Abstract

Il processo di revisione paritaria degli articoli di ricerca rappresenta una parte fondamentale nell'editoria accademica; il suo abuso a scopo di ritorno personale può influire sull'ammissione dei lavori presentati alle conferenze scientifiche. L'obiettivo di questa tesi è la costruzione di uno strumento per la raccolta di dati bibliometrici utili ad analisi volte alla ricerca di bias sistemico dovuto alla composizione della commissione di conferenze scientifiche. L'implementazione proposta utilizza tecniche avanzate di Natural Language Processing (NLP) per l'estrapolazione delle informazioni e il riconoscimento delle entità presenti nelle Call For Papers dei maggiori eventi accademici internazionali.

Indice

Elenco delle figure

Elenco dei codici

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 Obiettivo

Ottenere una citazione ad un proprio lavoro in una pubblicazione accademica porta ad un potenziale incremento dell'indice H della persona, risultando di conseguenza in un aumento del posizionamento bibliometrico. Ciò avviene in quanto questa metrica ha lo scopo di essere "un utile parametro con cui confrontare, in modo imparziale, individui diversi che competono per la stessa risorsa quando un importante criterio di valutazione è la realizzazione scientifica" (Hirsch, 2005)[1], che nella pratica rappresenta una componente di valutazione dell'autore in ambiente di ricerca. Per alcune persone può essere quindi importante massimizzare tale fattore, in modo da ottenere maggiore esposizione e riconoscimento da parte della comunità scientifica.

Per poter essere presentati a conferenze accademiche, gli articoli

di ricerca vengono selezionati tramite un processo di peer review da parte di una commissione di valutazione composta da membri della comunità scientifica. Queste pubblicazioni contengono solitamente vari riferimenti a lavori svolti da altri autori; durante la revisione paritaria, la presenza in esse di riferimenti bibliografici ad un membro del comitato di programma potrebbe indurre quest'ultimo - volontariamente o inconsapevolmente - ad approvare il lavoro in quanto la sua pubblicazione rappresenta un aumento del posizionamento bibliografico del revisore e costituisce quindi un ritorno personale in termini di esposizione. Questo fenomeno potrebbe sfociare in un potenziale abuso nel processo di peer review della conferenza e una conseguente perdita di imparzialità degli articoli pubblicati in essa.

L'obiettivo di questa tesi è la costruzione di CBAT (Committee Bias Analysis Tool): uno strumento software per la raccolta di dati bibliometrici relativi a conferenze scientifiche, alla composizione dei loro comitati di programma e degli articoli pubblicati in esse, utili ad analisi volte alla ricerca di bias sistemico nell'editoria accademica dovuto alla potenziale correlazione tra i riferimenti bibliografici presenti in un articolo scientifico e la probabilità che esso venga pubblicato in una conferenza.

1.2 Nozioni propedeutiche alla lettura

La comprensione della presente tesi necessita della conoscenza di alcune nozioni di base sul contesto della bibliometria e sul funzionamento del processo di editoria accademica, le quali sono descritte di seguito.

1.2.1 Editoria accademica

L'editoria accademica è il settore dell'editoria che si occupa della distribuzione di ricerche accademiche, spesso pubblicate sotto forma di riviste, libri o tesi. La maggior parte delle riviste scientifiche si occupano di una singola disciplina, mentre altre sono interdisciplinari e pubblicano lavori di diversi ambiti. Il processo di pubblicazione è diviso in due fasi distinte: la revisione paritaria e la produzione. Il contenuto dell'articolo, corredato da immagini e figure, viene sottoposto a una o più fasi di revisione paritaria (peer review), che garantiscono un livello di qualità adeguato, in quanto un documento viene considerato valido nel momento in cui uno o più revisori considerano il suo contenuto adatto alla pubblicazione. Questo meccanismo consente inoltre una sicurezza contro il plagio, in quanto i revisori hanno spesso familiarità con le sorgenti consultate dall'autore. Il processo di produzione, controllato dalla casa editrice, sottopone poi l'articolo a un controllo tipografico e lo include in una rivista, pubblicandolo.

1.2.2 Conferenza accademica

Una conferenza accademica (o congresso di ricerca, simposio) è un evento al quale i ricercatori partecipano per presentare i loro risultati e conoscere le ultime attività svolte nel loro campo. Questi eventi sono solitamente organizzati da associazioni o gruppi di ricercatori indipendenti sotto il controllo di un comitato scientifico o tecnico che garantisce la qualità tecnica della ricerca presentata. L'incontro è annunciato tramite una Call For Papers (CFP) o una Call For Abstracts, la quale descrive il tema generale dell'edizione della conferenza ed elenca gli argomenti e le formalità della riunione, quali il tipo di abstract o documento che deve essere presentato, a chi e entro quale scadenza. Le Call For Paper sono anche spesso elencate su siti di annunci di conferenze e sui siti web delle stesse. Molte conferenze collezionano una serie di articoli accademici presentati in esse sotto gli "atti del convegno" (Conference proceedings).

1.2.3 Comitato di organizzazione

Le conferenze necessitano di un comitato di organizzazione composto da un team di individui con specifici ruoli e responsabilità. In eventi di grandi dimensioni si ritrovano tipicamente i seguenti ruoli:

- Steering committee: è responsabile dell'organizzazione generale, della pianificazione finanziaria e della nomina del presidente generale

- Presidente generale (General Chair): prende tutte le decisioni finali riguardanti la conferenza, inclusi ruoli del personale e luogo
- Presidente di programma (Program Chair): sviluppa la Call For Paper e prende responsabilità del processo di revisione paritaria
- Comitato di programma (Program Committee): è nominato dal presidente di programma per occuparsi di parti distinte della peer review
- Altri presidenti e comitati: si occupano di parti diverse dell' organizzazione come finanza, web o pubblicità.

1.2.4 Citazioni

Una citazione è un'espressione alfanumerica inclusa nel corpo di un testo che denota una voce nella sezione di riferimenti bibliografici, e che fa riferimento a una sorgente pubblicata o non pubblicata. Tipicamente una citazione può includere il nome dell'autore, data, casa editrice, titolo della rivista e il Digital Object Identifier (DOI). Esse hanno diversi scopi: riconoscere la pertinenza delle opere altrui rispetto all'argomento della discussione, utilizzare il lavoro di altri senza creare plagio, consentire al lettore di determinare in modo indipendente se il materiale di riferimento supporta l'argomento dell'autore nel modo sostenuto, e aiutare il lettore a valutare la robustezza e la validità del materiale utilizzato dall'autore.

1.2.5 Indice di Hirsch

L'indice H, (o indice di Hirsch, H-index), è un indicatore bibliometrico usato per quantificare la prolificità e l'impatto scientifico di un autore. Esso è definito come il massimo valore di h tale che un dato autore o giornale ha pubblicato h articoli che sono stati citati almeno h volte. L'indice è progettato per migliorare misure più semplici come il numero totale di citazioni o pubblicazioni.

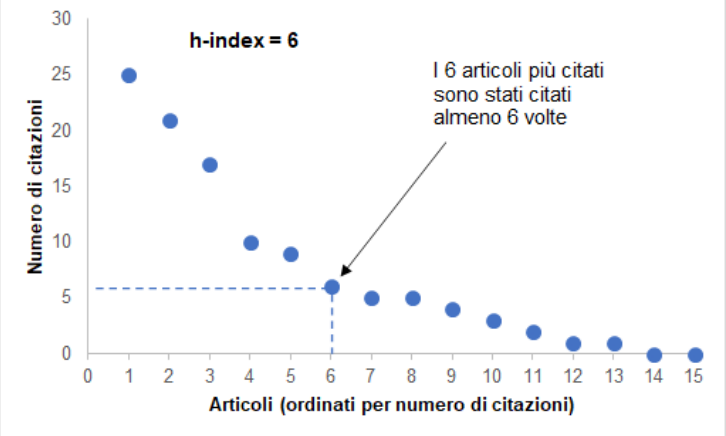

Figura 1.1: Indice di Hirsch

Capitolo 2

Architettura del progetto

2.1 Panoramica del progetto

L'obiettivo del software è il collezionamento di dati bibliometrici al fine dell'analisi statistica della comunità scientifica. Le informazioni richieste per la costruzione di queste metriche riguardano gli autori presenti nel comitato di programma dell'evento e le citazioni contenute negli articoli presentati in esso: per ottenere ciò il software esegue varie estrazioni di dati da sorgenti disparate e le immagazzina in un database. Data una conferenza accademica in input, esso ne esegue la ricerca del nome sul web e ne estrae le varie edizioni. Per ogni edizione esso ne ricava poi la Call For Papers (ricercandola anche sul sito web della

conferenza) per poterne estrarre il comitato di programma ed ottenere le informazioni degli autori appartenenti ad esso. Vengono inoltre ottenuti gli articoli presentati all'evento tramite servizi API (Application Programming Interface) e da essi vengono estratti i riferimenti a lavori di autori esterni.

2.2 Strumenti e tecnologie

Il linguaggio di programmazione scelto per lo sviluppo del progetto è Python 3 (versione 3.7). La preferenza deriva dal fatto che, essendo uno dei linguaggi leader per il settore Data Science, facilita la comprensione del programma e l'integrazione di esso da parte di esperti del settore, non necessariamente software developers. Esso vanta inoltre di una community estremamente attiva, risultando nella disponibilità di un elevato numero di componenti e librerie esterne. In aggiunta dispone di estrema velocità e flessibilità di programmazione, oltre ad offrire molte funzionalità per l'analisi dei dati.

I dati estratti dal software sono salvati in una base di dati non relazionale gestita da MongoDB. La scelta dell'uso di un database NoSQL proviene dall'intrinseca gerarchia delle informazioni rappresentate al suo interno e risulta in una minore complessità dell'architettura dati rispetto a una corrispettiva rappresentazione relazionale

Nel corso del progetto vengono utilizzati i seguenti servizi:

- WikiCFP [2]: una wiki semantica per Call For Papers nei campi di scienza e tecnologia, contenente oltre 50.000 CFP

- Scopus [3]: è il servizio leader mondiale per l'accesso programmatico a riviste, conferenze, articoli, citazioni, metriche e metadati. È un prodotto del gruppo di pubblicazione Elsevier disponibile attraverso sottoscrizione.

- Classifica di conferenze GII-GRIN-SCIE [4]: classifica dello Stage 1 della valutazione delle maggiori conferenze del settore Computer Science degli ultimi anni effettuata dal GGS e messa a disposizione dal Consorzio CINI

Tra le principali librerie Python usate nello sviluppo si trovano:

- mongoengine [5]: DOM (Document-Object Mapper) per usufruire con maggiore semplicità di MongoDB da Python

- pybliometrics [6]: wrapper API usato per estrarre dati dal database di Scopus quali informazioni su autori, pubblicazioni e relative citazioni. Questa libreria offre un mapping ad oggetti del risultato delle API JSON ed offre funzioni di caching che permettono la riduzione del numero di chiamate a Scopus

- beautifulsoup4 [7]: mette a disposizione funzionalità di navigazione e manipolazione di documenti HTML. È usato per il crawling del sito di WikiCFP e dei siti web delle conferenze

- fuzzywuzzy [8]: libreria che permette di determinare quanto due stringhe siano simili, si basa sulla distanza di Levenshtein, una misura di differenza tra stringhe

Per l'estrazione dei dati da testo libero si utilizza Natural Language Processing (NLP), un sottocampo dell'intelligenza artificiale che si concentra sul consentire ai computer di comprendere ed elaborare il linguaggio umano. Esso utilizza diverse fasi di machine learning concatenate in una pipeline che eseguono i seguenti passi:

- analisi lessicale (tokenizer): scomposizione di un'espressione in token
- analisi grammaticale (tagger): associazione delle parti del discorso a ciascuna parola nel testo
- analisi sintattica (parser): arrangiamento dei token in una struttura sintattica
- analisi semantica (ner, textcat): assegnazione di un significato alla struttura sintattica

In particolare si usa Named Entity Recognition (NER) per trovare e classificare entità dal testo in categorie predefinite come nomi, organizzazioni e località. A questo scopo si utilizzano due librerie distinte, in quanto offrono funzionalità uniche:

- SpaCy [9]: libreria per l'elaborazione avanzata del linguaggio naturale in Python. È fornito di modelli statistici e vettori di parole

pre-trained e supporta la tokenizzazione per oltre 50 lingue. È dotato di modelli di rete neurale convoluzionali per tagging, analisi e riconoscimento di entità

- probablepeople [10]: libreria Python per l'analisi di stringhe non strutturate di nomi o società in componenti, utilizzando metodi NLP avanzati. Usando un modello probabilistico, fa ipotesi plausibili sull'identificazione di componenti di nome o aziende, anche in casi in cui i parser basati su regole in genere falliscono.

