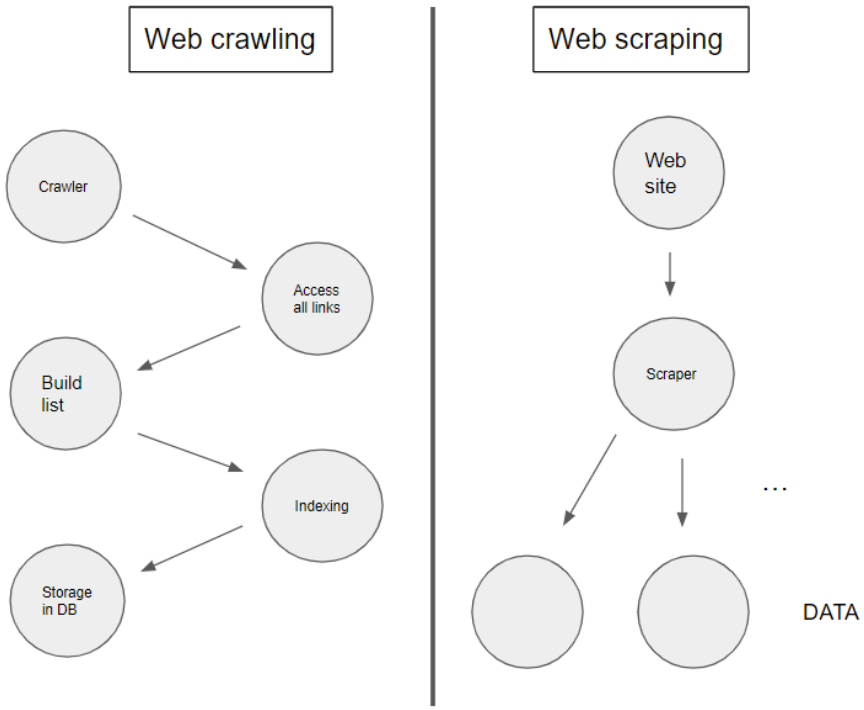

Figura 3.1 Rappresentazione di crawling e scraping

3.1.2 Strumenti

Il linguaggio Python oltre ad un'estrema semplicità nella gestione dei dati e delle strutture, è ricco di librerie facili da utilizzare ed efficaci che semplificano il web scraping. Un esempio utile anche ai fini di questa tesi è Beautiful Soup. Beautiful Soup è una libreria Python per estrarre dati da file HTML e XML [15]. In particolare permette di traformare il codice di una pagina HTML in input in un oggetto particolare da cui è possibile, attraverso semplici comandi, risalire a tutte le informazioni di contenuto e struttura.

3.1.3 Meccanismi di protezione

Esistono meccanismi per i siti pubblici che non desiderano essere sottoposti a scansione. Ad esempio, l'inclusione di un file *robots.txt* può richiedere ai bot di indicizzare solo specifiche parti di un sito Web o addirittura niente, inoltre esistono servizi molto famosi e utilizzati che servono a verificare che chi sta visitando il sito sia effettivamente un essere umano, ad esempio reCAPTCHA di Google. Provare a superare questi meccanismi è possibile, ma stanno diventando sempre più "intelligenti" e complicati.

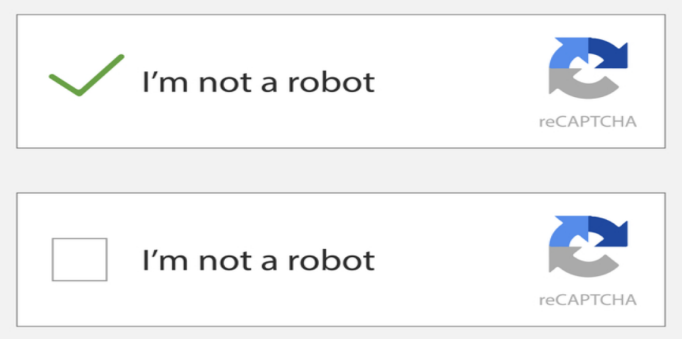

Figura 3.2 Servizio reCaptcha fornito da Google

4 Costruzione del dataset

4.1 Dataset e basi di conoscenza

Un dataset è un insieme di dati strutturati in forma relazionale creati per essere letti ed elaborati da appositi algoritmi [5]. Solitamente questo insieme di dati è molto vasto e può essere costruito in vari modi:

- **Creazione manuale:** è possibile creare manualmente tabelle o raccolte contenenti dati strutturati.
- **Creazione automatica o semi-automatica:** attraverso l'utilizzo di appositi software è possibile automatizzare le operazioni di ricerca, preprocessing e salvataggio dei dati.

Una knowledge-base (KB) è una libreria online di informazioni su un prodotto, servizio, reparto o argomento. Molte di esse sono strutturate attorno ad un'intelligenza artificiale in grado di interagire e rispondere all'input degli utenti. Altre sono semplicemente enciclopedie indicizzate o moduli leggibili da una macchina e costruiti per sfruttarne le informazioni. Le basi di conoscenza sono risorse preziose per lo sviluppo di applicazioni intelligenti, tra cui quelle di ricerca, risposta alle domande, data integration e recommendation systems. Per utilizzare KB di alta qualità bisogna ancora affidarsi quasi esclusivamente a dati strutturati o semi-strutturati la cui raccolta è stata curata dalla mano dell'uomo. Una tale dipendenza, la necessità di una supervisione umana, è un grosso ostacolo alla creazione di KB complete e sempre aggiornate. La popolazione di KB (KBP) è il compito di aumentare automaticamente una base di conoscenza con nuove informazioni verificate e ad oggi rappresenta un'argomento di studio molto popolare in ambito di Data Analysis [11].

4.2 Nozioni fondamentali di raccolta dei dati e preprocessing

La possibilità di portare avanti un task di Natural Language Processing (NLP), in ambito Machine Learning (ML), è resa possibile tanto dalla predisposizione di modelli e innovazioni tecnologiche, quanto dalla disponibilità di dati e dalla capacità di estrarre, manipolare e analizzare tali dati. Questi ultimi fattori si traducono in due fasi ben distinte dello sviluppo di un modello NLP: la creazione di una raccolta di dati e il loro preprocessamento [4] [5].
La scelta dei dati, così come la costruzione di una raccolta di dati, il dataset, sono dei passaggi fondamentali nel ciclo di costruzione di un modello, capaci di incidere significativamente sulle performance di quest'ultimo. Raccogliere dati, però, non basta. Bisogna anche saperli "preparare", in termini tecnici, preprocessare (in letteratura data preprocessing) per l'algoritmo, ovvero portarli in un formato che li renda predicibili e analizzabili dal compito NLP che stiamo per svolgere.
I passaggi standard che portano alla creazione di un dataset e al suo preprocessamento per task NLP sono quindi:

- **Data Collection:** reperimento dei dati;
- **Data Inspection:** osservazione dei dati per cercare eventuali errori;
- **Data Cleaning:** pulizia, formattazione e normalizzazione dei dati;
- **Data Annotation:** arricchimento attraverso metadati e informazioni che esplicitano le caratterstiche;
- **Data Reduction:** riduzione del volume;
- **Data Trasformation:** trasformazione dei dati in vettori numerici interpretabili dalle macchine.

