DBLP Project

Jacopo Manetti, June 2023

Introduzione

Questo progetto si concentra su un problema specifico nel campo dell'analisi delle reti di collaborazione scientifica: l'elaborazione e l'analisi del dataset DBLP. Questo dataset contiene una grande quantità di dati relativi alle pubblicazioni scientifiche, compresi dettagli come l'autore, il titolo, l'anno di pubblicazione, e molto altro.

Il DBLP è un database online di pubblicazioni informatiche che documenta milioni di articoli, spesso con metadati dettagliati, rendendolo un'importante fonte di dati per l'analisi delle reti scientifiche. Nonostante la sua importanza, il formato raw di DBLP può essere complesso da gestire, in quanto è organizzato come un file XML molto grande e intricato.

L' obiettivo principale di questa relazione è utilizzare i dati per rispondere a una serie di domande di ricerca.

Domande di Ricerca

01

Quale è la pubblicazione con il maggior numero di autori?

02

Quali sono le parole più utilizzate nei titoli delle pubblicazioni?

03

Quale è la pubblicazione con il maggior numero di autori popolari?

04

Quale coppia di autori ha collaborato di più?

Preparazione dei dati

L'obiettivo principale era ottenere otto dataframe dai file CSV estratti dal DBLP. Questa operazione era necessaria per analizzare i dati sia individualmente che nell'insieme.

Ogni file CSV aveva molte colonne, ma solo quelle contenenti 'autori', 'titoli', 'anno' erano rilevanti. Le colonne irrilevanti sono state quindi eliminate.

Abbiamo riscontrato problemi con la colonna 'anno' in due file, rappresentata come '2022|2022'. Questo è stato risolto utilizzando la funzione personalizzata check_year.

Infine, la colonna 'autori', presentata come stringa unica, è stata suddivisa in autori singoli. Questo è stato ottenuto utilizzando la funzione split di Python.

Funzione check_year

- Questa funzione prende in input una stringa e controlla se contiene il carattere '|'. Se è così, divide la stringa in due parti e confronta le due date. Se le date coincidono, converte l'anno in un intero. Se le date non coincidono, restituisce None. Se la stringa non contiene il carattere '|', la funzione converte direttamente l'anno in un intero.

```python
#funzione utile quando 'year' è un oggetto del tipo '2020|2020'
def check_year(year_string):
    if '|' in year_string:
        year1, year2 = year_string.split('|')
        if year1 == year2:
            return int(year1)  # converte in un intero
        else:
            return None  # restituisce None se le date non coincidono
    else:
        return int(year_string)  # se è un singolo anno, converte direttamente in intero
```

Codice di esempio per la creazione del dataframe

- Una volta che questi problemi sono stati risolti, abbiamo creato un dataframe per ciascuno dei sette file e infine li abbiamo uniti in un unico dataframe. Questo processo ci ha permesso di avere dati puliti e ordinati, pronti per l'analisi successiva.

```
# leggi il file .csv
df_6 = pd.read_csv('dblp-all-csv/out-dblp_phdthesis.csv', low_memory=False, delimiter=';')
# seleziona solo le colonne 'author', 'title' e 'year'
df_6 = df_6[['author', 'title', 'year']]
# Pulizia dei dati: rimuovere le righe con valori mancanti nelle colonne 'author' e 'title' e 'year'
df_6 = df_6.dropna(subset=['author', 'title', 'year'])
# trasforma la colonna 'year' in un intero
df_6['year'] = df_6['year'].apply(check_year)
# separa gli autori in una lista
df_6['author'] = df_6['author'].apply(lambda x: x.split('|'))

print(df_6)
```

Creazione del Grafo

- A causa dell'intimo rapporto tra autori e le loro pubblicazioni, abbiamo optato per un grafo bipartito per rappresentare queste relazioni. Questo tipo di grafo permette di mettere in evidenza le connessioni tra due diversi gruppi di entità, in questo caso gli autori e le loro opere.
- Il grafo bipartito si è rivelato la scelta migliore per diverse ragioni: ha rappresentato in modo naturale le connessioni tra autori e pubblicazioni, mostrando le relazioni multiple e bidirezionali, e ha fornito una visione chiara e intuitiva dell'intera rete di pubblicazioni e autori.
- Per costruire il grafo, abbiamo sfruttato la libreria Python NetworkX. Abbiamo implementato una funzione personalizzata, create_graph_from_df, per creare il grafo bipartito partendo dal dataframe elaborato in precedenza.