L'analisi statistica dei dati estratti dal progetto richiede principalmente la libreria SciPy, della quale si utilizzano i moduli matplotlib (libreria di plotting 2D) e NumPy.

2.3 Panoramica del progetto

2.3.1 Panoramica del codice

Il progetto è rappresentato concettualmente da una pipeline globale composta da due ulteriori pipeline che confluiscono in un unico flusso, come mostrato dalla rete di Petri in figura 2.1.

Il flusso ha inizio con la somministrazione al software di una lista di nomi di conferenze, attualmente estratte da un file in formato xlsx offerto dal consorzio CINI, riportante la classifica del 2017 dei maggiori eventi internazionali della disciplina Computer Science. Per ogni conferenza ne vengono estratte le edizioni disponibili tramite ricerca in

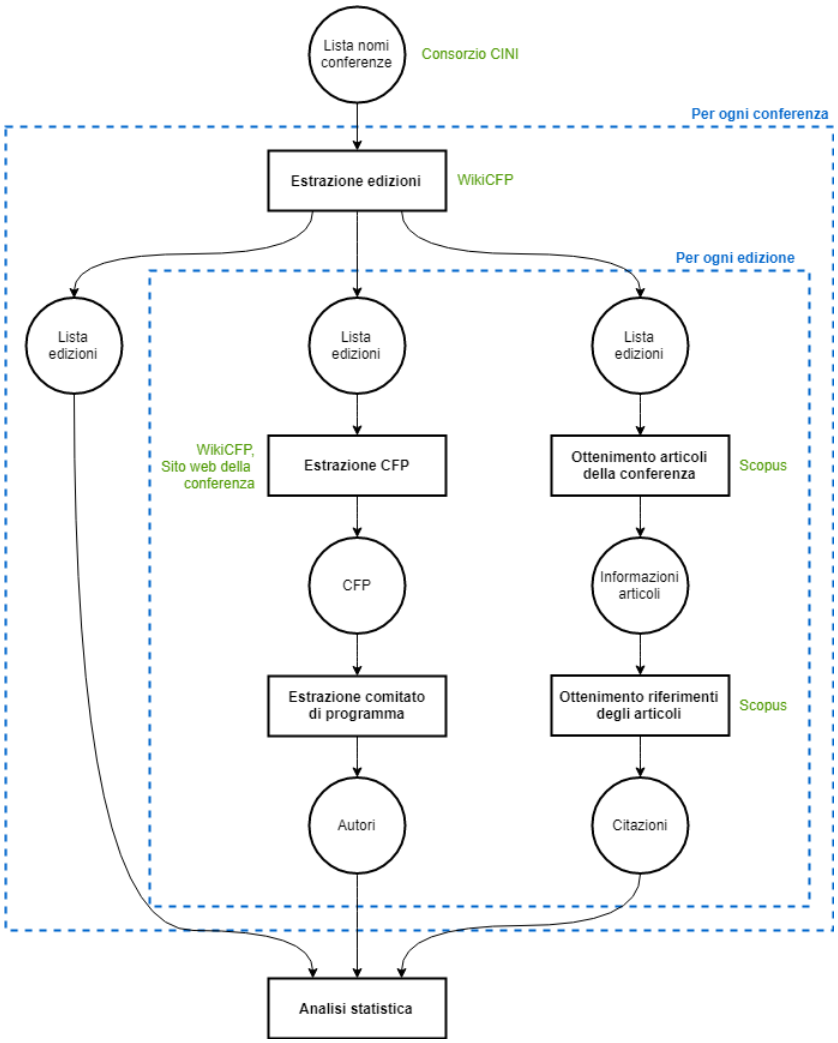

Figura 2.1: Rete di Petri rappresentante la struttura logica della raccolta dati

testo strutturato sul sito WikiCFP. Siccome questo servizio non offre la disponibilità di interfacce API e si deve di conseguenza ricorrere al crawling delle pagine, il nome dell'evento nei risultati di ricerca può non risultare esattamente uguale alla stringa ricercata (come nell'esempio mostrato in figura 2.2). Per questo motivo si preferisce usare come input di ricerca l'acronimo della conferenza.

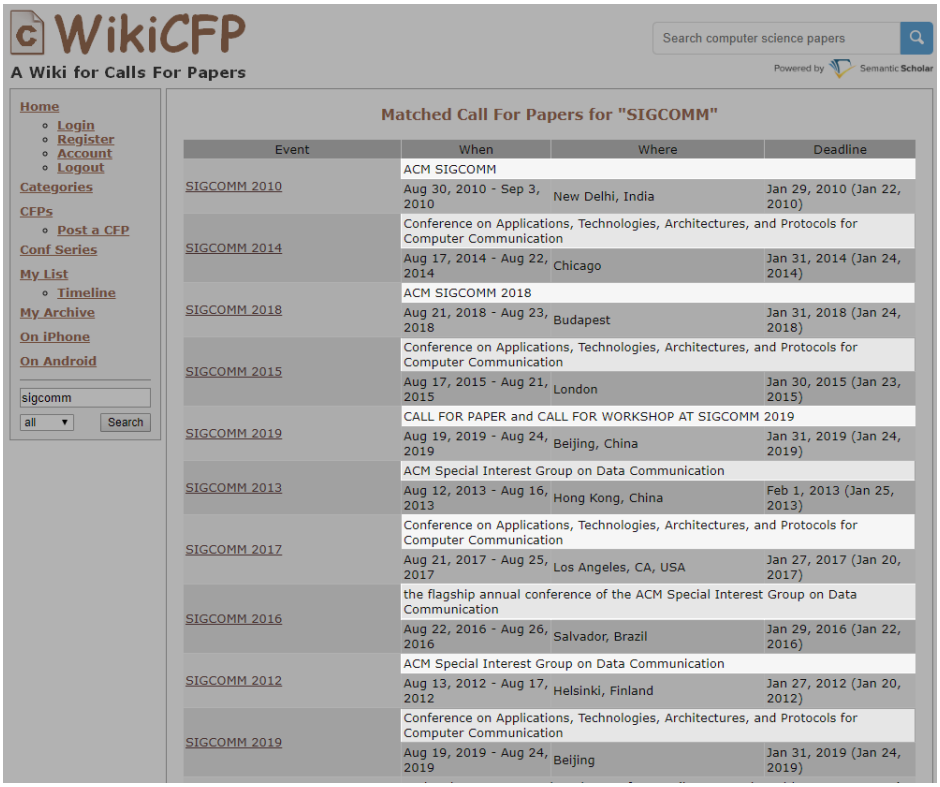

Figura 2.2: Inconsistenza dei risultati di ricerca di WikiCFP

Per ogni edizione partono quindi due flussi d'esecuzione distinti. Il primo inizia con la ricerca della Call For Paper sul sito WikiCFP. Essa, oltre alle informazioni sull'evento, riporta solitamente la lista dei vari comitati della conferenza. Ai fini dell'analisi statistica dei dati è d'interesse solo il comitato di programma in quanto solo questo comitato prende parte nel processo di revisione paritaria degli articoli presentati all'evento. In primo luogo essa viene ricercata sul sito WikiCFP e, nel caso essa non riportasse la lista del comitato di programma, viene ricercata sul sito web della conferenza. Una volta estratto il comitato di programma gli autori contenuti in esso vengono ricercati su Scopus

e salvati in base di dati persistente. L'altro lato della pipeline generale si occupa di ottenere informazioni sugli articoli pubblicati dalla conferenza, interfacciandosi con il database Scopus tramite la libreria pybliometrics. Vengono richiesti gli identificativi delle pubblicazioni dell'evento e, tramite di essi, si ottengono le citazioni in ogni articolo. Si ispezionano quindi gli autori dei lavori citati e viene salvato un riferimento ad essi su base di dati persistente, differenziando coloro che sono anche membri del comitato di programma da coloro che non lo sono, informazione utile alla parte di analisi statistica. Se in qualsiasi punto di questo processo l'estrazione di un elemento (CFP, comitato di programma, articoli e relativi riferimenti) fallisce, la conferenza viene scartata in quanto sono necessarie tutte queste informazioni per l'analisi dei dati.

2.3.2 Architettura statica del codice

Il codice sorgente è diviso in sei moduli distinti, i quali si interessano di ambiti specifici. La pipeline principale della raccolta dati è rappresentata dal metodo `_add_conference`, contenuto in `conference_manager.py`, da cui sono chiamati gli altri moduli. Nella tabella 2.1 vengono mostrate le funzionalità di ogni modulo del software.

Modulo	Funzione
`conference_manager.py`	Ricerca delle conferenze, pipeline principale di raccolta dati
`author_manager.py`	Ricerca degli autori tramite Scopus
`cfp_manager.py`	Ricerca della CFP su WikiCFP o su siti web esterni, estrazione della CFP in testo libero
`committee_manager.py`	Ricerca delle sezioni di comitato di programma, estrazione di nomi e affiliazioni dei membri del comitato di programma tramite Named-Entity Recognition
`paper_manager.py`	Ricerca delle pubblicazioni della conferenza ed estrazione delle citazioni da esse
`stats_manager.py`	Analisi di base dei dati, realizzazione dei grafici

Tabella 2.1: Funzionalità dei moduli presenti nel software

2.3.3 Models

Il software tiene traccia di tre modelli, che vengono salvati in diverse collezioni su base di dati persistente da mongoengine.

Conferenza (Conference)

Rappresenta una singola edizione di una conferenza. Viene salvato nome, acronimo, luogo, anno, aree di interesse, una lista di riferimenti agli articoli pubblicati in esso e una lista contenente i riferimenti ai membri del comitato di programma. Viene inoltre tenuto traccia dell'id e URL della Call For Paper sul sito WikiCFP.


```
_id: ObjectId("5d3db754eba7d742707056ca")
fullname: "ACM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTO..."
name: "ACM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTO..."
year: 2017
location: " Los Angeles, CA, USA"
acronym: "SIGCOMM"
› program_committee: Array
› papers: Array
wikicfp_id: "58424"
› subject_areas: Array
processing_status: "complete"
wikicfp_url: "http://www.wikicfp.com/cfp/servlet/event.showcfp?eventid=58424&copyown..."
```


Figura 2.3: Esempio di model di una conferenza

Autore (Author)

Rappresenta un qualsiasi autore di cui sia stato trovato un riferimento in un articolo, che sia o meno membro del comitato di programma. Ne viene salvato il nome, secondo nome e cognome dedotti dal software, oltre al nome completo estratto dal testo, affiliazione e nazione. Viene estratta anche una lista di EID (Electronic Identification), che rappresentano un identificativo univoco della persona nel database di Scopus. Siccome, come spiegato in dettaglio nei capitoli successivi, la ricerca di un autore su questo servizio può ritornare diversi risultati rappresentanti semanticamente la stessa persona, un oggetto di questo model può risultare associato a più EID. Inoltre, dal momento che il nome dell'autore ricercato viene estratto da testo libero e può quindi essere formattato in svariati modi, si verificano casi in cui la stringa di testo non viene riconosciuta come un nome proprio da parte della libreria di NLP. In questi casi, come piano di fallback, queste informazioni vengono ottenute tramite regole programmate e viene salvata l'indicazione che la persona non è stata estratta in modo esatto, in modo da poterla potenzialmente escludere dall'analisi statistica. Questa tematica viene approfondita nei prossimi capitoli.