Nel caso del NLP, per dati si intendono dati linguistici: parole, frasi, testi sia in forma scritta che parlata appartenenti al linguaggio umano. Durante il processo di costruzione di un dataset si devono tenere in considerazione quattro importanti fattori capaci di determinare la qualità del dataset stesso:

- Accuratezza: cercare di non includere dati che deviano da quelli aspettati.

- Completezza: assicurarsi che i dati raccolti siano sufficienti e rappresentativi del fenomeno che si vuole rappresentare.
- Consistenza: i dati, soprattutto se provengono da fonti diverse, devono essere uniformi.
- Interpretabilità: i dati devono essere comprensibili tanto dall'essere umano quanto dalla macchina che li processerà.

4.2.1 Data inspection

La fase di data inspection consiste nel rilevare quali sono quei gruppi di dati che sono sicuramente inutili agli scopi dell'applicazione. Ad esempio, consideriamo un'applicazione che ha lo scopo di raccogliere informazioni scritte su di una specifica pagina web. Molto probabilmente questa pagina conterrà molte informazioni extra rispetto a quelle significative per l'applicazione, o magari contiene addirittura dati completamente off-topic come pubblicità, informazioni di contatto o componenti stilistiche della pagina stessa. In questo caso una data inspection efficiente dovrebbe rimuovere questi tre esempi dai dati a disposizione, lasciando solo il contenuto effettivo della pagina. L'ispezione dei dati può essere accorpata al data cleaning.

4.2.2 Data cleaning

Il data cleaning è probabilmente l'operazione più importante se consideriamo il suo rapporto con il risultato finale. La regola "garbage in, garbage out" insegna che se i dati immessi in una macchina sono "sporchi", i risultati che dobbiamo attenderci lo saranno altrettanto. Questa regola si applica anche ai dati linguistici, che, se in formato grezzo, potrebbero portare dei problemi da non sottovalutare in fase di processamento. Il Data Cleaning, fase di preprocessing che si occupa di eliminare informazioni non necessarie e, insieme, normalizzare, formattare e standardizzare i dati, si rivela perciò essere un passo fondamentale per garantire la qualità dei dati che verranno processati dall'algoritmo. Per quanto essenziale questa fase possa essere, prima di scegliere una modalità o uno strumento di preprocessing è bene considerare, non solo i dati in possesso, ma anche il task che si vuole svolgere con

quei dati. Per alcuni task una tecnica di cleaning può essere fortemente consigliata, se non addirittura obbligatoria, per altri se ne può fare a meno senza che venga compromessa l'accuratezza del modello finale [8]. Ecco una lista dei task di data cleaning:

- Rimozione dei noisy data
- Lowercasing (formattazione del testo in lettere minuscole)
- Tokenizzazione: per frasi e per parole
- Eliminazione delle stopword
- Normalizzazione del testo
- Stemming (processo di riduzione di una parola alla sua radice verbale che si appone a suffissi e prefissi)
- Lemmatizzazione (riduzione di una parola alla radice da cui deriva mantenendo una forma standard per essa)

4.2.3 Data annotation

I dati di addestramento devono essere classificati e annotati correttamente per un caso d'uso specifico. In particolare in ambito di NLP prendono particolare importanza informazioni come le relazioni tra i dati che si hanno a disposizione e il loro peso all'interno dell'intero sistema. Si tratta in genere di dati aggiuntivi che vengono ricavati attraverso ulteriori computazioni partendo da quelli di cui si è già in possesso. La funzione seguente fornisce un esempio di come questi metadati vengono generati.

tf-idf

La funzione di peso tf-idf (term frequency–inverse document frequency) viene utilizzata in information retrieval per misurare l'importanza di un termine rispetto ad un documento o ad una collezione di documenti [1]. Tale funzione aumenta proporzionalmente al numero di volte che il termine è contenuto nel documento, ma cresce

in maniera inversamente proporzionale con la frequenza del termine nella collezione. L'idea alla base di questo comportamento è di dare più importanza ai termini che compaiono nel documento, ma che in generale sono poco frequenti. La funzione può essere scomposta in due fattori: Il primo fattore della funzione è il numero dei termini presenti nel documento. In genere questo numero viene diviso per la lunghezza del documento stesso per evitare che siano privilegiati i documenti più lunghi.

$$tf_{i,j} = \frac{n_{i,j}}{|d_j|}$$

dove $n_{i,j}$ è il numero di occorrenze del termine i nel documento j, mentre il denominatore $|d_j|$ è semplicemente la dimensione, espressa in numero di termini, del documento j. L'altro fattore della funzione indica l'importanza generale del termine i nella collezione:

$$idf_i = log_{10} * \frac{|D|}{|\{ d : i \in d\}|}$$

dove $|D|$ è il numero di documenti nella collezione, mentre il denominatore è il numero di documenti che contengono il termine i. Abbiamo quindi che:

$$(tf - idf)_{i,j} = tf_{i,j} * idf_i$$

4.2.4 Data reduction

Questa riduzione si riferisce semplicemente al processo di diminuzione del numero di attributi in un dataset mantenendo quanto più possibile la varietà nel set originale [12]. Spesso succede che anche utilizzando tecniche avanzate non si riesce a preservare tutta la varietà iniziale, ma ovviamente ci sono dei vantaggi da tenere in considerazione:

- Un numero inferiore di dimensioni nei dati significa meno tempo di addestramento e meno risorse di calcolo, aumentando le prestazioni complessive degli algoritmi di apprendimento automatico;
- Si evita il problema dell'overfitting;
- Risulta molto utile per la visualizzazione dei dati;
- C'è la possibilità di rimuovere noisy data.

4.3 Metodo di generazione incrementale

Questo metodo si propone di mostrare una possibile soluzione all'immenso lavoro manuale necessario alla creazione di un dataset atto ad allenare un classificatore testuale. Questo lavoro ovviamente non viene eliminato del tutto, ma utilizzato soltanto nelle fasi preliminari della generazione per garantire una base solida sugli argomenti trattati. Data la natura ricorsiva e perciò articolata del funzionamento, può essere utile definire alcune assunzioni e un'idea astratta dell'algoritmo prima di dedicarsi allo studio del codice specifico.

4.3.1 Assunzioni fondamentali

Trattandosi di un metodo incrementale e ricorsivo, il primo punto da tenere in considerazione è sicuramente quello del "caso base", infatti affinchè si proceda all'espansione automatica delle informazioni contenute nel dataset il programma deve essere già in grado, seppure con evidenti limitazioni, di effetturare una classificazione basata sui tag che intendiamo considerare. Per fare questo saranno indispensabili una versione minimale del dataset che sia il più precisa possibile e un classificatore allenato unicamente su di esso.

