Progetto per il corso di Social Computing A.A. 2024/25

1. Introduzione

Il progetto si focalizza sulla costruzione del grafo delle dipendenze di un pacchetto software, con l'obiettivo di descriverne le relazioni tra i vari componenti e moduli coinvolti.
In questo contesto, ogni nodo del grafo rappresenta un'unità di codice, come una libreria o un modulo, mentre ogni arco indica una relazione di dipendenza, e vengono quindi evidenziati i componenti che dipendono da altri per funzionare correttamente.
Il pacchetto da analizzare sarà scelto da `npm`, un ecosistema di pacchetti software per `Node.js`. Il grafo sarà generato tramite la libreria `NetworkX`. Successivamente, andrà prodotta una visualizzazione interattiva con `PyVis`. Infine, verrà richiesto di effettuare un'attività di analisi dei dati ottenuti.

2. Il Problema

Node.js è una piattaforma basata sul motore V8 di Google Chrome, che fornisce strumenti, risorse e servizi necessari per eseguire codice JavaScript anche al di fuori del browser. `npm` (Node Package Manager) è il gestore di pacchetti predefinito per Node.js e uno dei più grandi ecosistemi open-source al mondo. Ad oggi, il registro di npm (https://www.npmjs.com/) ospita milioni di pacchetti open-source. Ogni pacchetto `npm` è un modulo software che può essere utilizzato per aggiungere funzionalità specifiche a un progetto. I pacchetti spesso non sono autonomi: per funzionare possono dipendere da altri pacchetti, creando così una rete di dipendenze.
Le dipendenze software rappresentano le librerie esterne necessarie per il corretto funzionamento di un progetto. Ogni pacchetto elenca queste dipendenze nel proprio file `package.json`, dove sono specificati i nomi e le versioni richieste. Quando un pacchetto dipende da un altro che, a sua volta, dipende da altri pacchetti, si creano le sotto dipendenze. Queste possono estendersi su più livelli, poiché anche le sotto dipendenze dipendono da altri pacchetti.
Un grafo delle dipendenze è una rappresentazione delle relazioni tra i vari componenti di un sistema, come ad esempio i pacchetti in un progetto software.
In un grafo delle dipendenze:

- I nodi rappresentano i singoli pacchetti software.
- Gli archi (diretti) tra i nodi indicano le relazioni di dipendenza tra questi pacchetti:
    - se il pacchetto A dipende dal pacchetto B, ci sarà un arco diretto che va da A a B.

Durante la fase di installazione di un pacchetto, le dipendenze e le sotto dipendenze vengono installate ricorsivamente. Ad esempio:

- Il pacchetto A dipende da B.
- B, a sua volta, dipende da C e D.
- Ciò significa che A dipende indirettamente anche da C e D, che verranno anch'essi installati.

La costruzione ricorsiva del grafo continua finché non si raggiungono i nodi terminali, ossia pacchetti che non hanno ulteriori dipendenze.
I grafi delle dipendenze sono DAG, ovvero grafi diretti e aciclici. Queste caratteristiche garantiscono che, quando si “risolvono” le dipendenze (come, ad esempio, durante l'installazione di pacchetti), non ci siano conflitti o ambiguità sull'ordine in cui devono essere gestiti i pacchetti. In un DAG, ogni nodo può essere “processato” in un ordine topologico, ossia un ordine in cui tutte le dipendenze di un nodo sono processate prima del nodo stesso.
I grafi delle dipendenze possono diventare estremamente complessi, con migliaia di nodi e archi, nei progetti moderni che spesso usano vari pacchetti. Questo comporta alcune sfide, in particolare perché le sotto dipendenze possono introdurre vulnerabilità e problemi di sicurezza, che non sono sempre immediatamente visibili agli utilizzatori della libreria.

3. Obiettivo

L’obiettivo del progetto è scrivere un software in grado di generare un grafo delle dipendenze di un qualunque pacchetto `npm` scelto arbitrariamente dall’utente, usando `NetworkX`. Gli archi del grafo sono diretti, in base alla relazione di dipendenza presente tra i due pacchetti che l’arco collega. Il grafo deve includere anche le sotto dipendenze, a qualunque livello.
Una volta generato il grafo, il software deve calcolare alcune metriche, che sono dettagliate nella sezione successiva. Successivamente, il software deve produrre i risultati nel formato richiesto. Infine, deve generare una visualizzazione interattiva del grafo, usando `PyVis`.

