Social Network Analysis

Andrea Bagnoli
andrea.bagnoli@live.it
StudentID: 522553

Matteo Lupini
matteo.lupini5@gmail.com
StudentID: 580563

Simone Barandoni
simone.barandoni@gmail.com
StudentID: 523786

Lorenzo Mannocci
mannocci.lore@gmail.com
StudentID: 518263

INTRODUZIONE-RACCOLTA DATI

La rete sociale, oggetto di analisi nel seguente report, è una rete che è stata costruita mediante l'ausilio di un web crawler creato da noi in PHP. Il crawler lavora basandosi su una funzione ricorsiva che, partendo da un nodo iniziale dato, esplora tutti i link uscenti dal nodo e ricorsivamente visita in maniera sequenziale tutti i nuovi nodi scoperti. Ovviamente sono state gestite le possibili situazioni di duplicati e le situazioni in cui il link non risultava accessibile (la parte considerata è quella fino al sottodominio dell'URL). Abbiamo deciso di partire dall'url https://www.google.com/. Inoltre il web crawler in PHP è stato implementato in maniera tale che ad ogni iterazione della ricorsione tutta l'esplorazione dei nodi fosse scritta parallelamente in un file CSV. Così facendo abbiamo ottenuto un file dove da una parte abbiamo il sito web di partenza e nell'altra il sito web a cui punta il nodo sorgente.
Successivamente, solo per un motivo di performance dei vari algoritmi, si è deciso di rimpiazzare il valore nominale dei siti nel csv e sostituirlo con un ID. Come si può vedere nella cartella Github in cui sono contenuti i nostri dati abbiamo consegnato un file in cui abbiamo i valori nominali dei siti, uno con gli ID ed un ultimo file in cui è presente una mappatura (quasi come un dizionario) in cui ad ogni ID abbiamo fatto corrispondere il valore nominale del sito.

ESERCIZIO 0 – ANALISI DEL NETWORK

Per questo esercizio, la rete è stata analizzata mediante il linguaggio di programmazione R, utilizzando in prevalenza la libreria *igraph*. Sono stati condotti, inoltre, parallelamente dei confronti con tipi di modelli classici (Erdos-Renyi e Barabasi Albert) generati con stesse proprietà di base, ma non orientati. Il network estratto, quindi, è un network diretto composto da 26,426 **nodi** e da 116,081 **link**. Analizzando la **distribuzione dei gradi,** possiamo notare subito come il network ottenuto ha pressoché la stessa topologia di uno *scale free network model* seguendo come ci si poteva aspettare una *power law distribution*. Cioè si possono individuare pochi nodi fortemente connessi e tanti nodi con pochi link. Infatti, analizzando nel dettaglio gli istogrammi seguenti, riguardanti il grado dei nodi, si può notare come il caso del nostro network assomigli molto al caso di un Barabasi Model, mentre differisca da quello di un network casuale.

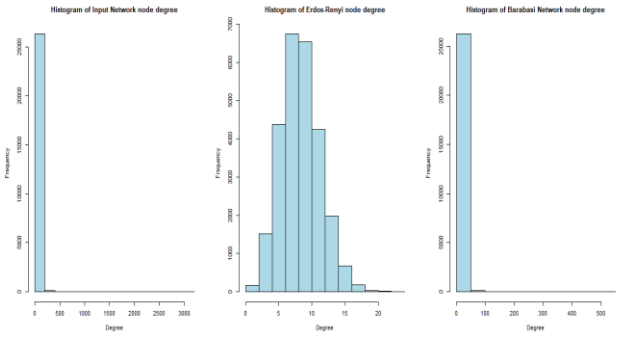

Figura 1: La figura rappresenta il confronto tra le distribuzioni dei gradi dei nodi nei tre network.

Nel modello Erdos-Renyi, infatti, abbiamo una distribuzione casuale simile ad una distribuzione normale con tanti nodi di grado vicino alla media. Negli altri due casi, abbiamo invece un risultato simile ad una distribuzione *power law*. Nello specifico notiamo alcuni nodi di grado molto elevato e

tantissimi nodi di grado molto piccolo. Osservazioni analoghe possono essere tratte anche analizzando singolarmente le varie distribuzioni in scala log-log dei 3 casi in esame[1].

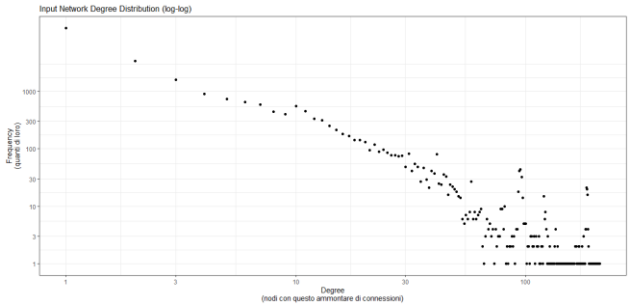

Figura 2: È rappresentata la distribuzione in scala logaritmica del grado dei nodi.