Pubblicazione (Paper)

Sono salvati l'identificativo Scopus dell'oggetto e i riferimenti agli autori citati nell'articolo. Questi ultimi vengono separati in due liste differenti

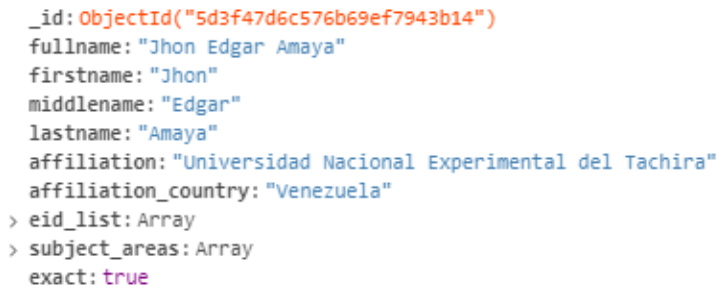

Figura 2.4: Esempio di model di un autore

per distinguere il fatto che la citazione è relativa ad un membro del comitato di programma o meno.

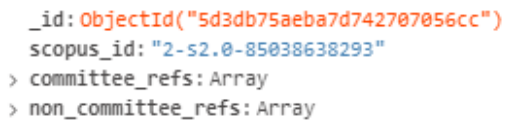

Figura 2.5: Esempio di model di una pubblicazione

2.4 Installazione e configurazione

Il software è open source e viene messo a disposizione della comunità grazie alla piattaforma Github. Per eseguire il software è necessario il download del codice sorgente tramite la clonazione della repository:

```
git clone https://github.com/fabiosangregorio/cbat.git
```

e la successiva installazione dei requisiti elencati nel file requirements.txt, tramite il comando:

```
pip install -r requirements.txt
```

L'accesso a Scopus necessita di una chiave API, reperibile all'indirizzo `https://dev.elsevier.com/apikey/manage`, che deve essere inse-

rita nella prima configurazione di pybliometrics seguendo la procedura indicata:

```
python
>> import pybliometrics
>> pybliometrics.utils.create_config()
```

Si nota che Scopus richiede che l'accesso avvenga tramite una rete presentante una sottoscrizione al servizio, per cui potrebbe essere necessario l'accesso tramite una VPN autorizzata per utilizzare il software. L'esecuzione del software avviene con l'avvio dello script `cbat/main.py`

```
cd cbat
python -m main.py
```

Attualmente i nomi delle conferenze da processare vengono ottenuti dal file `cbat/data/cini.xlsx`. Altre conferenze possono essere sottoposte al processo usando come input del metodo `_add_conference` una lista di oggetti Conference contenenti nome e acronimo dell'evento. Il risultato dello script è una base di dati popolata con i dati estratti relativi alle conferenze prese in input.

Il file `config.py` contiene una serie di variabili globali di configurazione che vengono utilizzate in tutto il software, e che possono essere regolate per controllare l'accuratezza della raccolta dati. Di seguito le variabili configurabili:

Nei capitoli successivi vengono studiate in dettaglio le dinamiche delle componenti principali della pipeline di data collection.

Variabile	Valore di default	Funzione
HEADINGS	["committee", "commission"]	Parole chiave usate per riconoscere l'inizio e fine delle sezioni dei comitati della conferenza
PROGRAM_HEADINGS	["program", "programme", "review"]	Parole chiave usate nel riconoscimento delle sezioni di testo libero comprendenti il comitato di programma
CONF_EDITIONS_LOWER_BOUNDARY	5	Numero di anni precedenti all'attuale per i quali cercare le edizioni delle conferenze
CONF_EXCLUDE_CURR_YEAR	True	Indica se escludere l'anno attuale nella ricerca delle edizioni delle conferenze
AUTH_NO_AFFILIATION_RATIO	0.5	Dopo l'estrazione del comitato di programma, se il rapporto tra gli autori per cui non è stato possibile estrarre l'affiliazione e gli autori totali estratti è maggiore di questa soglia la conferenza viene scartata
AUTH_NOT_EXACT_RATIO	0.5	Durante l'estrazione del comitato di programma, se il rapporto tra le persone che non sono state riconosciute come tali da NLP e le persone totali estratte è maggiore di questa soglia, si può inferire che probabilmente la sezione non contiene solo nomi e affiliazioni di autori ma anche altro testo, per cui nel risultato dell'estrazione si tiene conto solo delle persone estratte esattamente.
MIN_COMMITTEE_SIZE	5	Se l'estrazione del comitato di programma ritorna un numero di autori minore di questa soglia probabilmente l'estrazione non è avvenuta con successo, perciò la conferenza viene scartata
NER_LOSS_THRESHOLD	0.7	Soglia oltre la quale si inferisce che il NER ha perso una quantità significativa di dati nell'estrazione del comitato di programma (spiegato in dettaglio nei prossimi capitoli)
FUZZ_THRESHOLD	70	Soglia oltre la quale si considera sufficiente l'accuratezza dell'estrazione dell'affiliazione dell'autore
SPACY_MODEL	'en_core_web_sm'	Modello di rete neurale trained che deve essere usato da SpaCy nel NER
DB_NAME	'cbat'	Nome del database MongoDB a cui connettersi. Nel caso non esistesse (come alla prima esecuzione), esso verrà creato automaticamente
WIKICFP_BASE_URL	'http://www.wikicfp.com'	URL base del sito di WikiCFP

Tabella 2.2: Variabili globali di configurazione

Capitolo 3

Premessa: accuratezza di Named-Entity Recognition

Il software non usa direttamente Named-Entity Recognition per ottenere le informazioni degli autori da testo libero: un test effettuato su dati reali in precedenza allo sviluppo ha mostrato che l'accuratezza della tecnologia sull'estrazione di autori non è abbastanza alta da poter essere utilizzata autonomamente.

Il test prevedeva la somministrazione al NER di alcune Call For Paper con l'obiettivo di misurare l'accuratezza di estrazione di nomi propri presenti nel testo, utilizzando tre diversi modelli di reti neurali pre-trained.

È stato tenuto traccia delle entità taggate come persone per poi compararle con gli autori effettivamente presenti nel testo, estratti manualmente. In ogni prova sono state misurate le seguenti metriche in modo da calcolare la precisione dell'estrazione:

- corretti: numero di nomi estratti dal software presenti anche nel testo
- errati: numero di nomi estratti dal software, i quali però non sono presenti nel testo
- mancanti: numero di nomi non estratti dal software che sono però presenti nel testo

La performance del modello è stata calcolata con la formula

$$accuratezza = \frac{corretti - errati}{mancanti}.$$

In questo modo un alto tasso di nomi estratti ma non presenti nel testo porta a una penalizzazione dell'accuratezza, in quanto essi sono dati errati che continuerebbero nel processo di pipeline, portando a collezionamento errato di dati o a errori del software.

Nel caso migliore, su 62 persone presenti nella CFP, sono stati registrati 52 nomi estratti correttamente, 11 estratti ma non presenti nel testo, e 10 che il NER non ha riconosciuto come nomi, portando a un'accuratezza del 66.13%, mentre nel caso peggiore su 22 persone sono stati riportati 16 corretti, 13 errati e 6 mancanti, risultando in un'ac-

curatezza del 13.63%. Questi risultati sono verosimilmente dovuti al fatto che i modelli di SpaCy utilizzati sono trained su testi in lingua inglese, mentre i nomi contenuti nelle Call For Papers sono spesso di origini orientali, come Cina, Giappone o India e di conseguenza c'è alta probabilità che essi non vengano riconosciuti come tali.

I test sono stati effettuati con tre diversi modelli:

- `en_core_web_sm` (10 MB): rete in lingua inglese trained su OntoNotes (blog, news, commenti)
- `en_core_web_md` (91 MB): rete in lingua inglese trained su OntoNotes e CommonCrawl (blog, news, commenti)
- `xx_ent_wiki_sm` (3 MB): rete multilingua trained su Wikipedia, supporta l'identificazione di entità "persona" in inglese, tedesco, spagnolo, francese, italiano, portoghese e russo

Nonostante non abbia il peso minore `en_core_web_sm` ha avuto la performance peggiore nei test: questo è dovuto alla natura multilingua di `xx_ent_wiki_sm` che lo rende più adatto al contesto di utilizzo. In termini di tempo d'esecuzione, `en_core_web_sm` ha impiegato circa 2 secondi per il processamento di una CFP di media dimensione con un'esecuzione a singolo thread, `xx_ent_wiki_sm` attorno ai 6 secondi, mentre `en_core_web_md` ha impiegato più di 20 secondi. Presi in considerazione questi fattori, per lo sviluppo del progetto è stato scelto il modello `en_core_web_sm` (comunque configurabile nella costante `SPACY_MODEL`). Per sopperire alla bassa accuratezza è stato progettato un algoritmo

che fa un uso indiretto della i questa tecnologia per l'estrazione delle persone dal testo, descritto dettagliatamente nei prossimi capitoli.

Capitolo 4

Ottenimento del comitato di programma

Nell'architettura logica mostrata precedentemente l'ottenimento della Call For Paper e l'estrazione del comitato di programma sono mostrati come due componenti separati. Nell'implementazione, mostrata in figura 4.1, essi risultano invece inevitabilmente interlacciati, in quanto non tutte le informazioni necessarie sugli autori sono presenti in un unico luogo.

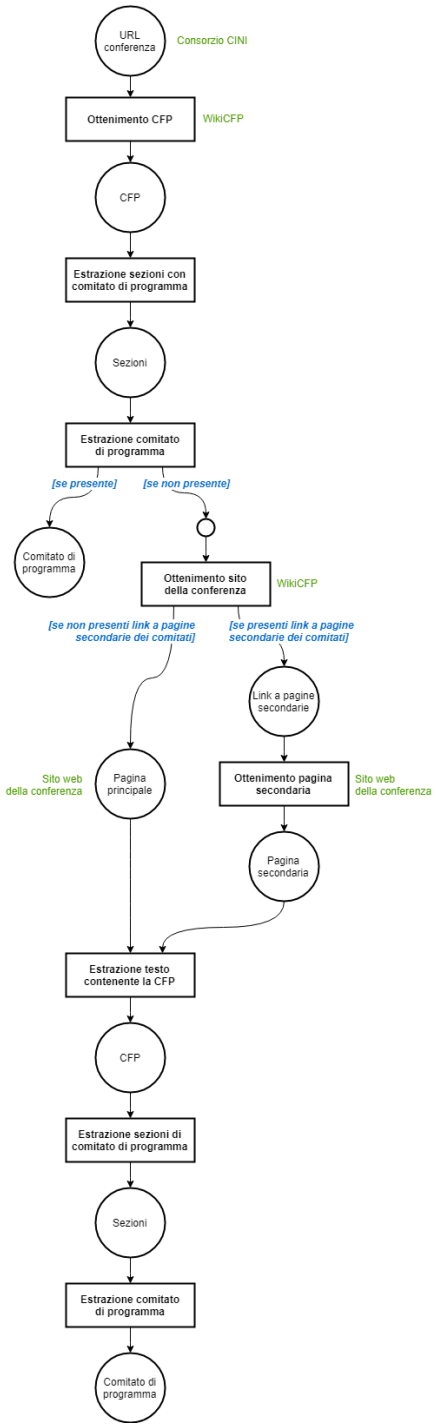

Figura 4.1: Algoritmo di ricerca del comitato di programma

Dall'URL di WikiCFP della conferenza (ottenuto in precedenza) viene scaricata la Call For Paper, estratte le sezioni di testo contenenti il comitato di programma e ottenuti gli autori contenuti in esse. Nella maggior parte dei casi però questa CFP riporta solo le informazioni dell'evento e non la composizione dei comitati, per cui può essere necessario dover analizzare il sito web della conferenza per ottenere queste informazioni (indirizzo che solitamente si trova sulla pagina di WikiCFP dell'evento).