Funzione create_graph_from_df

Il grafo G viene inizialmente alimentato con i dati dal dataframe df. Per ogni riga del dataframe, la funzione aggiunge un nodo al grafo per la pubblicazione (con l'attributo 'year') e per ogni autore. Inoltre, per ogni autore, aggiunge un arco al grafo tra la pubblicazione e l'autore, indicando così la relazione tra i due.

```python
def create_graph_from_df(G, df):
    for _, row in df.iterrows():
        title = row['title']
        year = row['year']
        G.add_node(title, bipartite=0, year=year)  # aggiungi il nodo della pubblicazione con l'attributo 'year'
        for author in row['author']:
            G.add_node(author, bipartite=1)  # aggiungi il nodo dell'autore
            G.add_edge(title, author)  # aggiungi l'arco tra l'autore e la pubblicazione
    return G
```

Quesiti da risolvere - Domanda 1: Pubblicazione con il maggior numero di autori

Il nostro obiettivo era identificare il nodo corrispondente a una pubblicazione che aveva il maggior numero di archi connessi, cioè il maggior numero di autori collegati.

Idea: trovare il nodo titolo con grado più alto

Funzione
find_publication_with_most_authors

In questa funzione, filtriamo prima di tutto le pubblicazioni basandoci sull'anno, considerando solo le pubblicazioni realizzate fino all'anno specificato. Successivamente, per ogni pubblicazione, contiamo il numero di autori (ovvero il grado del nodo nel grafo) e teniamo traccia della pubblicazione con il numero massimo di autori.

```python
def find_publication_with_most_authors(G, year):
    publications = [n for n, d in G.nodes(data=True) if d['bipartite'] == 0 and d['year'] <= year]

    max_authors = 0
    max_publication = None
    for publication in publications:
        num_authors = G.degree(publication)
        if num_authors > max_authors:
            max_authors = num_authors
            max_publication = publication

    return max_publication, max_authors
```

Output della Domanda 1

Esempio ottenuto dal grafo unificato:

- Fino all'anno 1960, la pubblicazione con il maggior numero di autori è 'The FORTRAN automatic coding system.' con 13 autori.
- Fino all'anno 1970, la pubblicazione con il maggior numero di autori è 'Review: Book Review.' con 72 autori.
- Fino all'anno 1980, la pubblicazione con il maggior numero di autori è 'Review: Book Review.' con 72 autori.
- Fino all'anno 1990, la pubblicazione con il maggior numero di autori è 'Vorwort.' con 208 autori.
- Fino all'anno 2000, la pubblicazione con il maggior numero di autori è 'Introduction.' con 2065 autori.
- Fino all'anno 2010, la pubblicazione con il maggior numero di autori è 'Preface.' con 4921 autori.
- Fino all'anno 2020, la pubblicazione con il maggior numero di autori è 'Preface.' con 4921 autori.
- Fino all'anno 2023, la pubblicazione con il maggior numero di autori è 'Preface.' con 4921 autori.

Quesiti da risolvere - Domanda 2: Parole più utilizzate nei titoli delle pubblicazioni

L'obiettivo della seconda domanda era di identificare le parole più frequentemente utilizzate nei titoli delle pubblicazioni. Abbiamo implementato un approccio basato sulla tokenizzazione dei titoli e successiva conta delle frequenze delle parole.