Si può notare come appunto nel nostro grafo (riportato a scopo di esempio nella figura sovrastante) ricalca in qualche maniera uno *scale free network* ed entrambi differiscono dal modello casuale.

Poiché il grado medio <k> è maggiore di 1, siamo di fronte, in tutti e tre i casi, ad un sistema supercritico con un'unica **componente gigante**. Tutti e tre i grafi hanno un unico cluster ed il grafo risulta connesso[2].

Ai fini delle analisi successive, prendendo come punto di riferimento alcuni dati estrapolati da esperti[3] che hanno analizzato la struttura complessiva di internet nel mondo, vediamo come i nostri risultati sono comunque lontani da questi ultimi. Infatti, <k>, che è di circa 4.4 nel nostro campione, sta al 6.8 nel vero valore di internet.

Per quanto riguarda invece l'**analisi dei path** vediamo che il nostro network ha un **diametro** di 99 con una **distanza media** di circa 19.5. Il modello Barabasi invece, che nel caso della distribuzione dei gradi è quello più simile al nostro network, in questo caso, si discosta molto dalla nostra rete (diametro di 7 e distanza media approssimativamente di 4.5). Il modello Erdos-Renyi risulta invece di diametro 9 con un percorso medio di circa 5.0[4]. Quindi, confrontando questi risultati, ci viene da pensare che la motivazione di una distanza media così alta e di un diametro così elevato sia dovuta sostanzialmente al fatto che si tratti di un grafo diretto, mentre i modelli di riferimento sono stati generati considerandoli indiretti. Per rappresentare graficamente la distribuzione degli shortest path, data la mole di dati, abbiamo estratto indipendentemente e casualmente 5000 nodi dai 3 grafi, sia nella colonna *source node* che nella colonna *target node*. Dopodiché abbiamo ricomposto la rete ed i risultati sono stati i seguenti:

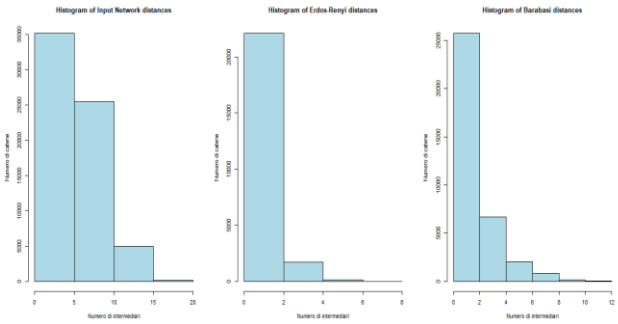

Figura 3: Rappresenta il confronto tra gli shortest path in un campione di 5000 nodi.

Cioè, fermo restando che i valori assoluti veri sono diversi perché si tratta per l'appunto di un campione, si può dire che la distanza minima media degli shortest path nel nostro caso resta maggiore rispetto agli altri due modelli perché nel primo istogramma abbiamo shortest path di distanza maggiore (visibile anche dall'asse delle x).

A livello, invece, di **global clustering coefficient** (approssimando però il grafo ad uno non orientato), il nostro grafo risulta essere abbastanza coeso con un global clustering coefficient di 0.147447 mentre ad esempio il modello random ha coefficiente di clustering molto basso di: 0.0003007959. La stessa cosa più o meno accade con *preferential attachment* (clustering coefficient nel nostro esperimento di 0.001735927). Il risultato seguente, in un certo senso, era atteso, dato che, i due modelli, in quanto tali, nel generare i grafi non tengono di conto delle possibili *triadic closure*, cosa che invece è lecito aspettarsi

[1] Vedi anche plot "Erdos-Renyi, Barabasi Network node degree distribution" in Github.

[2] In alcuni tentativi il modello Erdos-Renyi potrebbe anche risultare disconnesso con alcuni clusters singoli di un elemento.

[3] http://networksciencebook.com/chapter/3#networks-supercritical

[4] I risultati variano sensibilmente a seconda dell'esperimento.

attraverso gli *hyperlink* dei vari siti (considerando sempre l'approssimazione da grafo diretto a indiretto). Inoltre dalla seguente figura si può vedere, confermando ciò che viene descritto in letteratura[5], come il clustering coefficient, calcolato mediante il **local clustering coefficient** (approssimando sempre il nostro network ad un network indiretto), decresce all'aumentare del grado dei nodi. Questo è visibile dai seguenti grafici dove sull'asse x abbiamo il grado dei nodi e sull'asse y il local clustering coefficient in scala logaritmica.