Figura 4.2: Esempio di comitato di programma presente su WikiCFP

Nel sito web si ricerca inizialmente la presenza di link interni a pagine riguardanti il comitato di programma o, nel caso questi non fossero presenti, link a pagine concernenti tutti i comitati dell'evento. In caso esso esista si naviga a questa pagina, altrimenti si procede utilizzando

la pagina principale. Una volta ottenuto il documento HTML target avviene l'estrazione del corpo, il quale verosimilmente presenta il testo contenente il comitato di programma. Da esso vengono infine estratte le sezioni del comitato e con esse il comitato stesso. Se un qualsiasi componente di questo processo fallisce, come nel caso in cui il comitato di programma non fosse presente o il link al sito della conferenza non fosse mostrato su WikiCFP, la procedura viene interrotta e l'evento non viene preso in considerazione.

Come si può intuire, i siti web delle conferenze sono molto differenti l'un l'altro, le informazioni sono mostrate in formati disparati e non è possibile prevedere il loro contenuto; questo aggiunge un livello di complessità al processo che porta l'estrazione dati a essere non deterministica e di conseguenza a non andare sempre a buon fine.

Se non si è certi della correttezza dei dati estratti in una conferenza (ossia se è possibile che essi non rappresentino nomi di autori, ma altro testo) essa viene scartata, in quanto nel contesto dell'analisi dei dati l'accuratezza delle informazioni ottenute è di gran lunga più importante della loro quantità. Di conseguenza il software non è in grado di processare con successo la totalità degli eventi dati: su 100 edizioni date in input esso è in grado di ottenere il comitato di programma di 56 conferenze. Secondo test effettuati in conseguenza allo sviluppo, i dati di questo 56% presentano un'accuratezza del 95%.

Di seguito vengono analizzate in dettaglio le componenti principali dell'estrazione.

4.1 Estrazione della Call For Paper dal sito web della conferenza

Una volta ottenuta la pagina del sito web dell'evento avviene la ricerca del corpo del testo contenente la Call For Paper della conferenza o, idealmente, la composizione del comitato di programma, in modo da evitare l'estrazione di parti di pagina non inerenti allo scopo del software (quali eventuali sidebar, testate, elementi grafici, ecc.)

Prendiamo come esempio il documento (semplificato) seguente:

```
<div> Header </div>
<table>
  <tr>
    <td>Senior <h1>program committee</h1></td>
    <td>
      Matt Scott, MIT <br>
      Xin Yung, Harvard University <br>
    </td>
  </tr>
  <tr>
    <td>Steering committee</td>
    <td>
      Michael Mitzenmacher, Harvard University <br>
      Justine Sherry, Carnegie Mellon University
    <td>
  </tr>
  <tr>
    <td>Program committee</td>
    <td>
      Theo Benson, Duke University <br>
      Mark Crovella, Boston University <br>
      Nate Foster, Cornell University
    </td>
  </tr>
</table>
<div> Footer </div>
```

Listing 1: Esempio 1: documento di Call For Paper

In questo caso le sezioni di testo che si vogliono estrarre sono soltanto i due comitati di programma: lo steering committee non è di nostro interesse in quanto i suoi componenti non valutano gli articoli presentati per l'evento, e gli elementi header e footer non aggiungono informazioni utili all'ottenimento degli autori. Di conseguenza il

risultato del processo su questo esempio deve essere:

```
Senior program committee
Matt Scott, MIT
Xin Yung, Harvard University

Program committee
Theo Benson, Duke University
Mark Crovella, Boston University
Nate Foster, Cornell University
```

Per ottenere questo risultato, il processo ha inizio con la ricerca di tag HTML nella pagina contenti testo riferito al comitato, per mezzo di un'espressione regolare. Più precisamente, grazie alla libreria `beautifulsoup`, si ottiene il testo contenuto in ogni tag del body della pagina e nei suoi nodi figli (in modo ricorsivo), e lo si testa per la presenza di una o più permutazioni delle stringhe di testo configurabili nelle costanti `HEADINGS` e `PROGRAM_HEADINGS`, il cui risultato rappresenta i titoli più comuni delle sezioni riferite al comitato di programma trovati sperimentalmente sui siti web delle conferenze. (es. "Programme committee", "Program commission", "Review committee"...) Nell'esempio vengono estratti i tag `<h1>program committee</h1>` e `<td>Program committee</td>`.

Una volta identificato un tag HTML si vuole ottenere il corpo di testo di cui esso è titolo, il quale verosimilmente contiene gli autori relativi a quella sezione. Per fare ciò si attraversa il DOM (Document Object Model, costituisce l'albero dei tag di un documento HTML)

"all'indietro", ossia partendo da esso e visitando i tag contenitori finché il testo d'interesse viene trovato. Risalendo la gerarchia contenitore per contenitore, ad ogni iterazione vengono inclusi altri tag i quali potenzialmente contengono altre parole. Ad ogni ciclo si verifica quindi che la lunghezza del testo interno al contenitore sia significativamente maggiore di quella del titolo iniziale e, in caso affermativo, si salva quest'ultimo tag in una lista dei risultati e si conclude il ciclo deducendo che l'estrazione della sezione è avvenuta con successo.

Si nota che la lunghezza è arricchita con un numero di caratteri di guardia (pari a 20) per evitare che la risalita del DOM si fermi nel momento in cui si incontrano segni di punteggiatura o piccole etichette in prossimità del titolo. Come si può intuire, l'algoritmo non risulta perfetto in quanto è possibile che il testo nel contenitore designato non rappresenti una lista di autori ma altre informazioni non utili all'estrazione. Questa routine è però solo il primo passo della pipeline dedicata all'ottenimento del comitato; la validità dei dati estratti sarà controllata negli step successivi. Nel caso del primo titolo dell'esempio l'algoritmo naviga inizialmente al `<td>` padre; qui si nota che il testo a questa iterazione è "Senior program committee", la cui lunghezza non è significativamente differente dal titolo originale, per cui si continua la risalita al `<tr>`, che invece presenta un quantitativo di testo maggiore e di conseguenza viene estratto correttamente.

Si consideri invece l'esempio seguente, in cui cambia l'alberatura del documento:

```
<div>
  <h2>Senior program committee</h2>
  <ul>
    <li>Matt Scott, MIT</li>
    <li>Xin Yung, Harvard University</li>
  </ul>
  <h2>Steering committee</h2>
  <ul>
    <li>Michael Mitzenmacher, Harvard University</li>
    <li>Justine Sherry, Carnegie Mellon University</li>
  </ul>
  <h2>Program committee</h2>
  <ul>
    <li>Theo Benson, Duke University</li>
    <li>Mark Crovella, Boston University</li>
    <li>Nate Foster, Cornell University</li>
  </ul>
</div>
```

Listing 2: Esempio 2: documento di Call For Paper

Anche in questo caso si hanno due tag riferiti al comitato di programma e di conseguenza si vogliono estrarre entrambe le sezioni, volendo ottenere idealmente lo stesso risultato dell'esempio precedente. Risalendo l'alberatura per ambo i titoli l'algoritmo termina però allo stesso `<div>` padre, di conseguenza tutto il testo verrebbe estratto due volte invece che una. Per questo motivo, dopo aver trovato i contenitori del testo ricercato essi vengono trasformati in un `Set`, ossia una collezione non ordinata di elementi univoci, eliminando tutti i tag duplicati. La routine ritorna poi l'unione del testo contenuto in questi ultimi tag rimasti. Il valore ritornato dall'algoritmo in questo esempio

risulta quindi:

```
Senior program committee
Matt Scott, MIT
Xin Yung, Harvard University

Steering committee
Michael Mitzenmacher, Harvard University
Justine Sherry, Carnegie Mellon University

Program committee
Theo Benson, Duke University
Mark Crovella, Boston University
Nate Foster, Cornell University
```

Si ricorda che l'obiettivo di questa sezione della pipeline è estrarre la Call For Paper contenente il comitato di programma e non l'ottenimento esclusivo di esso, per cui anche se in questo caso è stato estratto pure lo steering committee l'algoritmo ha avuto successo.

Il codice che si occupa della parte appena descritta è mostrato di seguito:

Listing 3: Algoritmo di estrazione della Call For Papers

```
1 RE_P_PROGRAM_HEADINGS = [f'{ph}.*{h}' for h in HEADINGS for ph in
   ↪  PROGRAM_HEADINGS]
2 regex = re.compile('.*(' + '|'.join(RE_P_PROGRAM_HEADINGS) + ').*',
   ↪  re.IGNORECASE)
3 program_tags = [tag.parent for tag in html.body(text=regex)] #
   ↪  tag.parent gets the actual tag
4
5 parent_tags = []
6 for tag in program_tags:
7     prev_tag = tag
8     while True:
```

```
9           parent_tag = prev_tag.parent
10          if not parent_tag or len(parent_tag.text) >
             ↪  len(prev_tag.text) + 20:
11              parent_tags.append(parent_tag)
12              break
13          prev_tag = parent_tag
14
15  # get the text of unique parent tags
16  cfp_text = "\n".join([s for t in list(set(parent_tags)) for s in
     ↪  t.stripped_strings
17                        if not isinstance(t, NavigableString)])
18
19  if len(cfp_text) < len('\n'.join([t.text for t in program_tags])) +
     ↪  10:
20      cfp_text = html.text
```

4.2 Estrazione delle sezioni contenenti il comitato di programma

Una volta in possesso della Call For Paper ottenuta da WikiCFP o dal sito web della conferenza, si procede all'estrazione delle sole sezioni riferite al comitato di programma escludendo quelle che includono altri componenti dell'evento (ad esempio lo Steering Committee). Per ogni titolo riferito al comitato di programma si denota l'inizio della sezione come la riga di testo successiva al titolo stesso, mentre la fine della sezione viene indicata dal titolo successivo riferito a un qualsiasi comitato (configurabile nella costante `HEADINGS`) o dalla fine della Call For Paper. I testi separati rappresentano verosimilmente la lista degli

autori appartenenti al comitato e vengono quindi passati al prossimo componente della pipeline sotto forma di un vettore.

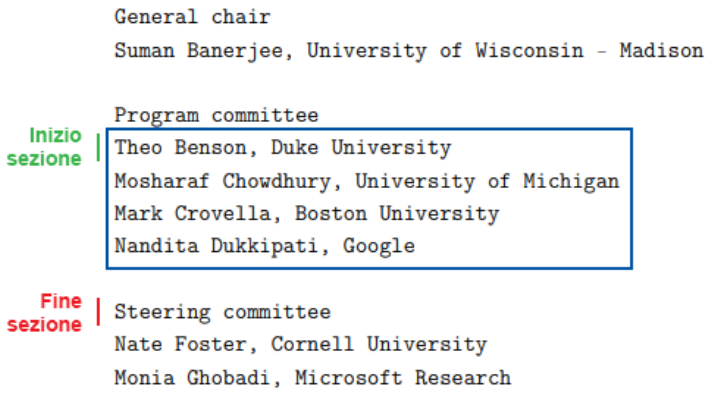

Figura 4.3: Denotazione di inizio e fine sezione

4.3 Estrazione del comitato di programma

Le singole sezioni di testo vengono messe come input alla parte principale della pipeline: l'algoritmo di estrazione del comitato di programma. Esso, tramite diverse iterazioni di Named-Entity Recognition e regole programmate, si occupa del riconoscimento e dell'estrazione dei nomi degli autori presenti nella sezione di testo libero data, assieme al parsing dell'eventuale affiliazione associata alla persona.