Funzione find_most_common_word

1. Il codice inizia creando un set di "stop words" da varie lingue, parole generalmente escluse durante l'analisi del testo in quanto non portano molto significato.
2. La funzione principale, find_most_common_word, lavora su un grafo dato come input e un anno specifico. La funzione analizza ogni componente connessa del grafo, selezionando i nodi che rappresentano le pubblicazioni fatte fino all'anno specificato.
3. Quando la funzione trova una componente con almeno 30 pubblicazioni, unisce i titoli di tutte le pubblicazioni in una singola stringa, poi divisa in parole individuali o "token". Le parole vengono trasformate in minuscolo per facilitare la conta.
4. L'elenco di parole viene filtrato per rimuovere tutte le stop words e le parole non alfabetiche. La funzione Counter poi conta le occorrenze di ciascuna parola nell'elenco, identificando le parole più comuni.
5. Questo processo si ripete per ogni componente connessa nel grafo, producendo un elenco di parole comuni per ciascuna componente. Una componente connessa in un grafo è un sottoinsieme di nodi che sono interconnessi tra loro.

```python
english_stopwords = set(stopwords.words('english'))
italian_stopwords = set(stopwords.words('italian'))
french_stopwords = set(stopwords.words('french'))
german_stopwords = set(stopwords.words('german'))
stop_words = english_stopwords.union(italian_stopwords, french_stopwords, german_stopwords)

def find_most_common_word(G, year):
    components = list(nx.connected_components(G))

    common_words = []

    for component in components:
        publications = [node for node in component if G.nodes[node]['bipartite'] == 0 and G.nodes[node]['year'] <= year]

        if len(publications) >= 30:
            titles = ' '.join(publications)
            words = nltk.word_tokenize(titles.lower())
            words = [word for word in words if word.isalpha() and word not in stop_words]
            common_word = Counter(words).most_common(1)
            common_words.append(common_word)

    return common_words
```

Output della Domanda 2

Esempio ottenuto dal grafo unificato (fino al 1990):

Fino all'anno 1960, le parole più comuni nelle componenti con almeno 30 pubblicazioni sono:
- 'computer' appare 143 volte

Fino all'anno 1970, le parole più comuni nelle componenti con almeno 30 pubblicazioni sono:
- 'computer' appare 843 volte
- 'note' appare 12 volte

Fino all'anno 1980, le parole più comuni nelle componenti con almeno 30 pubblicazioni sono:
- 'system' appare 3341 volte
- 'axiom' appare 4 volte
- 'note' appare 18 volte

Fino all'anno 1990, le parole più comuni nelle componenti con almeno 30 pubblicazioni sono:
- 'systems' appare 12473 volte
- 'codes' appare 24 volte
- 'isols' appare 9 volte
- 'logic' appare 14 volte
- 'axiom' appare 4 volte
- 'note' appare 18 volte
- 'pp' appare 17 volte
- 'pp' appare 28 volte
- 'isbn' appare 16 volte

Quesiti da risolvere - Domanda 3: Pubblicazione con il maggior numero di autori popolari

Un autore ha un punteggio di popolarità pari al suo numero di pubblicazioni (quindi al suo grado).

L'idea alla base di questa funzione è di trovare la pubblicazione con la somma dei gradi dei vicini (autori) più alta .

Funzione find_publication_with_most_popular_authors

La funzione inizia creando un dizionario author popularity in cui le chiavi sono i nomi degli autori e i valori sono i loro gradi nel grafo. Successivamente, si selezionano le pubblicazioni pubblicate fino all'anno di riferimento.

Per ciascuna pubblicazione, si calcola uno "score di popolarità" sommando le popolarità di tutti i suoi autori. Questo score rappresenta una misura della popolarità complessiva della pubblicazione, basata sulla popolarità dei suoi autori.

La funzione mantiene traccia della pubblicazione con lo score di popolarità più alto finora e, una volta che tutte le pubblicazioni sono state esaminate, restituisce quella pubblicazione e il suo score di popolarità.

```python
def find_publication_with_most_popular_authors(G, year):
    author_popularity = {n: G.degree(n) for n, d in G.nodes(data=True) if d['bipartite'] == 1}

    publications = [n for n, d in G.nodes(data=True) if d['bipartite'] == 0 and d['year'] <= year]

    max_popularity_score = 0
    max_popularity_publication = None
    for publication in publications:
        popularity_score = sum(author_popularity.get(author, 0) for author in G.neighbors(publication))
        if popularity_score > max_popularity_score:
            max_popularity_score = popularity_score
            max_popularity_publication = publication

    return max_popularity_publication, max_popularity_score
```

Output della Domanda 3

Esempio ottenuto dal grafo unificato:

- Fino all'anno 1980, la pubblicazione con il maggior numero di autori popolari è 'Author's reply.' con un punteggio di popolarità di 4301.
- Fino all'anno 1990, la pubblicazione con il maggior numero di autori popolari è 'Vorwort.' con un punteggio di popolarità di 16978.
- Fino all'anno 2000, la pubblicazione con il maggior numero di autori popolari è 'Introduction.' con un punteggio di popolarità di 189580.
- Fino all'anno 2010, la pubblicazione con il maggior numero di autori popolari è 'Editorial.' con un punteggio di popolarità di 551050.
- Fino all'anno 2020, la pubblicazione con il maggior numero di autori popolari è 'Editorial.' con un punteggio di popolarità di 551050.
- Fino all'anno 2023, la pubblicazione con il maggior numero di autori popolari è 'Editorial.' con un punteggio di popolarità di 551050.