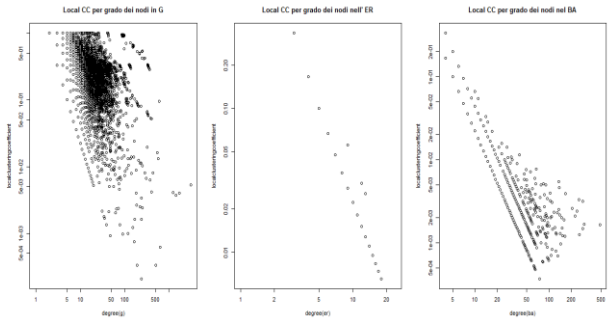

Figura 4: Rappresenta il decremento del local clustering coefficient all'aumentare del grado del nodo.

Per quanto riguarda la **densità** del grafo, ovvero il rapporto tra il numero di link e il numero di possibili link, otteniamo un valore in tutti i casi molto basso. Nel nostro grafo è di 0.0001659027, mentre negli altri due casi è all'incirca il doppio (0.000302715 nel Barabasi e 0.0003318053 nell' Erdos-Renyi). Comunque, in tutti e tre i casi, è un valore molto basso e lontano da 1, ovvero il caso dove avremmo un grafo completo. Le minime variazioni sono dovute all'orientamento del grafo se si confronta il nostro network con gli altri due modelli, ed al fatto che il modello Barabasi da noi generato ha un numero di link leggermente inferiore rispetto a quello dell'Erdos-Renyi[6].

Un'altra misura di densità utile in quest'analisi è il numero dei **triangoli** presenti nel grafo. Sempre approssimando il nostro ad un network indiretto abbiamo oltre 1,000,000 di triangoli contro i circa 1,000 del preferential attachment ed i circa 100 del modello casuale[7].

La prima misura di **centralità,** in parte già anticipata in precedenza, è la **degree centrality**. Nella nostra rete abbiamo che il nodo con degree centrality totale maggiore è di grado 3,117. Nello specifico trattandosi di un grafo diretto è stato opportuno calcolare sia la in-degree centrality che la out-degree centrality. Il nodo con in-degree più alta arriva a 3112, mentre il nodo con out-degree più alta a 634. Nel modello Barabasi il nodo di grado maggiore raggiunge una degree di circa 500 mentre nel modello casuale il nodo con degree massima arriva attorno ai 20.

Abbiamo calcolato, inoltre, la **betweenness centrality** e lo score massimo è di 0.1277338. La betweenness massima, invece, è intorno ai 0.06888905 nel Barabasi ed a 0.000876128 nel modello random. Intuitivamente, si nota che misurando l'importanza del nodo chiave delle reti sulla base della quantità di cammini minimi di cui fa parte, entrambi i modelli sono molto lontani dal picco del nodo 20 del nostro network, anche se il Barabasi ci si avvicina di più. Una precisazione è che nella maggior parte delle analisi sono stati inseriti risultati indicativi per i risultati riguardanti i due modelli badando più alle proporzioni che ai valori assoluti. Da queste analisi, dunque, notiamo come si era in parte già visto in alcune analisi precedenti, che il modello Barabasi è quello che capta vagamente alcuni aspetti topologici del nostro network, seppur avendo una betweenness massima molto lontana da quella del nostro network, ma sicuramente migliore del modello casuale. Stesse conclusioni più o meno possiamo trarre dal calcolo della **closeness centrality**. Lo score minimo[8] nodo nel nostro network è di 0.1125877. Nel modello Erdos-Renyi il picco minimo in un esperimento è stato di 0.1567775 mentre nel Barabasi di 0.1935302.

[5] http://networksciencebook.com/chapter/3#clustering-3-9

[6] Nel parametro m della funzione sample_pa della libreria igraph abbiamo inserito il grado medio/2 della nostro rete, ottenendo un Barabasi con stessi nodi ed oltre 105,000 link. Crediamo che i link ottenuti siano inferiori al network originale per un problema di implementazione matematica della funzione.

[7] Anche in questo caso a seconda dell'esperimento possiamo avere risultati leggermente diversi ma con le stesse proporzioni.

[8] Assumendo il nostro network come indiretto

ESERCIZIO 1: SPREADING

In questa fase andiamo ad analizzare il fenomeno della diffusione di una news tra siti web. Il passaggio da susceptible a infected è da intendere come l'adozione, ovvero la pubblicazione di una news da parte di un sito web. Supponiamo quindi che una news possa essere adottata solamente se passata da un altro sito web già "infetto". Ovviamente nella realtà non è così, in quanto un sito web può venire a conoscenza di una news anche tramite altri canali che non siano la network stessa. Possiamo quindi supporre che tutte le analisi fatte siano sovrastimate in ordine di tempo, ovvero che nella realtà il tempo di diffusione di una news sia minore.

L'analisi è stata fatta tramite l'utilizzo di due modelli: *SI model* e *threshold model*; ed è stata fatta sul nostro grafo e quello corrispondente per i modelli Erdos-Renyi e Barabasi.