Si è osservato empiricamente che tutte le Call For Papers presentano i membri dei comitati come una lista contenente informazioni ricorrenti

e formattate in modi simili. Il nome e cognome dell'autore sono presenti in ogni CFP, l'affiliazione dell'autore è sempre indicata a meno di rari casi, mentre la nazione riferita all'affiliazione è mancante per circa un quarto delle CFP studiate. Di seguito alcuni esempi dei formati più comuni:

```
Nome Cognome, Affiliazione, Nazione

Cognome Nome
Affiliazione, Nazione

Nome Cognome
Affiliazione
Nazione
```

Listing 4: Formati di autori più comuni

Come si può notare, l'informazione di un singolo autore può essere trovata su una o più righe di testo consecutive: per questo motivo l'algoritmo, invece di usufruire di Named-Entity Recognition per estrarre direttamente le informazioni, lo utilizza per calcolare ogni quante righe di testo viene trovato un nome proprio, per poi estrarlo secondo regole programmate.

In generale la routine è composta da tre parti, come mostrato nella rete di Petri in figura 4.4:

1. Calcolo dei parametri con cui estrarre le righe di testo
2. Estrazione degli elementi del nome proprio e parsing di affiliazione e nazione dell'autore

3. Rimozione degli autori duplicati

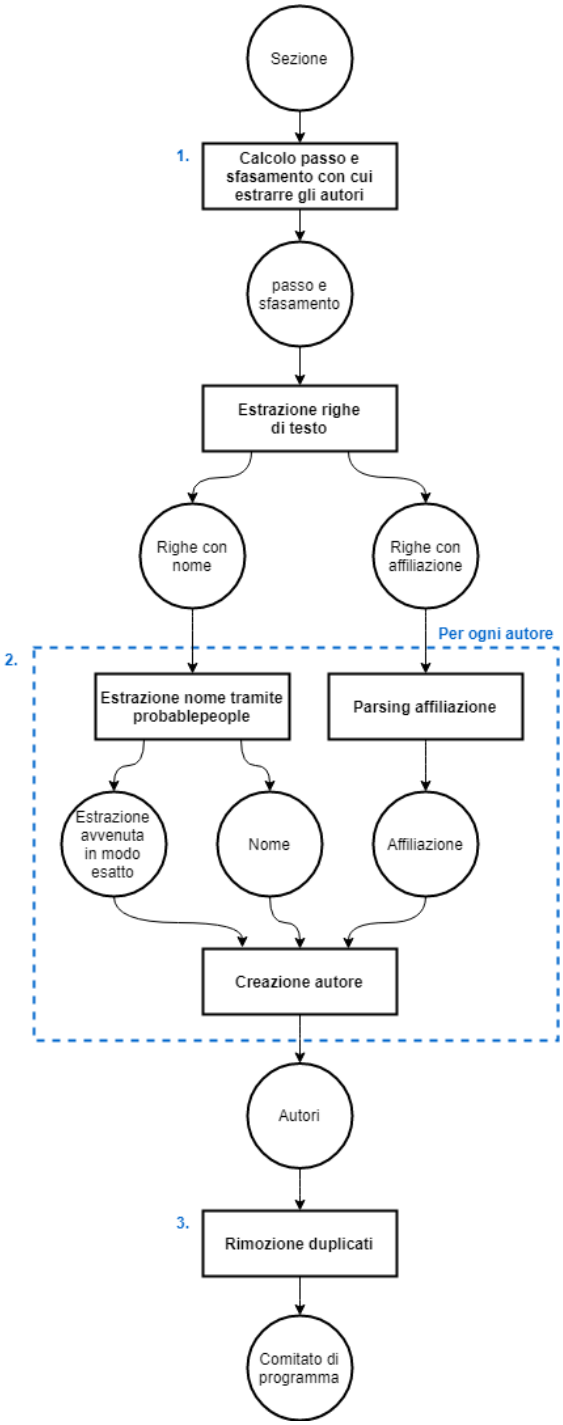

Figura 4.4: Rete di Petri dell'estrazione del comitato di programma

4.3.1 Calcolo del passo

La prima parte dell'algoritmo si occupa del calcolo del passo con cui considerare le righe di testo per estrarre gli autori, e dello sfasamento iniziale. Il passo rappresenta il numero di righe di testo che intercorrono tra un nome proprio di un autore e l'altro. La riga corrispondente a questo step viene interpretata come nome e cognome dell'autore, tutte le altre (se lo step è maggiore di 1) vengono considerate come affiliazione e nazione, in quanto si è notato che la totalità delle Call For Paper rispetta questo formato. Si prevede un passo massimo di quattro righe, dopo il quale si inferisce che la sezione non contenga solo una lista di autori ma anche altro testo, e viene di conseguenza esclusa. Lo sfasamento è invece preso in considerazione in quanto si è verificato che tra il titolo che denota la sezione e l'inizio della lista di autori in alcuni casi è presente del testo introduttivo che, se incluso nel conteggio delle righe, può portare al mancato riconoscimento degli autori e quindi al fallimento dell'algoritmo.

In particolare, questi parametri sono calcolati secondo il diagramma di flusso analizzato di seguito, mostrato anche in figura 4.5.

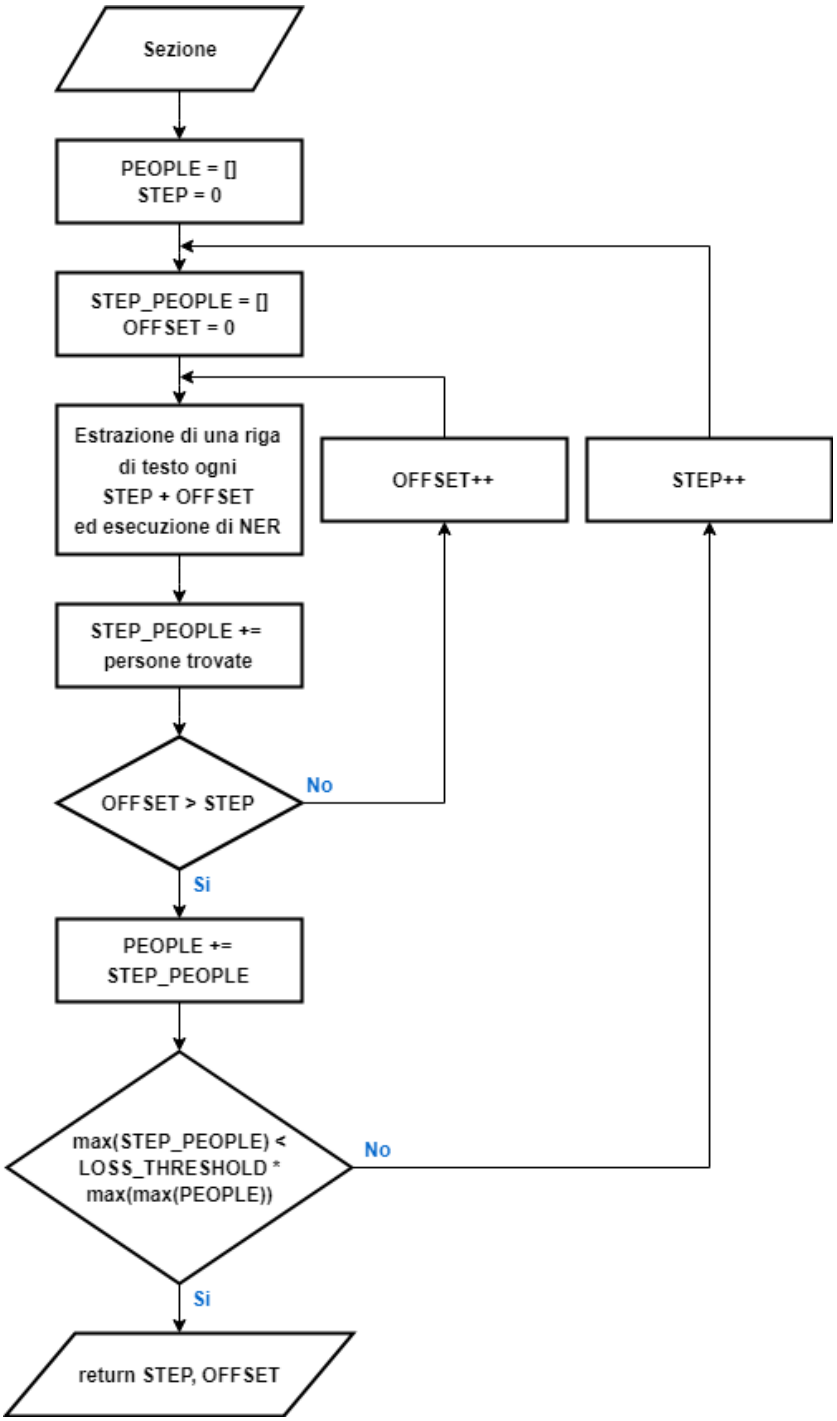

Figura 4.5: Diagramma di flusso della ricerca di step e offset da usare per l'estrazione

L'algoritmo è composto da due cicli annidati: il ciclo esterno incrementa il passo ad ogni iterazione, mentre quello interno incrementa lo sfasamento iniziale. Per ognuno di questi parametri vengono estratte le righe di testo che rispettano i criteri, ossia quelle di indice `i` tale che `i % (step + 1) == offset` (`step + 1` in quanto il passo parte da 0 e non è possibile la divisione per 0). Su questo documento appena formato viene effettuato Named-Entity Recognition grazie alla libreria SpaCy e ne viene analizzato il risultato. Per ogni step, ad ogni iterazione di offset viene salvato in un vettore il numero di occorrenze delle entità che sono state etichettate come persona, in cui ogni indice dell'array rappresenta uno sfasamento. L'algoritmo si conclude nel momento in cui viene raggiunto un passo maggiore di 4 o se il set di risultati del NER con un dato passo è significativamente ridotto: questo secondo caso è calcolato controllando se il numero massimo di nomi estratti in un'iterazione di step (quindi il massimo dell'array di sfasamento) è minore del 70% del numero di persone massimo estratto finora (soglia configurabile nella costante `NER_LOSS_THRESHOLD`). In questo caso si può inferire che, siccome al passo precedente sono state estratte molte più persone che a quello attuale, probabilmente lo step precedente contiene tutte le persone ed è di conseguenza quello corretto. Infine viene scelto lo sfasamento da utilizzare, calcolato come il massimo dell'array di sfasamento del passo scelto.

Prendiamo come esempio la Call For Paper della conferenza CCS 2018, di cui viene mostrato un estratto:

```
Alessandro Acquisti,
Carnegie Mellon University
Sadia Afroz,
ICSI
Gail-Joon Ahn,
Arizona State University and SAMSUNG Research
Devdatta Akhawe,
Dropbox
Ehab Al-Shaer,
UNC Charlotte
Sumayah Alrwais,
King Saud University
Dennis Andriesse,
VU Amsterdam
Giuseppe Ateniese,
Stevens Institute of Technology
Erman Ayday,
Case Western Reserve University
Foteini Baldimtsi,
George Mason University
Shehar Bano,
UCL
...
```

In questo testo si trovano nomi propri di persone ogni due righe, quindi ci si aspetta che il passo scelto sia 2. I risultati dell'estrazione del NER sull'intera CFP ai vari passi è la seguente:

```
Step 1: [147]
Step 2: [142, 5]
Step 3: [52, 48, 47]
```

Eseguendo il software si verifica che Named-Entity Recognition identifica con `step = 1` 147 entità come persone. Al passo 2 vengono pro-

cessati due documenti: quello con sfasamento 0 ha il seguente formato:

```
Alessandro Acquisti,
Sadia Afroz,
Gail-Joon Ahn,
Devdatta Akhawe,
Ehab Al-Shaer,
Sumayah Alrwais,
Dennis Andriesse,
Giuseppe Ateniese,
Erman Ayday,
Foteini Baldimtsi,
Shehar Bano,
...
```

mentre quello con offset 1 è il seguente:

```
Carnegie Mellon University
ICSI
Arizona State University and SAMSUNG Research
Dropbox
UNC Charlotte
King Saud University
VU Amsterdam
Stevens Institute of Technology
Case Western Reserve University
George Mason University
UCL
...
```

A questo passo vengono riconosciute 142 persone con nessuno sfasamento, mentre 5 con sfasamento 1 (questo risultato è dato dalla non perfezione della tecnologia NER, in quanto queste cinque sono affiliazioni che sono state identificate erroneamente come persone). Il massimo

di questi risultati è 142, il quale è maggiore del 70% del massimo dei risultati estratti finora (che è 147). Continuando con lo step 3 il NER ritorna circa 50 risultati per ogni offset, in quanto estraendo una riga su tre il documento alterna nomi e affiliazioni, come mostrato di seguito:

```
Alessandro Acquisti,
ICSI
Devdatta Akhawe,
UNC Charlotte
Dennis Andriesse,
Stevens Institute of Technology
Shehar Bano,
...
```

Il massimo è 52, che è minore del 70% di 147, per cui viene scelto il passo 2 in quanto contiene tutti i nomi e lo sfasamento 0 in quanto contiene il numero massimo di nomi estratti. Si noti che in realtà le persone presenti nel testo sono 142: estraendole solo con NER si sarebbero avuti cinque risultati errati che sarebbero stati portati avanti nella pipeline; usando questo processo si ha invece un'estrazione perfetta.