Dataset minimale

La forma del dataset può variare a seconda delle esigenze, ma per mantenere una formattazione generale e adattabile, verrà considerato come caso di esempio un documento che contiene triple del tipo *"verb object weight"*, una per ogni riga.

```
defeat poverty 1
demonstrate crisis 1
demonstrate importance 1
determine state 1
develop community 1
develop country 1
develop knowledge 1
```

Figura 4.1 Forma del dataset

Notiamo che sarà necessario un documento di questa forma per ognuno dei tag di cui va effettuato il riconoscimento. Ci sono due possibili modi per ottenere questo dataset minimale:

- **Costruzione completamente manuale:** inserimento manuale di tutte e tre le componenti;
- **Costruzione semiautomatica:** Sfruttando un algoritmo capace di estrarre le coppie verbo-oggetto significative da un testo, sarebbe necessario fornire un set di frasi appropriate per ogni tag e un metodo di calcolo standardizzato per il peso da attribuire alla coppia.

Assumiamo ovviamente che le informazioni contenute inizialmente nel dataset siano corrette. Possiamo farlo vista l'esigua quantità e il fatto che sono state valutate manualmente.

Funzione per la classificazione

L'utilizzo di coppie verbo-oggetto per la classificazione testuale non rappresenta uno standard operativo ed è l'argomento principale trattato nella tesi del co-autore di SDG Detector. Per i nostri scopi invece ci limiteremo a considerare una funzione ***check_ sdg(text)***, capace di capire quali SDG possono essere associati ad un testo in esame, senza andare nello specifico sui suoi dettagli implementativi.
Ciò che invece è importante capire è che nel caso base il classificatore cui si affida questa funzione è allenato utilizzando il dataset minimale descritto precedentemente.

4.3.2 Idea astratta del funzionamento

L'intera idea di funzionamento si basa su un principio di località delle informazioni piuttosto astratto:
"Se mi aspetto che questo testo parli di uno specifico argomento perchè ho un riscontro con ciò che è attualmente contenuto nel mio dataset, è molto probabile che le informazioni extra presenti nel testo in questione possano essermi di aiuto a riconoscere nuovamente questo argomento in futuro".
Questo principio rappresenta allo stesso tempo il principale punto di debolezza e di

forza del metodo: per renderci conto del primo punto basta notare che la valutazione di un documento che tratta una serie di argomenti sconnessi potrebbe generare confusione e portare a classificazioni errate. Possiamo però notare anche che, nonostante questo non sia legato a regole rigide, sappiamo che la grande maggioranza dei documenti rispetta comunque un filo logico ben delineato. Per minimizzare il rumore generato dalla classificazione di testi con argomenti eterogenei, i documenti vengono divisi in paragrafi che saranno analizzati singolarmente.

4.3.3 Procedimento

Il procedimento operativo può essere riassunto nei seguenti punti:

- Creazione del dataset minimale
- Allenamento del classificatore sul dataset minimale
- Ricerca dei documenti per la generazione
  - Costruzione delle query
  - Acquisizione degli url risultanti dalle query
  - Utilizzo degli url per ricavare il testo
  - Formattazione dei testi e divisione in paragrafi
  - Indicizzazione e ed etichettizzazione dei documenti
- Analisi dei documenti e espansione del dataset
  - Utilizzo del classificatore (funzione check_sdg()) sul singolo paragrafo
  - Valutazione del risultato dell'analisi e preparazione ai due scenari possibili:
    - Esito positivo della classificazione: si ricavano le coppie verbo-oggetto dal paragrafo, si confrontano con gli elementi presenti in una blacklist e eventualmente si aggiungono al dataset relativo all'etichetta cui fanno riferimento (in ordine alfabetico ed evitando l'inserimento di duplicati);
    - Esito negativo della classificazione: nessuna operazione particolare.

Esempio esplicativo

Consideriamo una situazione come quella in figura su di un testo in esame frutto di una ricerca per uno specifico argomento.

Figura 4.2 Esempio

Quelle sottolineate sono le coppie verbo-oggetto presenti nel testo e sono divise in questo modo:

- **In verde:** troviamo le coppie che sono già salvate nel nostro dataset e servono per il riconoscimento dell'argomento in questione.
- **In rosso:** troviamo le coppie che fanno riferimento all'argomento in esame ma che non fanno parte del nostro dataset (quindi le coppie che vorremmo raccogliere).
- **In blu:** troviamo le coppie che non c'entrano nulla con l'argomento in analisi.

L'avvenuto riconoscimento dell'argomento nei paragrafi indica che tutte le coppie non verdi presenti in questi verranno aggiunte al dataset.

Nota sulla divisione del testo

Analizzando l'esempio è facile notare l'importanza della divisione in paragrafi del testo, infatti senza questa accortezza l'analisi della pagina presa per l'esempio avrebbe dato sicuramente esito positivo e le coppie blu presenti sul fondo sarebbero state aggiute al dataset. Quello in figura potrebbe sembrare un caso particolare, ma bisogna sempre ricordare che tutti i documenti in esame vengono estrapolati da pagine web ed è quindi molto probabile che siano presenti altri articoli, pubblicità, intestazioni del sito e molti altri elementi che non vorremmo aggiungere al nostro dataset.

4.3.4 Considerazioni supplementari

Di seguito sono elencate alcune integrazioni che possono essere aggiunte all'idea di base per migliorarne le performance, a seconda deil caso secifico:

Filtri per coppie ed elementi indesiderati

L'utilizzo di una blacklist si rende necessario nel momento in cui durante la costruzione del dataset si presenta una grande quantità di coppie prive di significato o non associate all'argomento in questione. Praticamente le coppie blu presenti nell'esempio descritto precedentemente. L'idea migliore è probabilmente quella di implementare una lista di coppie, una di verbi e una di nomi da analizzare ogni qual volta si sta aggiungendo un elemento al dataset.

Allenamento ricorsivo del classificatore

L'idea è semplicemente quella di allenare nuovamente il classificatore da zero ogni qual volta il dataset subisce un'espansione significativa, in modo da renderlo capace di riconoscere un numero maggiore di coppie nei testi analizzati in seguito. Questo metodo porta però alcuni fattori da considerare, primo tra tutti il costo in termini di tempo necessario all'allenamento, in quanto questo è proporzionale alla dimensione stessa del dataset e inserire una soglia troppo bassa risulterebbe nel passare più tempo ad allenare il classificatore che a raccogliere informazioni. Un altro punto da considerare è il problema noto in letteratura come *AI Catastrophic forgetting*,

ovvero la tendenza di una rete neurale artificiale a dimenticare completamente e bruscamente le informazioni apprese in precedenza dopo aver appreso nuove informazioni [7]. Volendo analizzare cosa accomuna i due casi, in questa idea di training ricorsivo sicuramente c'è da considerare che il lavoro di allenamento effettuato in tutte le analisi precedenti a quella corrente andrebbe sprecato. Dopo una serie di riflessioni è stata elaborata la seguente conclusione: sarebbe senz'altro utile riuscire ad integrare tutte le nuove informazioni senza dover allenare nuovamente il classificatore, ma questo metodo riguarda strettamente la costruzione del dataset, perciò possiamo considerare che il fine ultimo venga correttamente raggiunto, seppur con un grande dispendio di risorse extra. Una volta che il dataset ha raggiunto una dimensione sufficiente si può disattivare questa funzionalità, in modo da permettere al classificatore di continuare ad operare e apprendere, ma con una solida base.