SI model

In particolare, è stato adottato il **SI model**, in quanto una volta che un sito web "adotta" una news, e passa da susceptible a infected, non può più tornare sui suoi passi e tornare quindi a susceptible, o peggio ancora removed. Definiamo con f d'ora in poi la frazione di nodi infetti iniziale e conΨl'iterazione per cui si ha che la percentuale di nodi infected (e quindi anche susceptible) è del 100%.

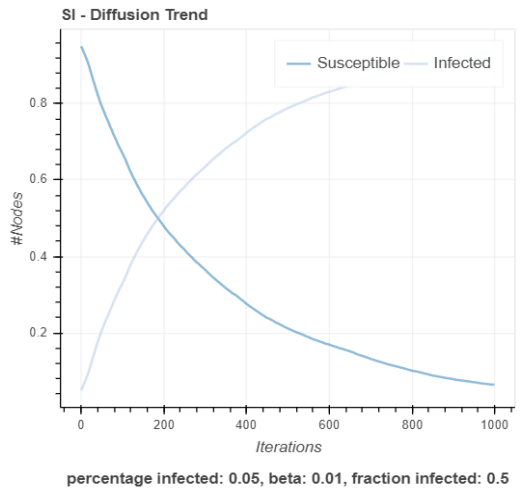

Figura 5: Diffusione sul nostro grafo, f=0.5, con β=0.01.

Nella tabella sottostante, fissato β, facciamo variare f e possiamo osservare una diminuzione del valore dell'iterazione a cui si raggiunge il 100% dei nodi infatti, all'aumentare della frazione iniziale di nodi infetti.

Tabella 1: Risultati nel nostro grafo con SI model.

β	0.01	0.01	0.01
f	0.2	0.5	0.7
Ψ	2000	1000	800

Mentre per l'ER Ψ=100 in tutti e tre i casi, e per il Barabasi $\Psi = 150$ sempre in tutti e tre i casi. Si nota quindi una netta diminuzione per i due modelli, dovuto soprattutto al fatto che sono grafi indiretti, e che quindi la diffusione è molto più rapida. E infine un caso estremo con $\beta = 0.1$ e $f = 0.7$, in cui si ha $\Psi = \{100, 15, 15\}$ rispettivamente per il nostro grafo, ER e Barabasi.

Nel caso del nostro grafo e del Barabasi abbiamo compiuto un'ulteriore analisi nel caso di diffusione a partire da un set di nodi infetti corrispondente all'1% degli hubs della rete (per ER no, in quanto in questo modello non sono presenti hubs), confrontandolo con una diffusione a partire da uno 1% randomico.

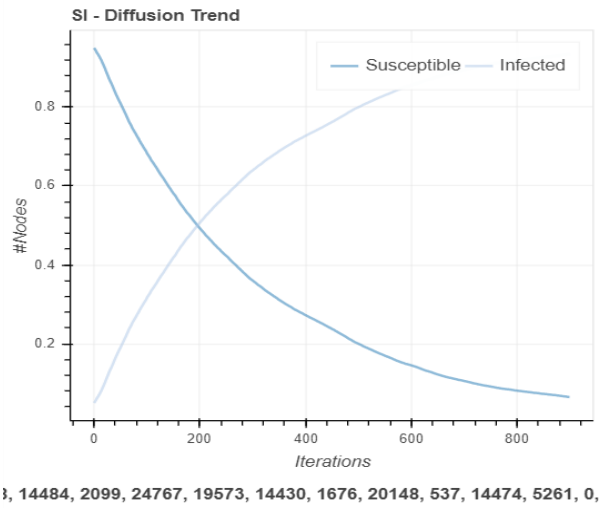

Figura 6: Diffusione sul nostro grafo, a partire dagli hubs, con β=0.01.

Si ha nel caso del nostro grafo con $\beta = 0.01$, $\Psi =$ 800 nodi infetti randomico e nel caso di diffusione a partire da hubs; mentre nel caso del Barabasi $\Psi =$ $\{150, 150\}$. Si può quindi notare come una diffusione

che parte da un set iniziale randomico di nodi e dagli hubs, non cambia, se non minimamente, relativamente al fattore tempo.
In entrambi casi simulando un caso estremo di $\beta = 0.006$ si ha un incremento netto di Ψ(2000 per il nostro grafo, e 1000 per il Barabasi).

Threshold model

Nel caso del **threshold model**, definiamo f frazione di nodi infetti iniziali; th la *threshold*; Φ la percentuale di nodi infetti finale.
Si può notare una diminuzione di Φ all'aumentare di th,come era lecito aspettarsi.