4. Cosa fare

Tutti i dettagli specifici su come svolgere il progetto vengono riportati in questa sezione.

4.1 Scelta del pacchetto seed

Ciascun team deve scegliere un pacchetto “seed” dal registro di npm, che abbia almeno 350 dipendenze, e scaricare i relativi dati.
I dati sulle dipendenze di ciascun pacchetto `npm` possono essere scaricati usando l’API (Application Programming Interface) ufficiale. Un API definisce come un programma può interagire con un altro per ottenere funzionalità o dati, e può essere “interrogata” sfruttando il protocollo HTTP. L’API ufficiale di `npm` è documentata al seguente link:

https://github.com/npm/registry/blob/main/docs/REGISTRY-API.md

Per praticità, viene fornito il seguente esempio di codice che scarica le dipendenze di primo livello del pacchetto `express` inviando una richiesta HTTP all’API, sfruttando la libreria `requests` di Python.

```
import requests
import json

PACKAGE = “express”
```

```
URL = “https://registry.npmjs.org/{}/latest”

u = URL.format(PACKAGE)
r = requests.get(u)
j = r.json()
dependencies = j[“dependencies”]

for e in dependencies:
   print(e)

with open(f”{PACKAGE}_dependencies.json”, “w”) as f:
   deplist = list(dependencies.keys())
   json.dump(deplist, f)
```

Una volta scelto il pacchetto “seed”, scaricate i dati relativi alle sue dipendenze e producete un’adeguata serializzazione dei dati in formato JSON.

4.2 Costruzione e visualizzazione del grafo

Il software dovrà costruire il grafo delle dipendenze del pacchetto scelto secondo le istruzioni seguenti.

1. Caricare in memoria i dati serializzati relativi alle dipendenze del pacchetto scelto.
2. Utilizzando `NetworkX`, costruire il grafo diretto dove:
    a. Ciascun nodo è un pacchetto da cui il pacchetto seed dipende.
    b. Ogni arco indica una relazione di dipendenza tra due pacchetti.
    c. Ogni sotto dipendenza è inclusa nel grafo, fino al livello più profondo.
3. Successivamente, costruire un secondo grafo dove:
    a. Il grafo prodotto al punto 2 viene trasformato in indiretto.
    b. il grafo indiretto viene aumentato utilizzando la tecnica del preferential attachment:
        i. Scegliendo `m`, aggiungere almeno altri `350` nodi e `350*m` archi.
4. Per ciascuno dei due grafi:
    a. Produrre una visualizzazione statica del grafo:
        i. Utilizzare il Layout di Fruchterman-Reingold.
5. Utilizzando `PyVis`, produrre una visualizzazione interattiva per ciascuno dei due grafi in cui:
    a. La dimensione di ciascun nodo dipende dal numero di archi in ingresso al nodo.
    b. Il colore di ciascun nodo viene determinato secondo i seguenti criteri:
        i. Grigio per i nodi appartenenti al primo quartile della distribuzione dei gradi (ossia, i nodi con grado più basso),
        ii. Blu per i nodi nel secondo quartile,
        iii. Viola per i nodi nel terzo quartile,
        iv. Giallo per i nodi nel quarto quartile.
6. Serializzare opportunamente i due grafi creati.

4.3 Analisi dei dati

Una volta prodotti i due grafi e le loro visualizzazioni secondo le istruzioni presentate nella sezione precedente, si dovrà svolgere un'attività di analisi dei dati secondo le istruzioni seguenti.