4.3.5 Considerazioni sulla valutazione dei risultati

Per effettuare un analisi concreta sulle performance e l'accuratezza di generazione si possono prendere in considerazione i risultati ottenuti nell'ambito dell'applicazione che questa tesi utilizza come caso di studio e che viene trattata più amipamente nel capitolo successivo. È importante notare che i fattori che influenzano queste due caratteristiche sono strettamente legate alla reperibilità delle informazioni stesse e alla qualità dei documenti che è possibile trovare attraverso una ricerca in rete.

5 Caso di studio: SDG Detector

5.1 Sustainable Development Goals

L'Agenda 2030 per lo sviluppo sostenibile, adottata da tutti gli Stati membri delle Nazioni Unite nel 2015, fornisce un progetto condiviso per la pace e la prosperità per le persone e il pianeta, ora e in futuro. Al centro ci sono i 17 Obiettivi di sviluppo sostenibile (SDG), che sono un urgente invito all'azione da parte di tutti i paesi - sviluppati e in via di sviluppo - in un partenariato globale [20].
Essi riconoscono l'importanza di combattere la povertà ed elaborare strategie che migliorano la salute e l'istruzione, riducono le disuguaglianze e stimolano la crescita economica, il tutto affrontando il cambiamento climatico e lavorando per preservare i nostri oceani e le nostre foreste.

5.1.1 Cosa dicono i SDG

Come già detto gli obiettivi sono 17 e si dividono a loro volta in 169 "traguardi" specifici da raggiungere entro il 2030. Ognuno dei traguardi è a sua volta fornito di un numero variabile di identificatori utilizzati per determinare l'avvenuto raggiungimento del traguardo. Ecco una lista dei SDG:

- **No poverty**

  Porre fine alla povertà in tutte le sue forme ovunque.

Figura 5.1, SDG numero 1

- **Zero hunger** Porre fine alla fame, raggiungere la sicurezza alimentare e una migliore nutrizione e promuovere un'agricoltura sostenibile,

Figura 5.2, SDG numero 2

- **Good health and well-being** Garantire una vita sana e promuovere il benessere per tutti a tutte le età.

Figura 5.3, SDG numero 3

- **Quality education** Garantire un'istruzione di qualità inclusiva ed equa e promuovere opportunità di apprendimento longevo per tutti.

Figura 5.4, SDG numero 4

- **Gender equality** Raggiungere l'uguaglianza di genere e responsabilizzare tutte le donne e le ragazze.

Figura 5.5, SDG numero 5

- **Clean water and sanitation** Garantire la disponibilità e la gestione sostenibile dell'acqua e dei servizi igienico-sanitari per tutti.

Figura 5.6, SDG numero 6

- **Affordable and clean energy** Garantire a tutti l'accesso a un'energia conveniente, affidabile, sostenibile e moderna.

Figura 5.7, SDG numero 7

- **Decent work and economic growth** Promuovere una crescita economica sostenuta, inclusiva e sostenibile, un'occupazione piena e produttiva e un lavoro dignitoso per tutti.

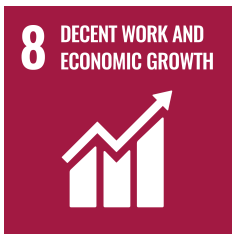

Figura 5.8, SDG numero 8

- **Industry, innovation and infrastructure** Costruire infrastrutture resilienti, promuovere un'industrializzazione inclusiva e sostenibile e promuovere l'innovazione.

Figura 5.9, SDG numero 9

- **Reduced inequalities** Ridurre la disuguaglianza all'interno e tra i paesi.

Figura 5.10, SDG numero 10

- **Sustainable cities and communities** Rendi le città e gli insediamenti umani inclusivi, sicuri, resilienti e sostenibili.

Figura 5.11, SDG numero 11

- **Responsible consumption and production** Garantire modelli di produzione e consumo sostenibili.

Figura 5.12, SDG numero 12

- **Climate action** Adottare misure urgenti per combattere il cambiamento climatico e i suoi impatti.

Figura 5.13, SDG numero 13

- **Life below water** Conservare e utilizzare in modo sostenibile gli oceani, i mari e le risorse marine per lo sviluppo sostenibile.

Figura 5.14, SDG numero 14

- **Life on land** Proteggere, ripristinare e promuovere l'uso sostenibile degli ecosistemi terrestri, gestire in modo sostenibile le foreste, combattere la desertificazione, arrestare e invertire il degrado del suolo e fermare la perdita di biodiversità.

Figura 5.15, SDG numero 15

- **Peace, justice and strong institutions** Promuovere società pacifiche e inclusive per lo sviluppo sostenibile, fornire accesso alla giustizia per tutti e costruire istituzioni efficaci, responsabili e inclusive a tutti i livelli.

Figura 5.16, SDG numero 16

- **Partnerships for the goals** Rafforzare i mezzi di attuazione e rivitalizzare il partenariato globale per lo sviluppo sostenibile.

Figura 5.17, SDG numero 17

5.1.2 L'importanza di riconoscere i SDG

Le imprese di tutto il mondo, di qualsiasi dimensione e settore produttivo, sono chiamate a dare un contributo importante attraverso nuovi modelli di business responsabile, gli investimenti, l'innovazione, lo sviluppo tecnologico e l'attivazione di collaborazioni. Diventa quindi una sfida importante quella di riuscire a rilevare quali obiettivi sono citati all'interno di un documento o una pagina web di un'azienda per capire quanto questa si dedichi allo sviluppo sostenibile.

5.2 Obiettivo dell'applicazione

SDG Detector è un'applicazione sviluppata in Python che permette di classificare un numero arbitrario di documenti di testo e pagine disponibili in rete determinando se all'interno vi sono correlazioni con uno o più Sustainable Development Goals. Il codice sorgente può essere consultato al seguente url:
https://github.com/StefanoColamonaco/SDG_Detector [17]

5.2.1 Input

L'applicazione prende in input un file JSON chiamato "urlsList.json". Esso contiene un array di oggetti nella forma espressa nell'immagine seguente:

Figura 5.18 Formato di input dell'applicazione

Dove:

- **type** serve ad indicare se il target dell'URL è una pagina web ("S") o un file txt ("F");
- **url** è una stringa contenente l'indirizzo della pagina.

5.2.2 Output

L'output consiste in una semplice visualizzazione a schermo dei risultati derivanti dall'analisi di ogni documento. In particolare verrà mostrato l'url di riferimento del documento analizzato seguito da 17 righe, ognuna identificante un SDG e una spunta verde o una X rossa a seconda che quest'ultimo sia stato riconosciuto nel testo durante l'analisi.