Tabella 2: Threshold model con f=0.3 e threshold che varia.

f	0.3	0.3	0.3
th	0.25	0.50	0.75
Φ	0.69	0.16	0.09

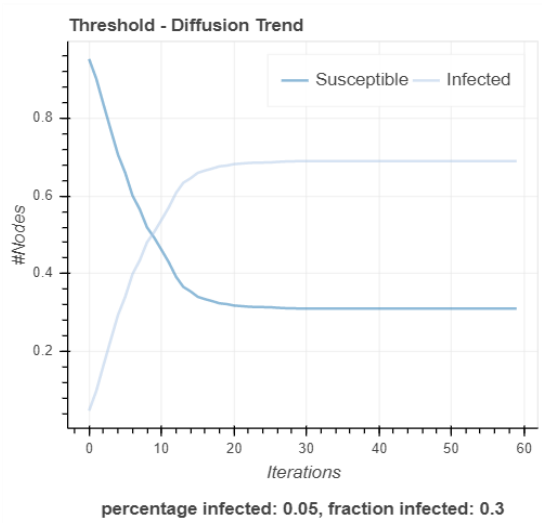

Figura 7: Diffusione sul nostro grafo, con th=0.25, e f=0.3.

In questa seconda tabella si hanno dei casi più interessanti: un primo caso base di riferimento e una diffusione a partire da un set di nodi infetti corrispondente agli hubs, in cui non si vede un miglioramento nella percentuale finale di diffusione e anche i tempi sono gli stessi.

Tabella 3: Threshold model sul nostro grafo: partendo da sinistra il caso di riferimento, diffusione a partire dagli hubs e infini con threshold crescente con il degree di nodi.

f	0.01	0.01 hubs	0.01
th	0.25	0.25	degree
Φ	0.67	0.64	1

Nel terzo caso è stata invece assegnata una threshold crescente in base al grado del nodo. L'idea è che il grado del nodo indichi l'importanza di un sito web all'interno del web. Di conseguenza è più difficile che un sito web con un'alta importanza, si faccia convincere dalla news proveniente da un altro sito (soprattutto in ottica di fake news). È stata utilizzata una funzione di interpolazione I: $[deg_{min}, deg_{max}] \rightarrow [0,1]$. In questo caso si osserva il raggiungimento del 100% di nodi della rete infetti, ma soprattutto una rapidissima diffusione in poche iterazioni, come si può osservare dal grafico sottostante.

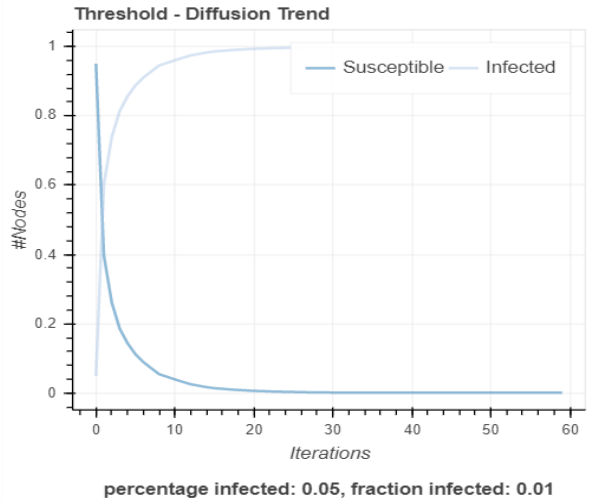

Figura 8: Diffusione sul nostro grafo, con threshold basato sul grado, e f = 0.1.

Nel caso del Barabasi e dell'ER, con $th \leq 0.25$ si ha un rapido raggiungimento di $\Phi = 1$. Per $th \geq 0.25$ si ha invece una diffusione poco superiore allo 0 (0.05 perché in realtà il modello parte da questa soglia) per entrambi i modelli.
Questo probabilmente è dovuto al fatto che nel ER model non sono presenti hubs, mentre nel Barabasi ci sono ma essendo la selezione iniziale dei nodi

infetti randomica, è molto poco probabile che la diffusione parta da un set di nodi contenente un hub. Nel caso del Barabasi, la diffusione a partire degli hubs porta a un $\Phi = 1$ a differenza del nostro grafo. Sempre $\Phi = 1$ per una diffusione basata sul grado del nodo: la diffusione avviene più rapidamente rispetto al nostro grafo (probabilmente perché il Barabasi è indiretto).

ESERCIZIO 2: LINK PREDICTION

Per affrontare il problema della Link Prediction, sono stati rimossi randomicamente dal network il 20% dei link presenti, poco più di 20mila, lasciando invariato il numero di nodi. L'obiettivo è stato utilizzare diversi approcci non supervisionati per la previsione di nuovi link nel grafo, escludendo quelli presenti (l'80% di quelli originali), per poi confrontare i risultati con l'insieme dei link rimossi.

Sono state utilizzate diverse *neighborhood measures* (Common Neighbours, Adamic Adar e Preferential Attachment) ed una *Path-based measure* (Katz). Si è cercato di eseguire la Link Prediction anche con la misura Sim Rank, ma l'algoritmo ha richiesto troppa RAM per poter essere completato, anche andando a ridurre le dimensioni del network.