Quesiti da risolvere - Domanda 4: Coppia di autori che ha collaborato di più

Per raggiungere questo obiettivo è stato costruito un nuovo grafo invece di collegare autori e pubblicazioni, abbiamo creato un grafo in cui gli autori erano collegati direttamente tra loro nel caso avessero collaborato su una pubblicazione.

Idea: Aggiungere un peso agli archi, incrementarlo ogni volta che 2 autori collaborano e trovare l'arco di peso massimo.

Funzione create_author_graph_from_df

Per ogni riga presente nel dataframe, abbiamo preso l'elenco dei suoi autori e abbiamo generato un collegamento, o "arco", tra ogni coppia di autori.

Se la coppia aveva già collaborato in passato, abbiamo incrementato il "peso" del loro collegamento nel grafo di uno. Se la coppia non era stata ancora collegata, abbiamo creato un nuovo arco con un peso iniziale di uno.

```python
def create_author_graph_from_df(G, df):
    for _, row in df.iterrows():
        authors = row['author']
        # Creiamo un edge per ogni coppia di autori in ogni pubblicazione
        for i in range(len(authors)):
            for j in range(i + 1, len(authors)):
                if G.has_edge(authors[i], authors[j]):
                    # se l'edge esiste già, incrementiamo il peso
                    G[authors[i]][authors[j]]['weight'] += 1
                else:
                    # altrimenti, creiamo un nuovo edge con peso 1
                    G.add_edge(authors[i], authors[j], weight=1)
    return G
```

Funzione find_max_collaboration

La funzione *find max collaboration* esamina ogni collegamento tra gli autori nel grafo e restituisce la coppia di autori con il peso più elevato, ovvero con il numero maggiore di collaborazioni.

In pratica, scandisce tutte le connessioni nel grafo e tiene traccia del collegamento con il peso massimo. Alla fine del processo, restituisce la coppia di autori (ovvero il collegamento) che ha il peso più alto, e il relativo peso stesso

```python
def find_max_collaboration(G):
    max_weight = 0
    max_edge = None
    for u, v, data in G.edges(data=True):
        if data['weight'] > max_weight:
            max_weight = data['weight']
            max_edge = (u, v)
    return max_edge, max_weight
```

Output della Domanda 4

Gli autori che hanno collaborato di più nel nostro set di dati sono **Makoto Takizawa 0001** e **Tomoya Enokido**, con un totale di **550 collaborazioni**.

Considerazioni finali sull’output

1. Evidenziamo un trend di crescita nel numero di autori per pubblicazione. Il numero di autori per singola pubblicazione è aumentato esponenzialmente nel corso del tempo, da 13 negli anni '60 a 4921 nel 2023.

Questo aumento può riflettere un cambiamento nel modo in cui la ricerca viene condotta, con una maggiore enfasi sulla collaborazione e la ricerca interdisciplinare.

2. Analizzando le parole frequenti nelle pubblicazioni scientifiche, abbiamo notato che negli anni '60 e '70 la parola 'computer' era predominante, riflettendo l'emergere dell'informatica. Negli anni '80, termini come 'system', 'logic' e 'codes' suggerivano una crescente specializzazione. Con l'arrivo del nuovo millennio, nel 2000, la parola 'using' indicava un focus spostato verso il software e le applicazioni. Dal 2010, l'emergere di parole come 'database', 'web', 'search' sottolineava l'importanza del web e della gestione dei dati. Infine, nel 2023, parole come 'circuits', 'information', 'network' e 'security' segnalavano le nuove aree di ricerca nell'informatica.

3. Il File 5 (mastersthesis) e 6 (phdthesis) mostrano che tutte le tesi di laurea hanno un solo autore, tipico per questo tipo di lavoro. Non si trovano parole comuni nelle componenti con almeno 30 pubblicazioni, a causa del fatto che il numero di tesi scritte da una singola persona è molto esiguo.

4. In generale, sembra che il punteggio di popolarità dei documenti aumenti nel tempo, probabilmente riflettendo l'espansione della comunità accademica e l'aumento di autori che diventano 'popolari'.