1. Calcolare le seguenti misure per ciascuno dei due grafi:
    a. Centro,
    b. Raggio,
    c. Distanza media,
    d. Distanza massima,
    e. Coefficiente di clustering medio,
    f. Transitività.
2. Calcolare le seguenti misure di centralità per i nodi dei due grafi:
    a. Betweenness centrality,
    b. Closeness centrality,
    c. Degree centrality,
    d. In-degree centrality,
    e. Out-degree centrality,
    f. Page rank.
3. Calcolare i seguenti coefficienti per stimare la "small-worldness" dei grafi:
    a. Coefficiente omega,
    b. Coefficiente sigma.
4. Serializzare correttamente le misure nel formato descritto nel seguente JSON.

```
{
   "global_metrics": {
       "graph_center": [ "node1", "node2" ],
       "radius": 0,
       "average_distance": 0.0,
       "max_distance": 0.0,
       "clustering_coefficient": 0.0,
       "transitivity": 0.0
   },
   "centrality_measures": {
       "node_name": {
           "betweenness_centrality": 0.0,
           "closeness_centrality": 0.0,
           "degree_centrality": 0.0,
           "out_degree_centrality": 0.0,
           "in_degree_centrality": 0.0,
           "pagerank": 0.0
       }
   }
   "small_worldness": {
      "omega": 0.0,
```

```
    "sigma": 0.0
  }
}
```

4.4 Relazione

Si dovrà infine documentare tutto il lavoro svolto ed eventuali assunzioni in una relazione di **al massimo** 5 pagine. Organizzare la relazione secondo la seguente struttura:

1. *Sommario* (al massimo mezza pagina): breve riassunto della relazione che riepiloga la metodologia seguita e i risultati ottenuti.
2. *Metodologia* (al massimo una-due pagine): descrizione della costruzione dei grafi e della loro visualizzazione:
    a. Indicate anche il pacchetto seed scelto dal registro di `npm`.
3. *Risultati* (al massimo una-due pagine): presentazione dei risultati ottenuti durante la fase di analisi dei dati.
4. *Conclusioni* (al massimo mezza pagina): considerazioni conclusive, eventuali attività extra.

Informazioni aggiuntive

- Tutte le misure, proprietà e verifiche richieste su grafi sono definite nella documentazione di NetworkX:
    - Consultatela per capire come rispondere alle varie richieste.
- Consegnate un notebook adeguatamente commentato e diviso in sezioni rispettando i punti del progetto “dall’alto verso il basso”.

5. Come consegnare

1. Dovete costituire dei gruppi che **devono** essere formati da **quattro** persone (i gruppi più o meno numerosi verranno penalizzati)
2. Si devono consegnare i seguenti elementi:
    - Relazione di **al massimo 5** pagine:
        - Scrivete anche i vostri nomi cognomi e numeri di matricola.
    - Risultati prodotti durante la fase di analisi dei dati, nel formato indicato, verificando di avere:
        - Una cartella `data/` contenente la serializzazione JSON delle dipendenze del pacchetto “seed” scelto,
        - Una cartella `graphs/` contenente una serializzazione JSON di ciascun grafo,
        - Una cartella `html/` contenente le visualizzazioni interattive prodotte mediante `PyVis`,
        - Il codice prodotto (in un **unico notebook** adeguatamente strutturato e commentato).
3. Consegnate quanto sopra via mail a tutti i docenti. Scrivete un unico messaggio indirizzato a
    - mizzaro@uniud.it
    - michael.soprano@uniud.it
    - akebli.hafsa@spes.uniud.it
    - michele.lizzit@uniud.it

    - e aggiungete in copia tutti i membri del gruppo.
    - Oggetto della mail nel formato:
        - `[Progetto SocCom - 2024/2025 - Consegna] cognome1_cognome2_cognome3_cognome4`
4. **Scadenza: Mercoledì 8 Gennaio 2025 AoE Timezone**
    - La scadenza è **rigida**: progetti consegnati in ritardo non verranno considerati
5. Punteggio:
    - 6 punti in trentesimi per progetti “perfetti” e che vanno anche oltre le consegne; indicativamente, i migliori 10%,
    - 5 punti per i seguenti 20%,
    - 4 punti per i seguenti 20%,
    - 3 punti per i seguenti 20%,
    - 2 punti seguenti 20%,
    - 1 punto per i seguenti 10%,
    - 0 punti per progetti non adeguati (a discrezione dei docenti) o per chi non consegna.