Dopo aver calcolato i *predicted link* con ognuno dei diversi approcci, sono stati presi in considerazione solo gli n archi con punteggio più alto, dove n è uguale al numero di link inizialmente rimossi dal network (il 20%). Questo perché gli algoritmi di Link Prediction assegnano un punteggio a tutti i possibili archi non ancora presenti nel network, per cui quelli con punteggio più alto sono quelli che, secondo l'algoritmo, appariranno con più probabilità.

Quindi gli n link previsti con punteggio più alto sono stati confrontati con gli n link rimossi dal network, per individuare quanti fossero presenti in entrambi gli insiemi e calcolare di conseguenza l'accuracy dei predittori.

- Common Neighbours: ha prodotto una accuracy del 24,22%. Anche dal grafico precision/recall seguente si può osservare come i risultati non siano soddisfacenti, poiché la curva ha un picco iniziale, ma poi scende rapidamente a valori di precision molto bassi.

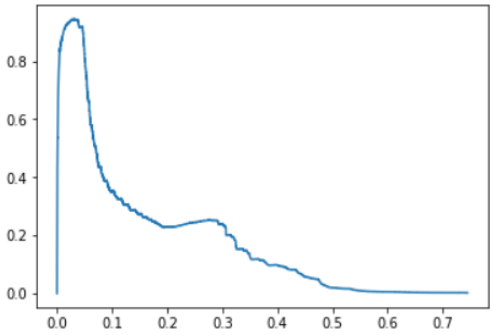

Figura 9: Grafico Precision/Recall Common Neighbours.

- Jaccard: ha restituito il risultato peggiore, lo 0,009% di accuracy; infatti dal grafico è possibile notare che i valori di precision sull'asse y non superano lo 0,01.

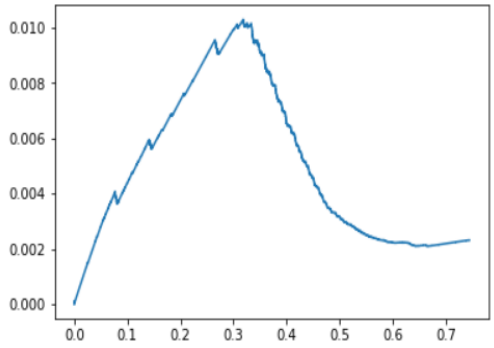

Figura 10: Grafico Precision/Recall Jaccard.

- Adamic Adar: ha prodotto una accuracy del 24,32%, ed il suo grafico precision/recall è molto simile a quello ottenuto con la misura Common Neighbours.

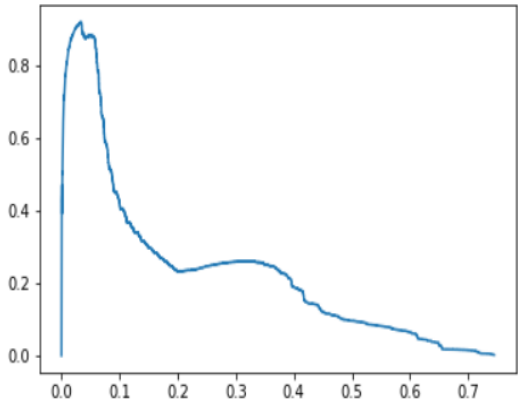

Figura 11: Grafico Precision/Recall Adamic Adar.

- Katz: eseguendo l'algoritmo con la misura Katz sul network, si avevano problemi di memoria per la sua computazione. Lo si è quindi avviato su una versione ridotta del grafo, ottenuta eliminando la metà dei nodi, scegliendo ognuno di essi casualmente. I risultati sono rimasti molto simili a quelli ottenuti con Common Neighbours e con Adamic Adar, sia osservando il grafico, sia relativamente alla accuracy, che è stata del 24,19%.

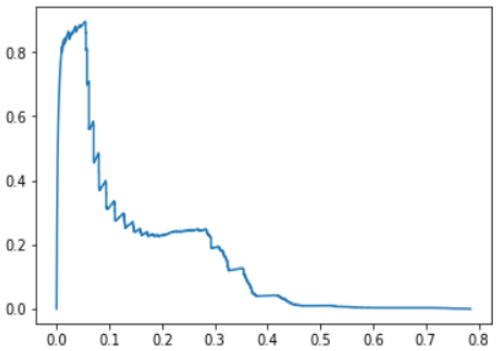

Figura 12: Grafico Precision/Recall Katz.