5.2.3 Fasi del funzionamento

Nel funzionamento si possono distinguere tre fasi principali:

- Estrapolazione del testo dal sito: avviene in modo diverso a seconda del valore "type" precedentemente descritto, ma in entrambi i casi il testo presente viene estrapolato e spogliato da eventuali tag o caratteri superflui attraverso librerie apposite (BeautifulSoup).
- Inizializzazione dei dati per il matching: viene utilizzata la libreria Natural Language ToolKit di Python per allenare un Naive Bayes Classifier su ogni

SDG sfruttando le informazioni presenti nel dataset e i documenti che sono già stati classificati.

- Nell'ultima fase viene utilizzato il classificatore precedentemente inizializzato per verificare se le corrispondenze nel testo in input sono quantitativamente e qualitativamente sufficienti a dare per assodata la presenza del SDG. In caso affermativo verrà fornito un ouput come desciritto nella sezione precedente.

5.3 Popolazione del dataset

5.3.1 Ricerca e formattazione dei documenti di allenamento

Per quanto riguarda la parte di ricerca dei documenti, questa è stata compiuta attraverso un semplice algoritmo di generazione di query specifico per l'ambito di cui l'applicazione si occupa. I dati utilizzato sono: un array contenente una serie di keywords identificanti i 17 SDG e uno contenente i nomi delle compagnie che vengono prese in esame per la ricerca di documenti.

SDGOBJECTIVES = ["No poverty", "Zero hunger", "Good health and well-being", "Quality education", "Gender equality", "Clean water and sanitation", "Affordable and clean energy", "Decent work and economic growth", "Industry, innovation and infrastructure", "Reduced inequalities", "Sustainable cities and communities", "Responsible consumption and production", "Climate action", "Life below water", "Life on land", "Peace, justice and strong institutions", "Partnerships for the sustainable goals"]

TRAININGCOMPANIES = ["Tesla", "Adidas","Nike","Adobe", "Ibm", "Nvidia","Wikipedia","Unibo", "Google", "Microsoft"]

Da questi dati, viene generata una query come semplice concatenazione di stringhe e viene passata ad una funzione che sfrutta le API di Google per ricavare gli url delle pagine che rispondono alla query. Il numero degli url utilizzati dipende dal valore della variabile DOCSPERSDG e questi ultimi vengono poi passati ad una funzione *site* che sfruttando il pacchetto *requests* di Python e *BeautifulSoup* ricava il testo dalla pagina web e lo rende manipolabile facilmente eliminando tutti i tag html.

```
def site(url):
    headers = {'User-Agent': ''}    # Valid header needed
    response = requests.get(str(url),headers=headers)
    html = response.text
    soup = BeautifulSoup(html, features="html.parser")
    text = soup.get_text()
    return text
```

Una volta ottenuti, questi testi vengono liberati da eventuali caratteri speciali residui, divisi in paragrafi da un numero fissato di frasi e infine indicizzati e inseriti in appositi file JSON. Contemporaneamente viene aggiornato un file speciale chiamato *documentsRegister.json* contenente informazioni importanti su ognuno dei testi salvati, in particolare: nome con cui il documento è stato salvato, SDG cui fa riferimento, compagnia utilizzata nella query e url del sito.

Figura 5.19 dataRegister.json (immagine parziale)

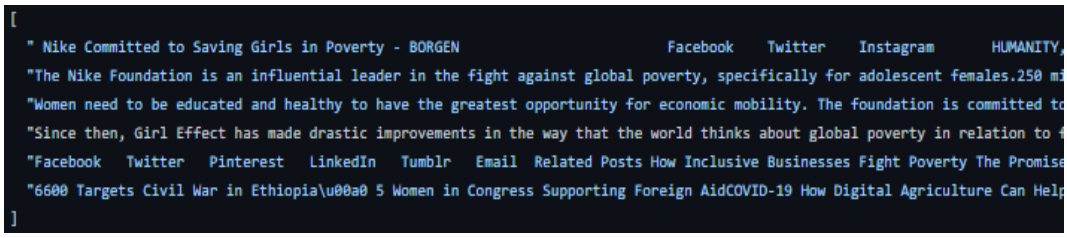

Figura 5.20 Esempio di documento salvato (immagine parziale)

Vale la pena soffermarsi ad analizzare il meccanismo di divisione dei documenti in gruppi di frasi. I benefici di questa pratica sono già stati esposti all'interno del capitolo precedente, ma resta da definire il numero ottimale di frasi che andranno a

comporre ognuno dei frammenti. Questo numero dovrebbe essere scelto in modo da minimizzare il numero di falsi positivi durante la classificazione, ma deve essere abbastanza grande da consentire al classificatore di comprendere meglio il contesto attorno al quale si pone una frase o un insieme di esse. Nell'ambito di questo progetto è stato deciso di utilizzare un numero fissato pari a 5 frasi per frammento, un numero frutto di un'analisi riguardante il numero medio di frasi necessarie ad esprimere un'idea di media complessità. Probabilmente però questo non è il modo giusto di scegliere questo numero vista la sua importanza ai fini dell'estrapolazione delle coppie e sarà necessaria un'ulteriore riflessione in ambito di future works.

5.3.2 Classificazioni intermedie, estrazione delle coppie e aggiornamento del dataset

Una volta ottenuta una solida base di documenti a disposizione si può passare al livello successivo, ovvero utiizzarli per espandere il dataset. Il primo passo è quello di configurare e allenare il classificatore sul dataset corrente. Non andremo nel dettaglio su come avvengono questi due passaggi in quanto sono trattati esplicitamente nella tesi del co-produttore di questa applicazione e come già spiegato nel capitolo precedente utilizzeremo la funzione *check_sdg(text)* come black box. L'unica supposizione considerata è che questa funzione analizza il testo in input e restituisce in output un array di 17 elementi dove in posizione *i-1* troviamo il valore 1 se è stato riconosciuto il SDG *i*, 0 altrimenti.

I passi successivi sono riassumibili in questo modo: per ognuno dei documenti a disposizione si considerano le informazioni recuperate dal file *dataRegister* e tutti i frammenti che lo compongono presi singolarmente. Ogni frammento viene dato in input alla funzione di classificazione e si controlla che il goal associato al documento sia stato riconosciuto all'interno di esso. In caso affermativo si procede con l'estrapolazione delle coppie verbo-oggetto presenti nel testo in esame, le si ordinano alfabeticamente, si rimuovono eventuali duplicati e si procede a sovrascrivere il dataset corrente inserendo le nuove informazioni.