Il codice che si occupa dell'esecuzione di NER per ogni sfasamento è il seguente:

```
for offset in range(0, step + 1):
    n_people = 0
    step_lines = [l for i, l in enumerate(text_lines) if (
                  len(l) >= 4 and i \% (step + 1) == offset)]
    for doc in nlp.pipe(step_lines, n_threads=16, batch_size=10000):
        if len([e.text for e in doc.ents if e.label_ == 'PERSON']):
            n_people += 1
```

Listing 5: Esecuzione di Named-Entity Recognition

Siccome si è verificato che l'elaborazione NER di SpaCy dell'intera Call For Paper può occupare fino a 10 secondi eseguita single-thread su di un processore medio, il documento viene suddiviso in singole righe e ne viene eseguito il tagging delle entità in parallelo su ognuna in modo da sfruttare il multiprocessing della libreria, risultando in un tempo d'esecuzione inferiore al secondo per CFP di medie dimensioni.

4.3.2 Estrazione degli elementi del nome proprio e parsing di affiliazione e nazione dell'autore

Una volta identificata la riga contenente le informazioni dell'autore grazie alla subroutine precedente, ne viene estratto il nome completo tramite l'espressione regolare:

```
re.compile(r"^\W*([\"\w\. '']+)", re.MULTILINE)
```

Il testo rimanente viene interpretato come affiliazione e nazione dell'autore. Vengono quindi estratti dal nome intero i singoli elementi quali nome, cognome, secondi nomi e iniziali, necessario in quanto le

API di Scopus necessitano in input nome e cognome come elementi separati per la ricerca di autori. A questo scopo si utilizza la libreria probablepeople, la quale, grazie a metodi avanzati di natural language processing, è in grado di eseguire il tagging degli elementi del nome passato come input per identificarne i componenti. Essendo NLP un metodo probabilistico si verificano casi in cui probablepeople riconosce il nome come ragione sociale di un'azienda, fallisce nel riconoscimento dei componenti, o lo riconosce come altro testo: in questi casi viene tentata un'estrazione basata sui formati di nome più comuni, descritta di seguito. Siano A, B e C delle stringhe di testo. Nel caso del formato `A, B C` si inferisce che A rappresenti il cognome, mentre le restanti B e C siano nome e secondi nomi. Nel formato `A B C` C viene invece interpretato come cognome, mentre i restanti A e B come nome e secondi nomi. Siccome quest'ultima è un'estrazione basata su regole programmate e sono presenti casi in cui queste non sono rispettate, come ad esempio la presenza di persone con più cognomi, il software annota l'autore come non estratto in modo esatto, in modo da poterlo potenzialmente escludere o prendere minormente in considerazione nell'analisi statistica. In ogni caso, se l'estrazione dei componenti del nome fallisce non risulta possibile ricercare tale autore, per cui esso viene escluso dai risultati della routine.

La funzione che si occupa dell'estrazione del nome è mostrata di seguito:

Listing 6: Algoritmo di estrazione del nome

```
def _extract_person_name(name):
    person = None
    try:
        pp_result = pp.tag(name)
    except Exception:
        pp_result = [None, None]

    if(pp_result[1] == "Person"):
        r = pp_result[0]
        r = {k: i.replace('(', '').replace(')', '') for k, i in
         ↪  r.items()}
        person = Author(
            fullname=name,
            firstname=r.get('GivenName'),
            middlename=(r.get('MiddleName') or
                        r.get('MiddleInitial') or
                        r.get('Nickname')),
            lastname=r.get('LastName') or r.get('Surname'))
    else:
        person = Author(fullname=name)

    if not person.firstname or not person.lastname:
        split_char = ',' if ',' in person.fullname else ' '
        splitted = person.fullname.split(split_char)
        first_word = splitted.pop(0).strip()
        last_words = " ".join(splitted).strip()
        if ',' in person.fullname:
            person.lastname = first_word
            person.firstname = last_words
        else:
            person.firstname = first_word
            person.lastname = last_words
        person.exact = False

    if not person.firstname or not person.lastname:
        return None
    return person
```

4.3.3 Rimozione degli autori duplicati

L'ultima parte dell'algoritmo si occupa della rimozione degli autori duplicati dai risultati della funzione, in modo da ottenere un comitato di programma composto da autori unici. Questo è necessario perché si verifica che in alcuni casi lo stesso autore è presentato più volte nella Call For Paper, per esempio perché è un presidente di comitato. Per evitare la rimozione errata di casi di omonimia, gli autori vengono dichiarati duplicati e quindi esclusi secondo una chiave realizzata dall'interpolazione delle stringhe di nome completo, affiliazione e nazione estratti: per ogni membro del comitato viene generata questa chiave e, se ad un'iterazione essa è già stata osservata, il membro viene rimosso.

4.3.4 Implementazione dell'algoritmo

Di seguito si riporta l'implementazione dell'intero algoritmo di estrazione del comitato di programma:

Listing 7: Algoritmo di estrazione del comitato di programma

```
1  def extract_committee(program_sections, nlp):
2      loss_threshold = NER_LOSS_THRESHOLD
3      program_committee = list()
4      for section in program_sections:
5          n_section_people = list()
6          step = 0
7          text_lines = section.splitlines()
8          while True:
9              n_step_people = list()
10             for offset in range(0, step + 1):
11                 n_people = 0
```

```
12                  step_lines = [l for i, l in enumerate(text_lines) if
                     ↪  (
13                      len(l) >= 4 and i % (step + 1) == offset)]
14                  for doc in nlp.pipe(step_lines, n_threads=16,
                     ↪  batch_size=10000):
15                      if len([e.text for e in doc.ents if e.label_ ==
                         ↪  'PERSON']):
16                          n_people += 1
17                  n_step_people.append(n_people)
18              n_section_people.append(n_step_people)
19              if(max(n_section_people[step]) < loss_threshold *
                 ↪  max([max(i)
20                 for i in n_section_people])):
21                  break
22              step += 1
23              if step > 3:
24                  return []
25
26          offset =
             ↪  n_section_people[-2].index(max(n_section_people[-2]))
27          regex = re.compile(r"^\W*([\"\w\. '’]+)", re.MULTILINE)
28          section_people = list()
29          for i in range(offset, len(text_lines), step):
30              results = regex.search(text_lines[i])
31              name = results.group(1).strip() if results else ""
32              STRIP_CHARS = string.punctuation + " ()--"
33              if step == 1:
34                  affiliation = text_lines[i].replace(name,
                     ↪  "").strip(STRIP_CHARS)
35              else:
36                  affiliation = ', '.join([l.strip(STRIP_CHARS) for l
37                                           in text_lines[(i + 1):(i +
                                             ↪  step)]])
38
39              affiliation, affiliation_country =
                 ↪  _extract_affiliation(affiliation)
40              person = _extract_person_name(name)
41              if not person:
```

```
42                  continue
43              person.affiliation = affiliation
44              person.affiliation_country = affiliation_country
45              section_people.append(person)
46
47          n_not_exact = len([True for p in section_people if not
            ↪ p.exact])
48          if n_not_exact / len(section_people) >= 0.5:
49              p_to_add = [p for p in section_people if p.exact]
50              if p_to_add:
51                  program_committee += p_to_add
52          else:
53              program_committee += section_people
54
55      seen = set()
56      unique_committee = []
57      for a in program_committee:
58          compare = (a.getattr('fullname') + a.getattr('affiliation')
            ↪ +
59                     a.getattr('affiliation_country'))
60          if compare not in seen:
61              unique_committee.append(a)
62              seen.add(compare)
63
64      if len(unique_committee) < 5:
65          return []
66      return unique_committee
```

Capitolo 5

Ottenimento dei dati degli autori

Una volta estratte le informazioni degli autori che compongono il comitato di programma, essi vengono ricercati su Scopus per poterne ricavare l'Electronic Identifier (EID) e vengono salvati su base di dati persistente, in modo da poter successivamente confrontare i loro dati nella parte di analisi statistica.

5.1 Electronic Identifier (EID) molteplici

Elsevier usa lo Scopus Author Identifier per assegnare ogni autore nel database a un identificativo univoco e raggruppare tutti i documenti scritti da lui. Per determinare quali nomi dovrebbero essere raggruppati sotto un unico identificativo l'algoritmo di Scopus tiene conto di vari

dati tra cui affiliazione, aree di interesse, nazione, co-autori, ecc.[11] Siccome Elsevier stessa dichiara che "Nonostante la raffinatezza del profilo algoritmico utilizzato da Scopus, gli algoritmi non possono sempre abbinare tutti i documenti a un singolo profilo con una precisione del 100%"[12], è possibile che, ricercando un autore per nome, più risultati di ricerca rappresentino semanticamente la stessa persona, come mostrato nell'esempio in figura 5.1.

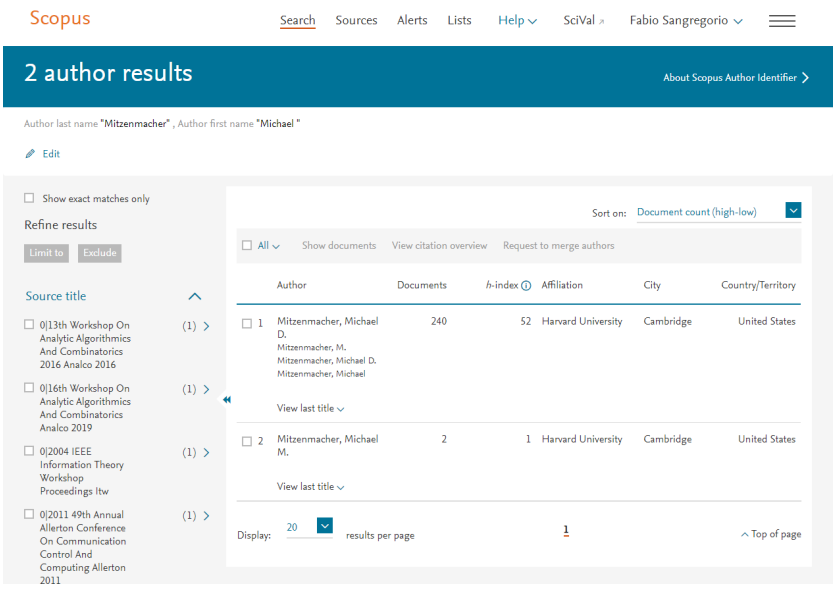

Figura 5.1: Esempio di più risultati di ricerca che rappresentano semanticamente lo stesso autore

Ad un autore possono quindi essere associati più EID, i quali vengono salvati in database e saranno utilizzati nelle fasi successive della raccolta dati. Per esempio, dopo l'ottenimento delle citazioni presenti in una pubblicazione tramite API Scopus, il salvataggio degli autori citati da essa avviene tramite il confronto dei loro EID con quelli nel

database locale, in modo da evitare la creazione di autori duplicati, in quanto queste citazioni potrebbero essere registrate su Scopus su un autore diverso rappresentante però la stessa persona del database locale.