Nessuno dei quattro approcci ha mostrato risultati particolarmente soddisfacenti comparando i ~20000 link previsti con punteggio maggiore con i ~20000 rimossi dal network: nessuno di essi ha superato una accuracy del 24%, ed il peggiore, Jaccard, ha raggiunto addirittura meno dello 0,01% dei link previsti correttamente.
Osservando l'andamento delle curve nei grafici, abbiamo approfondito il picco iniziale di precision che appare in tutte le curve (tranne in Jaccard). Calcolando il numero di previsioni corrette su un numero inferiore di link, vale a dire quelli con score più alto, si è notato che la accuracy è estremamente più elevata: considerando solo i 100 *top-link*, con Common Neighbours ne sono stati previsti correttamente 84, con Adamic Adar 68 e con Katz 91; con Jaccard 0, come era prevedibile osservando il suo grafico.
Più si incrementa il numero di *top-link* considerati, minore risulta l'accuracy.

In conclusione, la Link Prediction si è dimostrata adatta nel prevedere correttamente un numero ridotto di link, ovvero quelli a cui ha assegnato uno score maggiore.

ESERCIZIO 3: PROBLEMA APERTO

Nell'applicazione del problema aperto si è cercato di portare argomenti che risultano più affini ad un'analisi di network legate al web. L'idea è stata quella di implementare, sempre mediante Networkx, alcuni **algoritmi di rimozione dei nodi** per poi comparare i risultati, sia tra i nuovi metodi proposti da noi, sia tra i metodi affrontati durante il corso. Dal punto di vista dei risultati è stato preso in considerazione il variare della grandezza della componente più grande in maniera tale da analizzare effettivamente la resilienza del network, il task è stato effettuato sul grafo considerato indiretto.

Il **primo step** di questo esercizio è stato quello di computare i metodi già affrontati nella parte teorica:
Il primo metodo (**Target Remove**) è un algoritmo di rimozione che, partendo dai nodi con degree maggiore, rimuove gli hubs presenti fino ad arrivare ai nodi con grado 1. Come visto nel corso, questo algoritmo tende a separare quasi subito il grafo, questo comportamento è dato dal fatto che i primi elementi che vengono rimossi sono quegli elementi che hanno grado massimo e quindi diretti responsabili del comportamento molto simile allo scale free network.

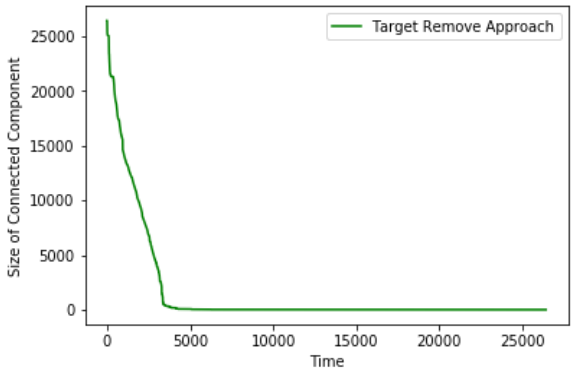

Figura 13: Variazione del Connected Component con Target Remove.

Il crollo del network nel nostro caso avviene verso l'iterazione numero 3000.

Il secondo algoritmo (**Random Remove**) è un algoritmo che rimuove in maniera randomica, data una determinata probabilità, i nodi del grafo. Come visto anche a lezione questo algoritmo, almeno per il tipo di network preso in esame in questo progetto, non è molto efficiente poiché è cieco nei confronti del grado dei nodi. La probabilità che vengano rimossi nodi con grado minimo è esponenzialmente più alta rispetto alla rimozione dei nodi con grado massimo (responsabili del comportamento secondo scale free network).

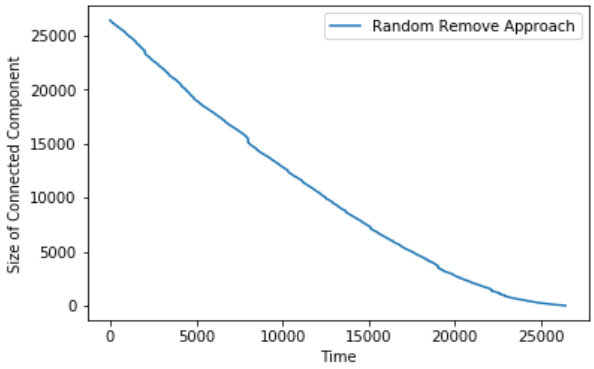

Figura 14: Variazione del Connected Component con Random Remove.

Come è possibile vedere dal grafico, il network crolla al minimo valore di grandezza del componente solo quando il numero di nodi rimosso arriva circa alla totalità dei nodi presente nel grafo.