Passiamo ora a descrivere la funzione che si occupa dell'estrapolazione delle coppie verbo-oggetto:

```
nlp = stanza.Pipeline(lang='en', processors='tokenize,mwt,pos,
    lemma,depparse,constituency')


    def vrbobj_pairs(text):
        try:
            doc = nlp(text)
            allPairs = []
            for sentence in doc.sentences:
                pairs = extrapolatePairs(sentence.words)
                allPairs = allPairs + pairs
            return allPairs
        except:
            print("Error in constituency parsing")
            return []
```

Commentiamo il codice appena mostrato. L'input di questa funzione non è l'intero testo presente in una pagina, bensì solo uno dei frammenti che fanno parte del documento e che sono stati riconosciuti dal classificatore sul goal specificato. Il costrutto *try-except* si rende necessario in quanto la funzione *nlp* potrebbe generare errori durante la fase di constituency parsing e in tal caso il testo viene ignorato. Grazie all'oggetto generato da stanza possiamo manipolare facilmente il testo in input, dividendolo in frasi e andando ad analizzarle una per volta. Il passo successivo, nonchè il fulcro dell'operazione di estrapolazione delle coppie è dato dalla funzione *extrapolatePairs*, che esegue i seguenti passi:

- Ottiene una lista delle parole appartenenti alla frase che abbiano un POS tag di tipo NOUN (nome).
- Per ognuno degli elementi della lista, si risale il Constituency Parsing Tree generato da Stanza per la frase corrente fino a raggiungere il primo predicato verbale associato al nome in analisi.

- Una volta ottenuti, questi due elementi vengono formattati attraverso le consuete operazioni di stemming e lemmatzzazione, si controlla che la coppia risultante sia valida (sfruttando una blacklist) e infine si assegna loro un peso.

Per quanto riguarda l'assegnazione del peso, la versione attuale dell'applicazione associa un peso fissato a tutte le coppie valide. Sarebbe interessante, magari attraverso approfondimenti o lavori correlati, esplorare metodi che vadano a migliorare le performance del classificatore attraverso una valutazione precisa dei pesi da assegnare (ad esempio utilizzando tf-idf o metodi sperimentali).

5.3.3 Analisi dei risultati della generazione

Questa analisi prende in considerazione 3 step che presi insieme descrivono la costruzione del dataset finale:

Versione iniziale

Questa versione rappresenta il dataset minimale che serve all'allenamento iniziale del classificatore. Per ottenerla sono state date in input all'algoritmo di estrapolazione delle coppie verbo-oggetto tutte le frasi presenti sul sito sdgs.un.org [20] nelle sezioni **Targets and indicators** e **Progress and info**. Notiamo che la natura delle frasi utilizzate e un attento controllo delle coppie risultanti sono stati fattori fondamentali affinchè si avesse una base solida e precisa su cui fondare le analisi successive. Di seguito è riportata una tabella che mostra la distribuzione delle coppie per i vari SDG:

Numero del SDG	Numero di coppie
1	95
2	31
3	31
4	23
5	20
6	24
7	13
8	32
9	28
10	27
11	27
12	39
13	22
14	47
15	42
16	24
17	59

Per un totale di 584 coppie complessive.

Ecco un grafico che mostra la distribuzione di queste ultime:

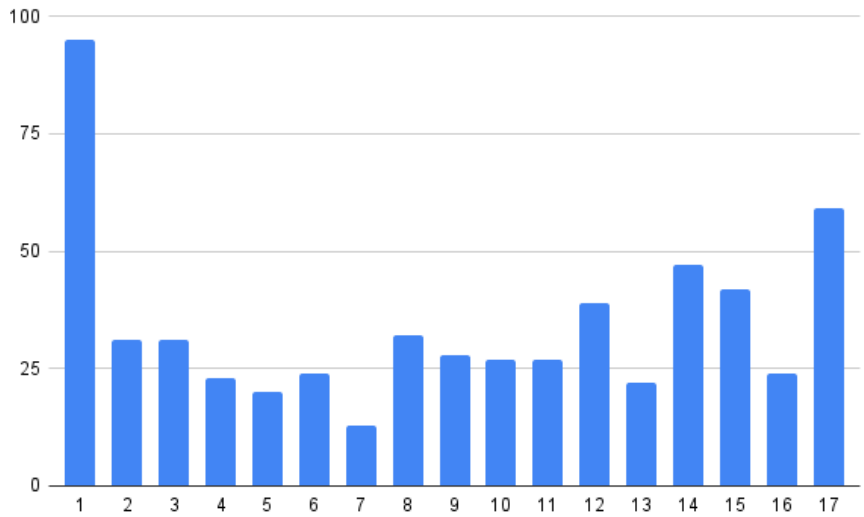

Figura 5.21 Distribuzione delle coppie per il dataset iniziale

Versione 1.0

Questa versione è stata ottenuta allenando il classificatore sul dataset iniziale e facendolo lavorare su 68 documenti. Questo numero deriva dall'utilizzo di una lista composta da 4 compagnie scegliendo un solo documento per ogni SDG associato alla compagnia. Le compagnie considerate sono: "Wikipedia", "Unibo", "Google" e "Microsoft".

Per questa generazione è stata utilizzata una divisione in paragrafi composti da 5 frasi ciascuno.

Numero del SDG	Numero di coppie
1	240
2	118
3	289
4	121
5	50
6	93
7	52
8	140
9	136
10	118
11	209
12	200
13	22
14	131
15	42
16	128
17	75

Per un totale di 2164 coppie complessive e una crescita rispetto alla versione precedente del 270.54%.

Ecco un grafico che mostra la distribuzione delle coppie in relazione con il passo precedente:

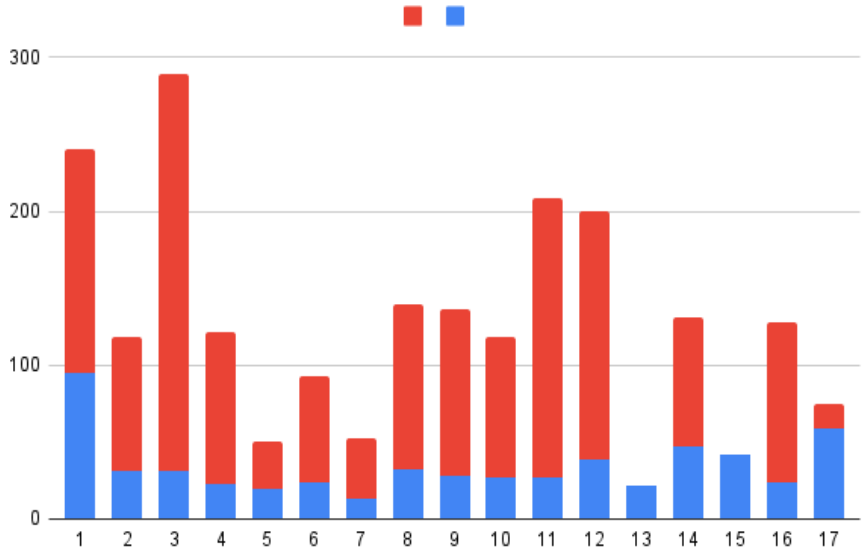

Figura 5.22 Distribuzione delle coppie 1.0

Una volta terminata la costruzione di questa versione del dataset è stata effettuata un'analisi approfondita della correttezza delle coppie presenti all'interno di esso per ognuno degli SDG. Da questa analisi sono risultate le seguenti 49 coppie fuori contesto, alcune delle quali si ripetevano più volte all'interno delle raccolte:

account percent, address challenge, aggregate sex, attribute percent, be addition, be datum, be indicator, be people, be year, benefit form, build disaster, build means, cart cart, cart item, charge motorcycle, charge rider, come head, contribute percent, cover percent, cover cent, drop percent, end 1.b, end dimension, end ed, end form, end way, ensure ed, fall percent, fall worker, go bed, go night, go people, have datum, have percent, help datum, increase month, jump jump, lack half, leave one, maintain ed, mantain title, meet need, occur day, project percent, push shock, reduce decade, report total, tabl proposal, waste child

In seguito queste coppie sono state eliminate dalle raccolte del dataset e aggiunte alla blacklist.