5.2 Distanza di Levenshtein e fuzzywuzzy

La ricerca degli autori sulla piattaforma Scopus avviene tramite query basata su nome, cognome, affiliazione e nazione. E' possibile però che l'affiliazione nei risultati di ricerca sia scritta in modo diverso da quella nella query anche se semanticamente equivalente (ad esempio "Cambridge University" e "University of Cambridge"), perciò si usa la distanza di Levenshtein per determinare la similarità delle due affiliazioni.

La distanza di Levenshtein tra due stringhe a e b (di lunghezza |a| e |b|) è definita matematicamente come $lev_{a,b}(|a|,|b|)$, dove

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{se } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{altrimenti.} \end{cases}$$

Essa rappresenta il numero minimo di modifiche elementari che consentono di trasformare una stringa A in una stringa B. Si considerano modifiche elementari:

- la cancellazione di un carattere

- la sostituzione di un carattere con un altro
- l'inserimento di un carattere

Ad esempio, la distanza di Levenshtein tra le parole "conferenza" e "concorrenza" è 3, in quanto le modifiche necessarie sono:

- "conferenza" → "concerenza" (sostituzione di "f" con "c")
- "concerenza" → "concorenza" (sostituzione di "e" con "o")
- "concorenza" → "concorrenza" (inserimento di "r")

5.2.1 Funzionamento di fuzzywuzzy

Il progetto utilizza la libreria fuzzywuzzy per computare questa metrica in modo da calcolare il grado di confidenza che due stringhe siano simili. Essa offre vari rapporti di similitudine, ciascuno utile in diversi casi d'uso, i quali ritornano un punteggio da 0 a 100.

Ratio

Il rapporto più semplice è "ratio", il quale si basa esclusivamente sulla distanza di Levenshtein.

```
fuzz.ratio("Harvard", "Harvard University")
>> 56
```

Partial ratio

Data una stringa di lunghezza k e una di lunghezza m tale che m > k, "partial_ratio" ritorna il punteggio della corrispondenza migliore delle sottostringhe di lunghezza k.

```
fuzz.partial_ratio("Harvard", "Harvard University")
>> 100
```

Token sort ratio

Le funzioni token tokenizzano le stringhe e le preprocessano togliendo i segni di punteggiatura. In "token_sort_ratio" i token sono ordinati alfabeticamente e poi uniti. Dopo di ciò viene eseguito un semplice "ratio" per ottenere il punteggio.

```
fuzz.partial_ratio("University of Cambridge", "Cambridge
↪ University")
fuzz.token_sort_ratio("University of Cambridge", "Cambridge
↪ University")
>> 45
>> 93
```

Token set ratio

Se le stringhe sono di lunghezza molto diversa fra loro "token_sort_ratio" non basta. "token_set_ratio" rimuove i token comuni (l'intersezione

tra le due stringhe) poi esegue un semplice "ratio" a coppie tra le nuove stringhe seguenti:

- s1 = token ordinati nell'intersezione
- s2 = token ordinati nell'intersezione + token rimanenti ordinati della stringa 1
- s2 = token ordinati nell'intersezione + token rimanenti ordinati della stringa 2

Siccome s1 rimane sempre uguale, il punteggio incrementa quando esso occupa una percentuale maggiore dell'intera stringa e il resto delle stringhe sono più simili.

```
fuzz.token_sort_ratio("University of Cambridge", "College of
↪ Cambridge")
fuzz.token_set_ratio("University of Cambridge", "College of
↪ Cambridge")
>> 56
>> 75
```

5.2.2 Scelta del rapporto

Siccome nel caso d'uso del progetto i risultati di ricerca dell'affiliazione possono rientrare in uno qualsiasi dei casi elencati sopra, sono stati effettuati test su dati reali per scegliere l'algoritmo di assegnazione di punteggio migliore. token_set_ratio, essendo il più complesso tra

quelli descritti, è risultato in una percentuale maggiore di assegnazioni corrette anche quando gli altri scorer falliscono, ottenendo una maggiore confidenza. Di seguito si mostrano due esempi di questi test, in formato "punteggio: affiliazione"

```
1  Affiliazione reale: Chinese Academy of Sciences, China
2
3  Risultati:
4  ratio
5  55: Chinese Academy of Sciences, Institute for Information
    ↪  Engineering, SKLOIS, Beijing, China
6  48: Xiamen University, China
7
8  partial_ratio
9  91: Chinese Academy of Sciences, Institute for Information
    ↪  Engineering, SKLOIS, Beijing, China
10 62: China University of Geosciences, School of Geophysics and
    ↪  Information Technology, Beijing, China
11
12 token_sort_ratio
13 55: Chinese Academy of Sciences, Institute for Information
    ↪  Engineering, SKLOIS, Beijing, China
14 46: Xiamen University, China
15
16 token_set_ratio
17 100: Chinese Academy of Sciences, Institute for Information
    ↪  Engineering, SKLOIS, Beijing, China
18 52: University of Science and Technology of China, Department of
    ↪  Electronics Science and Technology, Hefei, China
```

Listing 8: Esempio 1: test per la scelta dello scorer

Listing 9: Esempio 2: test per la scelta dello scorer

```
1  Affiliazione reale: Wuhan University, China
2
```

```
Risultati:
ratio
45: University of Alberta, Edmonton, AB, Canada
38: Wuhan University, State Key Laboratory of Software Engineering, School of Computer Science, China
36: Zhongnan University of Economics and Law, School of Information and Safety Engineering, Wuhan
35: Shanghai Jiao Tong University, Med-X Research Institute, School of Biomedical Engineering, China

partial_ratio
83: Wuhan University, State Key Laboratory of Software Engineering, School of Computer Science, China
74: Zhongnan University of Economics and Law, School of Information and Safety Engineering, Wuhan
61: Shanghai Jiao Tong University, Med-X Research Institute, School of Biomedical Engineering, China
39: University of Alberta, Edmonton, AB, Canada

token_sort_ratio
45: University of Alberta, Edmonton, AB, Canada
38: Wuhan University, State Key Laboratory of Software Engineering, School of Computer Science, China
37: Zhongnan University of Economics and Law, School of Information and Safety Engineering, Wuhan
30: Shanghai Jiao Tong University, Med-X Research Institute, School of Biomedical Engineering, China

token_set_ratio
100: Wuhan University, State Key Laboratory of Software Engineering, School of Computer Science, China
84: Zhongnan University of Economics and Law, School of Information and Safety Engineering, Wuhan
84: Shanghai Jiao Tong University, Med-X Research Institute, School of Biomedical Engineering, China
62: University of Alberta, Edmonton, AB, Canada
```

5.3 Descrizione del processo

Il processo è diviso logicamente in due parti sequenziali: la prima sezione si occupa della ricerca dell'autore all'interno del database Scopus per mezzo delle API a disposizione, la seconda è dedicata a salvataggio e indicizzazione dell'autore nel database locale.

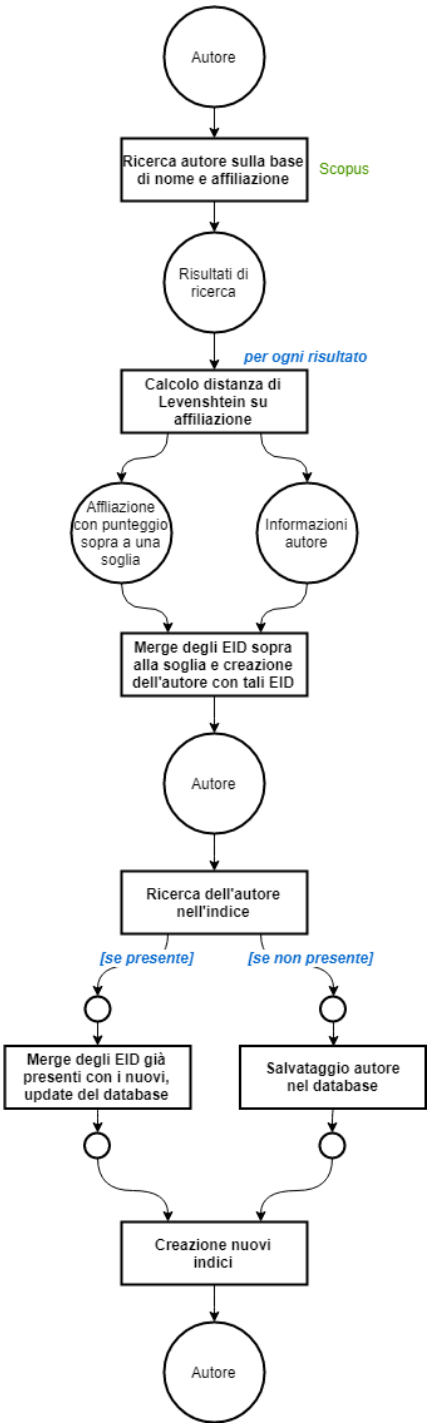

Figura 5.2: Algoritmo di ottenimento degli autori

5.3.1 Ricerca dell'autore

Inizialmente si esegue la ricerca dell'autore sulla base di nome, secondi nomi, cognome e e affiliazione. Le API di Scopus ritornano risultati di ricerca aventi nome e cognome esattamente identici rispetto a quelli ricercati, mentre l'affiliazione è spesso simile ma non medesima. Per questo motivo si utilizza la funzione `fuzz.extractOne()` della libreria fuzzywuzzy per estrarre l'autore corretto sulla base dell'affiliazione: questo metodo accetta un vettore di stringhe (nel caso d'uso attuale le affiliazioni dei risultati di ricerca) e ritorna la stringa con punteggio migliore rispetto a un dato scorer (nel progetto `fuzz.token_set_ratio`). L'affiliazione estratta è quella con punteggio maggiore nel caso in cui esso superi una certa soglia configurabile tramite la costante `FUZZ_THRESHOLD`; nel caso contrario si sostiene che la confidenza sulla correttezza dell'affiliazione più simile rispetto all'originale è troppo bassa, per cui la ricerca dell'autore fallisce, ultimamente escludendolo dai risultati. Siccome una stessa persona può essere rappresentata da più risultati di ricerca, una volta trovata l'affiliazione corretta si cercano tra i risultati gli autori con affiliazione identica a questa e ne vengono quindi salvati gli Electronic Identifier in un vettore.

Di seguito viene mostrato il codice descritto:

```
def find_author(author):
    score_threshold = 70
    aff = author.getattr('affiliation')
    if not aff:
        return None
    query = 'AUTHFIRST("{}") AND AUTHLASTNAME("{}") {}'.format(
        author.getattr('firstname'),
        ' '.join(filter(None, [author.getattr('middlename'),
                  author.getattr('lastname')])),
        f'AND AFFIL("{aff}")' if aff else '')
    try:
        possible_people = AuthorSearch(query).authors
    except Exception:
        return

    if not possible_people:
        return None
    aff_list = [f"{p.affiliation}, {p.country}" for p in
     ↪  possible_people]
    affiliation, fuzz_score = process.extractOne(author.affiliation,
                                  aff_list,
                                  scorer=fuzz.token_set_ratio)
    if fuzz_score > FUZZ_THRESHOLD:
        author.eid_list = [int(p.eid.split('-')[-1]) for p in
         ↪  possible_people
                           if affiliation.lower() ==
                            ↪  f"{p.affiliation},
                            ↪  {p.country}".lower()]
    else:
        return None

    return author
```

Listing 10: Esempio 2: test per la scelta dello scorer

5.3.2 Salvataggio in database locale e indice

Il salvataggio della persona nel database locale inizia ricercando la presenza di autori in possesso di uno degli EID nella lista di identificativi della persona che si vuole salvare. Mongoengine offre la possibilità di effettuare una query su un attributo di tipo lista fornendo un vettore di valori come parametro, tramite il seguente metodo:

```
Author.objects(eid__in=scopus_author.eid_list).first()
```

Siccome questa operazione viene eseguita per ogni citazione presente in ogni articolo trovato nella conferenza (in media circa 100 citazioni in ogni articolo, 300 articoli per conferenza), essa rappresenta una parte significativa della pipeline e, con l'aumento del numero di conferenze analizzate, essa risulta poco scalabile e non efficiente (in quanto si tratta di verificare per ogni elemento di un vettore la presenza di esso in un altro vettore). Per aumentare l'efficienza è stato realizzato un indice che associa ogni EID al suo autore, risultando in un miglioramento delle performance di circa il 15%. Nel caso l'autore non sia già presente nel database locale esso viene inserito, altrimenti vengono unite le due liste di EID tramite una collezione Set (collezione non ordinata di elementi univoci) eliminando quindi gli eventuali duplicati,e successivamente aggiornato l'oggetto nella base di dati.