Versione 2.0 - finale

Questa volta il classificatore è stato allenato sul dataset 1.0 e ha avuto a disposizione 130 documenti derivanti da 8 compagnie. Moltiplicando i 17 SDG per le 8 compagnie il risultato dovrebbe essere 136, ma di questi documenti ne sono stati eliminati 6 poichè erano in formato PDF e di difficile lettura. Le compagnie prese in esame

questa volta sono state (in quest ordine): “Nike”, ”Adobe”, ”Ibm”, “Nvidia”, "Wikipedia", "Unibo", "Google", "Microsoft". Ecco il registro aggiornato delle coppie a disposizione:

Numero del SDG	Numero di coppie
1	285
2	107
3	344
4	197
5	212
6	189
7	79
8	174
9	141
10	135
11	327
12	220
13	54
14	179
15	72
16	166
17	80

Per un totale di 2961 coppie complessive. Un simile rallentamento nell’espansione del dataset rispetto al passo precedente potrebbe sembrare frutto di errori di valutazione, ma ci sono da considerare due fattori fondamentali:

- Vista la conformazione della query bisogna considerare che metà dei documenti utilizzati erano già stati classificati una volta, perciò il loro rendimento molto probabilmente è stato inferiore;
- Rispetto al risultato della versione 1 bisogna considerare l’eliminazione delle coppie aggiunte alla blacklist, alcune delle quali comparivano in più di un

SDG, per un totale di 98 occorrenze eliminate e riducendo il numero di coppie di partenza a 2066.

Quindi la crescita (dopo l'intervento della blacklist) è stata del 43%. Notiamo inoltre che vista la natura stessa del problema, la specificità delle etichette, e visto l'utilizzo di tecniche per evitare la comparsa di duplicati e sinonimi (lemmatizzazione e stemming), la crescita del dataset è limitata.
Ecco un grafico che mostra la distribuzione delle coppie in relazione con i passi precedenti:

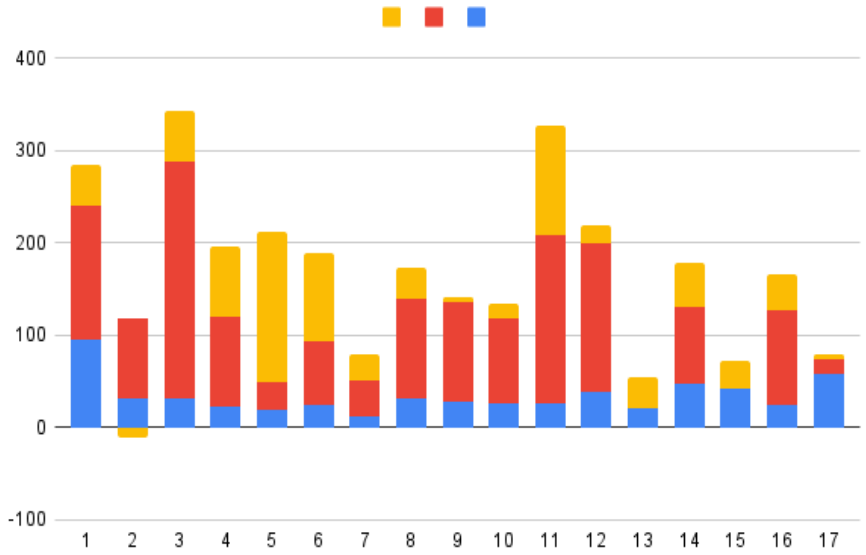

Figura 5.23 Distribuzione delle coppie 2.0

La decrescita delle coppie nel SDG 2 è dovuta alla pulizia avvenuta in seguito al passo precedente.
È interessante notare che non vi sono particolari relazioni tra il numero di coppie nel dataset per una specifica etichetta e la crescita che queste avranno in termini di volume rispetto alle altre. Probabilmente il tutto è legato semplicemente all'ampiezza dell'argomento e alla qualità dei documenti utilizzati.

Versione 3.0 - tentativo

In questa versione sono state mantenute la stessa configurazione utilizzata per costruire la 2.0 e gli stessi documenti, ma con il dataset aggiornato. Il risultato dimostra le idee espresse nelle note della versione precdente, infatti nonostante i paragrafi

analizzati siano stati classificati correttamente non è stata aggiunta alcuna coppia. Di seguito un diagramma che mostra l'andamento della crescita del dataset nelle quattro fasi esposte:

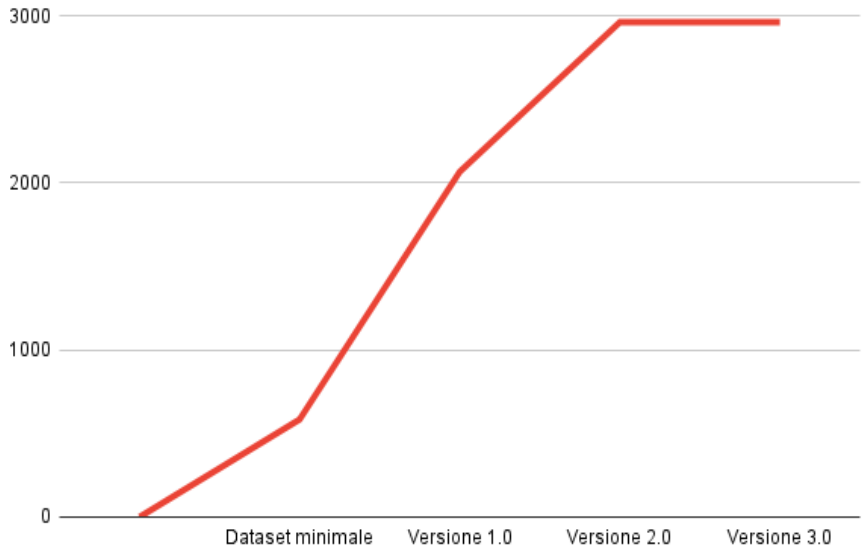

Figura 5.24 Andamento della crescita del dataset

5.4 Classificazione e output

Come già espresso in più occasioni all'interno del documento, la presente non rivolge particolare attenzione al classificatore e al fine ultimo dell'applicazione, bensì alla sola costruzione del dataset su cui essa si basa. Una volta ottenuto un dataset dalla dimensione discreta e un buon numero di documenti di test, costituiti dagli stessi utilizzati per l'espansione, si può passare ad utilizzare il classificatore su documenti reali presi dal web come descritto nella sezione 5.2.1. Il co-autore dell'applicazione si è occupato di testare modelli e input differenti per valutare l'accuratezza dell'applicazione, ottenendo in media buoni risultati che sono riportati all'interno del suo elaborato.