Capitolo 6

Conclusioni

Il software è stato testato su 50 delle conferenze classificate come migliori dal GII-GRIN-SCIE Conference Rating 2017 processando un totale di 110 edizioni differenti. Da queste sono stati estratti 222.648 autori, oltre a 988 membri di comitato di programma, e 11.626 articoli nei quali sono presenti una totalità di 892.650 riferimenti ad autori. Di questi, 20.402 sono ad un membro del comitato di programma. Ciò porta a una media di:

- 2 edizioni per conferenza estratte con successo
- 9 membri di comitato di programma per evento
- 105 pubblicazioni per conferenza
- 77 citazioni per articolo
- di cui 2 citazioni ad un membro di comitato di programma

Inoltre, nella base di dati sono presenti 160 autori in possesso molteplici EID, in conseguenza al motivo spiegato nei capitoli precedenti. La query sulla base di dati usata per ottenere questo dato è mostrata di seguito:

```
len(
  list(Author.objects.aggregate({
    '\$project': {
      '_id': 0,
      'multiple_eids': {'\$gt': [{'\$size': "\$eid_list"}, 1]},
    }
  }, {
    '\$match': {
      'multiple_eids': True
    }
  }))
)
```

Listing 11: Query che restituisce gli autori in possesso di molteplici EID

I dati estratti dal software permettono disparati studi. Lo strumento fornisce una prima analisi delle pubblicazioni collezionate, come mostrato di seguito.

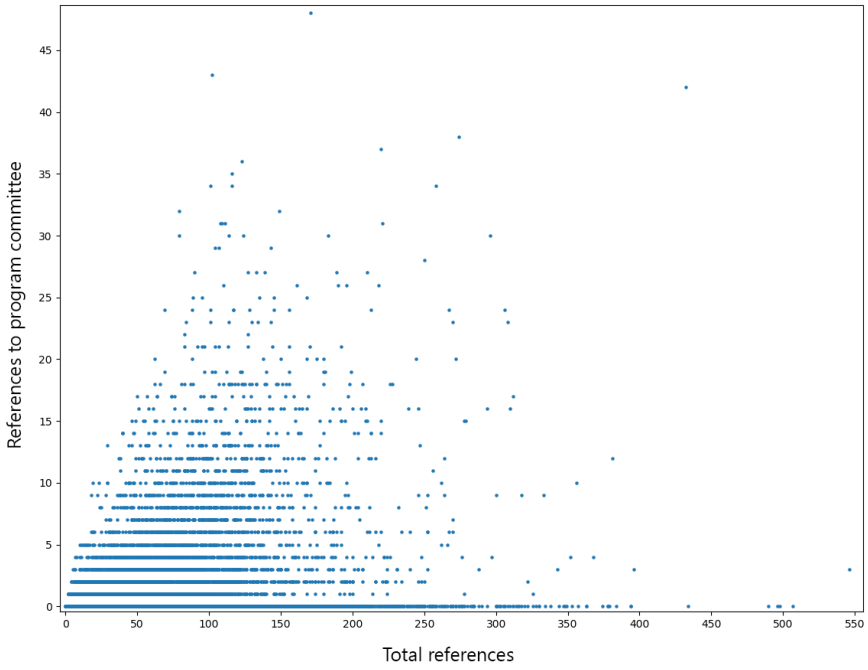

Figura 6.1: Distribuzione delle citazioni a membri del comitato di programma in relazione ai riferimenti totali presenti negli articoli

Il grafico in figura 6.1 descrive la distribuzione delle citazioni a membri del comitato di programma in relazione ai riferimenti totali presenti negli articoli, in cui ogni punto rappresenta una singola pubblicazione. Si nota che la maggior parte delle paper ne presenta nessuno o pochi, mentre svariate ne contengono una quantità considerevole in rapporto alle loro citazioni totali. Più precisamente, alcune arrivano anche ad avere il 60% di tutti i loro riferimenti a un membro del comitato di programma, come si vede dalla rappresentazione in figura 6.2 che mostra le pubblicazioni ordinate in modo decrescente secondo il rapporto

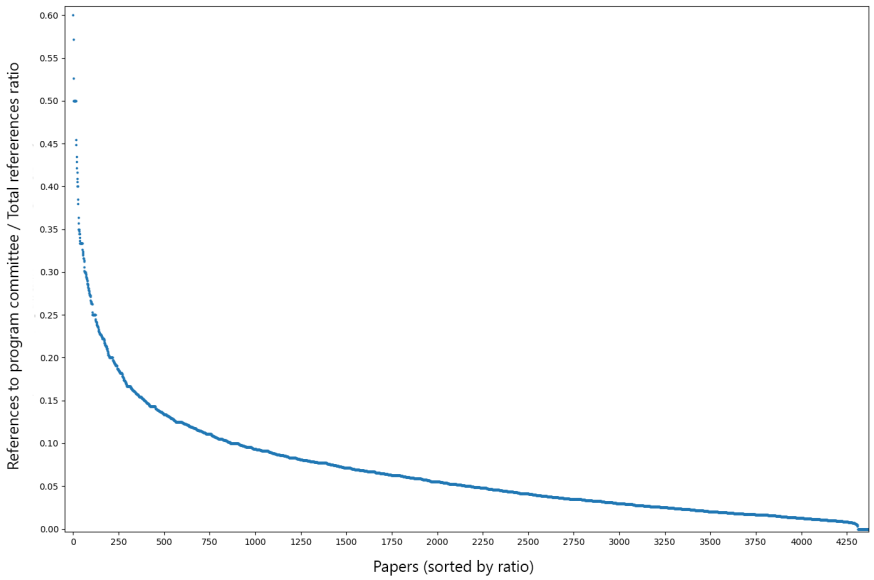

Figura 6.2: Pubblicazioni ordinate in modo decrescente secondo il rapporto delle citazioni al comitato rispetto alle loro citazioni totali

delle citazioni al comitato rispetto alle loro citazioni totali.

Si nota che la maggior parte degli articoli (circa il 90%) contengono al loro interno meno del 10% di riferimenti al comitato, mentre le prime 1.000 ne presentano una quantità discreta. Il 2% delle paper (le 250 con rapporto più alto) possiede più del 20% delle sue citazioni totali al comitato di programma. Potrebbe quindi risultare interessante l'analisi approfondita di questa parte minore della base di dati, studiando le edizioni delle singole conferenze che presentano questa anomalia e indagando gli autori citati in questi articoli, in modo da verificare la possibilità della presenza di un bias sistemico nell'editoria accademica.

L'ottenimento delle pubblicazioni che presentano il rateo maggiore

di citazioni al comitato di programma è possibile tramite la query di aggregazione seguente:

```
list(Paper.objects.aggregate(
    {
        '\$addFields': {
            'total_refs': {
                '\$add': [{'\$size': '\$committee_refs'}, {'\$size':
                 ↪  '\$non_committee_refs'}]
            },
            'committee_refs': {'\$size': '\$committee_refs'}
        }
    }, {
        '\$addFields': {
            'div': {
                '\$cond': [
                    {'\$eq': ['\$total_refs', 0]},
                    0,
                    {'\$divide': ['\$committee_refs',
                     ↪  '\$total_refs']}
                ]
            }
        }
    }, {
        '\$sort': {'div': -1}
    }, {
        '\$limit': 10
    }
))
```

Listing 12: Query che restituisce gli autori in possesso di molteplici EID

6.1 Sviluppi futuri

Il codice del software è progettato in modo modulare e facilita quindi l'integrazione di sviluppi futuri. Ogni elemento della pipeline richiede in ingresso un tipo ben definito di dato (ad esempio 'Author' o 'Conference') ed è indipendente dagli altri step, e mette a disposizione in output un tipo definito, permettendo facilmente l'aggiunta o la sostituzione dei componenti. Ad esempio, l'aggiunta di una nuova sorgente di Call For Papers è possibile tramite l'aggiunta di un componente posizionato a monte dello step di estrazione del comitato di programma e che espone in output il testo della CFP, il quale poi continuerà negli elementi successivi della pipeline.

Il progetto presenta tuttavia parti non ottimizzate, le quali possono essere migliorate per aumentare la performance del software. Al momento della pubblicazione, la libreria open source pybliometrics è ancora in sviluppo e di conseguenza non presenta delle ottimizzazioni che porterebbero ad una maggiore efficienza ed efficacia del software. Tra queste, essa attualmente dispone della possibilità di eseguire chiamate alle API Scopus di tipo "retrieval" ponendo in ingresso solo un identificativo da ottenere: le API di questo servizio accettano invece più identificativi in un'unica richiesta. Questa ottimizzazione è in sviluppo dal team di pybliometrics e porterebbe alla drastica diminuzione del numero di chiamate al servizio esterno: per ogni edizione di conferenza, infatti, le API Scopus vengono interpellate in media 114 volte, risul-

tando in 12.540 chiamate per i 110 eventi estratti ad oggi. Ciò risulta importante nel caso di estrazione massiva di dati, in quanto Elsevier pone delle limitazioni settimanali sulla quantità di richieste erogabili ad una chiave API (d'interesse del software sono 10.000 per Abstract Retrieval e 5.000 per Author search [13]) Inoltre, Scopus mette a disposizione un parametro in ogni tipo di richiesta chiamato "fields", il quale permette di specificare i campi del risultato che devono essere ritornati nella chiamata, funzionalità di cui pybliometrics non dispone ancora. Questa ottimizzazione porterebbe a una potenziale diminuzione del tempo di processamento delle chiamate e un minor peso dei risultati, aumentando quindi la velocità del software.

Bibliografia

[1] J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences*, 102(46):16569–16572, 2005.

[2] Wikicfp: A wiki for calls for papers. `http://www.wikicfp.com`.

[3] Scopus. `https://www.elsevier.com/solutions/scopus`.

[4] The gii-grin-scie conference rating. `https://www.elsevier.com/solutions/http://www.consorzio-cini.it/gii-grin-scie-rating.html`.

[5] mongoengine: a document-object mapper for working with mongodb from python. `http://mongoengine.org`.

[6] M. E. Rose and J. R. Kitchin. pybliometrics: Scriptable bibliometrics using a python interface to scopus. *SoftwareX*, 10, 2019.

[7] beautifulsoup4: Python library designed for quick turnaround projects like screen-scraping. https://www.crummy.com/software/BeautifulSoup/.

[8] fuzzywuzzy: Fuzzy string matching in python. https://github.com/seatgeek/fuzzywuzzy.

[9] Spacy: Industrial-strength natural language processing. https://spacy.io/.

[10] probablepeople: python library for parsing unstructured western names into name components. https://github.com/datamade/probablepeople.

[11] Scopus. Scopus author identifier. https://service.elsevier.com/app/answers/detail/a_id/11212/supporthub/scopus/.

[12] Scopus. Check, correct, submit: How to ensure accuracy in your scopus author profile. https://blog.scopus.com/posts/check-correct-submit-how-to-ensure-accuracy-in-your-scopus-author-profile.

[13] Elsevier. Scopus api quota limits. https://dev.elsevier.com/api_key_settings.html.