6 Conclusioni

La presente tesi si poneva due obiettivi: spiegare il metodo sperimentale di costruzione ed esporre un'applicazione pratica di quest ultimo attraverso il tool ***SDG Detector***. Possiamo concludere il documento affermando che entrambi gli obiettivi sono stati raggiunti con buoni risultati, il metodo esposto permette infatti la costruzione automatica di un dataset di coppie verbo-oggetto utili alla classificazione testuale a partire da un insieme minimale di informazioni. Lo studio delle performance di generazione ha prodotto ottimi risultati sia in termini di volume che di accuratezza, raggiungendo una crescita complessiva che si avvicina al 500% su un numero limitato di espansioni computabili anche da un elaboratore di media gamma. Il metodo nella sua implementazione si è rivelato efficace permettendo di implementare una piccola base di conoscenza utilizzabile per lo studio dei SDG e una versione sufficientemente performante di SDG Detector in grado di classificare documenti con una precisione al livello di altri famosi algoritmi noti in letteratura. Questo tipo di considerazioni sono però oggetto della tesi del co-produttore dell'applicazione, Filip Radovic. Per quanto riguarda l'utilità effettiva del software, SDG Detector rappresenta un importante strumento per lo studio e la classificazione di aziende, con particolare attenzione alla loro sostenibilità, un argomento sempre più presente nell'informatica e nella vita di tutti i giorni.

6.1 Limitazioni e studi futuri

Il metodo e la sua implementazione nella forma attuale non sono completi, infatti ci sono ancora alcune funzionalità da implementare e delle caratteristiche da migliorare:

- **Generalizzare anche il metodo di costruire le query:** al momento a logica per le query che ricercano i documenti di allenamento va costruita in modo manuale, mentre sarebbe utile avere la possibilità di costruirle automaticamente semplicemente fornendo un contesto e le etichette specifiche.

- **Studio dei documenti PDF:** nella versione attuale i documenti PDF vengono scartati, sarebbe invece utile combinare la ricerca con un software in grado di leggere questo tipo di documenti, che solitamente risultano anche strutturati meglio rispetto alle pagine web.

- **Numero di frasi che compongono un frammento:** attualmente questo numero è fisso per ogni documento, ma come espresso precedentemente questa potrebbe non essere la strada giusta. Si potrebbe infatti pensare di rendere questo numero adattivo rispetto alla dimensione o alla struttura del documento che si sta analizzando.

- **Implementazione di un metodo per la gestione dei pesi:** al momento viene assegnato il medesimo peso a tutte le coppie presenti all'interno del dataset, mentre una valutazione più accurata della vicinanza tra la coppia e l'argomento in esame e l'assegnamento di un peso che dipende da questa potrebbero portare a migliorare notevolmente l'accuratezza del classificatore (ad esempio tf-idf).

- **Implementazione di una blacklist a riempimento automatico:** avere una blacklist formata correttamente consente di diminuire sensibilmente il numero di coppie errate all'interno del dataset e per questa via consente di aumentare sensibilmente l'accuratezza dei classificatori che si basano su di esso. Al momento la blacklist può essere riempita solo manualmente, ma se si trovasse un metodo per espanderla automaticamente, magari attraverso il valore dei pesi o meccanismi di riconoscimento del contesto, sicuramente si avrebbero notevoli vantaggi.

Bibliografia

[1] Akiko Aizawa. «An information-theoretic perspective of tf–idf measures». In: *Information Processing & Management* 39.1 (2003), pp. 45–65. ISSN: 0306-4573.

[2] Erik Cambria e Bebo White. «Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]». In: *IEEE Computational Intelligence Magazine* 9.2 (2014), pp. 48–57.

[3] KR1442 Chowdhary. «Natural language processing». In: *Fundamentals of artificial intelligence* (2020), pp. 603–649.

[4] A Famili et al. «Data preprocessing and intelligent data analysis». In: *Intelligent data analysis* 1.1 (1997), pp. 3–23.

[5] Salvador García, Julián Luengo e Francisco Herrera. *Data preprocessing in data mining*. Vol. 72. Springer, 2015.

[6] Julia Hirschberg e Christopher D Manning. «Advances in natural language processing». In: *Science* 349.6245 (2015), pp. 261–266.

[7] James Kirkpatrick et al. «Overcoming catastrophic forgetting in neural networks». In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[8] Sanjay Krishnan et al. «Activeclean: Interactive data cleaning for statistical modeling». In: *Proceedings of the VLDB Endowment* 9.12 (2016), pp. 948–959.

[9] Edward Loper e Steven Bird. «Nltk: The natural language toolkit». In: *arXiv preprint cs/0205028* (2002).

[10] Christopher D Manning et al. «The Stanford CoreNLP natural language processing toolkit». In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014, pp. 55–60.

[11] Filipe Mesquita et al. «KnowledgeNet: A Benchmark Dataset for Knowledge Base Population». In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, nov. 2019, pp. 749–758.

[12] Emily Namey et al. «Data reduction techniques for large qualitative data sets». In: *Handbook for team-based qualitative research* 2.1 (2008), pp. 137–161.

[13] Christopher Olston, Marc Najork et al. «Web crawling». In: *Foundations and Trends® in Information Retrieval* 4.3 (2010), pp. 175–246.

[14] Peng Qi et al. «Stanza: A Python natural language processing toolkit for many human languages». In: *arXiv preprint arXiv:2003.07082* (2020).

[15] Leonard Richardson. «Beautiful soup documentation». In: *Dosegljivo: https://www. crummy. com/software/BeautifulSoup/bs4/doc/.[Dostopano: 7. 7. 2018]* (2007).

[16] Irina Rish et al. «An empirical study of the naive Bayes classifier». In: *IJCAI 2001 workshop on empirical methods in artificial intelligence*. Vol. 3. 22. 2001, pp. 41–46.

[17] *SDG Detector source code*. https://github.com/StefanoColamonaco/SDG_Detector.

[18] Ah-Hwee Tan et al. «Text mining: The state of the art and the challenges». In: *Proceedings of the pakdd 1999 workshop on knowledge disocovery from advanced databases*. Vol. 8. 1999, pp. 65–70.

[19] M Thangaraj e M Sivakami. «Text classification techniques: A literature review». In: *Interdisciplinary Journal of Information, Knowledge, and Management* 13 (2018), p. 117.

[20] *The 17 goals - Sustainable Development*. https://sdgs.un.org/goals.

[21] Kristina Toutanova et al. «Feature-rich part-of-speech tagging with a cyclic dependency network». In: *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. 2003, pp. 252–259.

[22] Bo Zhao. «Web scraping». In: *Encyclopedia of big data* (2017), pp. 1–3.