Il **secondo step** è stato quello di implementare metodi di rimozione ideati secondo alcuni principi:
Il primo algoritmo implementato tiene conto più possibile del fatto che più un sito è importante più è alta la probabilità che tale sito abbia delle **repliche** (cosa che aumenta in maniera esponenziale la difficoltà a rimuovere tale nodo). In primis si è calcolato il grado del nodo come metrica di importanza all'interno del network, successivamente è stata fatta una mappatura, mediante interpolation, per riportare i gradi (da 1 a 3113) in un range da 0 a 100. Questo range calcolato rappresenta una percentuale di difficoltà nella rimozione del nodo (più la percentuale è bassa, meno il nodo è importante e minore sarà la probabilità che tale nodo abbia delle repliche). Quello che è stato implementato è un algoritmo che, partendo da due parametri che rappresentano il limite superiore e il limite inferiore rispetto la percentuale di presenza di repliche, rimuove tutti i nodi presenti in quel range.
Nel nostro caso si è cercato un valore che fungesse da tradeoff tra percentuale di probabilità di avere repliche e grado; si è optato per un valore dei parametri compreso tra 1% e 9% (sono stati rimossi quindi tutti quei nodi che avevano grado tra 274 e 1). Il risultato è il seguente.

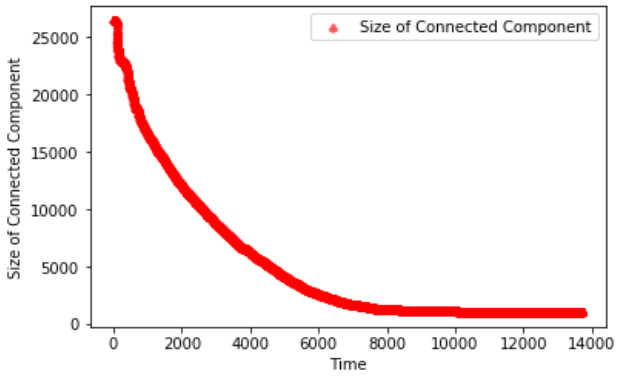

Figura 15: Variazione del Connected Component con attacco considerando la presenza di repliche.

Come possiamo vedere dal grafico il comportamento risulta molto simile al target remove ma, in realtà, il decrescere della grandezza della componente più grande è meno evidente in questo metodo rispetto al target. Molto probabilmente questo è dato dal fatto che in questo approccio non vengono rimossi i veri hub (quelli con probabilità altissima di avere delle repliche) e quindi la componente più grande riesce a rimanere più a lungo unita. Lo si può notare anche dal fatto che nelle iterazioni finali la componente connessa ha dimensione relativamente alta (1000 nodi). Il crollo arriva al suo valore più basso verso l'iterazione 7000.

Il secondo algoritmo implementato è un algoritmo che tiene conto del possibile **tempo di eliminazione dei nodi** (approssimato con il degree del nodo). In questo approccio più un nodo ha grado elevato e più, almeno in teoria, dovrebbe essere difficile rimuoverlo.

L'idea dietro questo algoritmo è quello di eliminare i nodi più centrali del network non tralasciando però il tempo di rimozione di tale nodo, si cerca di identificare quei nodi che sono molto importanti per il network ma che siano i più semplici da eliminare. Quello che viene fatto prima di tutto è computare le misure di centralità per ogni nodo (per completezza sono state fatte sia le computazioni della betweenness centrality che della closeness centrality poiché sono le due metriche affrontate durante il corso). Successivamente viene calcolato uno score per ogni nodo, dopo aver standardizzato i valori di centralità e degree mediante interpolazione, in cui a numeratore abbiamo la centralità e a denominatore il tempo di rimozione del nodo.
Successivamente sono stati rimossi i nodi partendo da tale score, il risultato è il seguente:

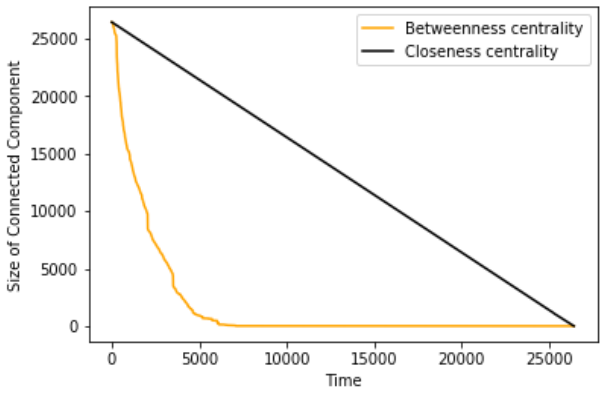

Figura 16: Variazione della componente connessa sfruttando l'ausilio di misure di centralità.

Sfruttando la betweenness centrality il grafo diventa sconnesso verso l'iterazione 7000 (quindi effettivamente la logica è molto simile al target remove, con la differenza che nel target remove la caduta del grafo avviene in molto meno tempo. Questo comportamento può essere spiegato dal fatto che ovviamente i nodi più centrali sono anche quelli con degree maggiore, ma alcuni nodi con minor degree possono superare quelli con grado maggiore scoraggiando la loro rimozione in fatto di tempo da qui la differenza di 2500 iterazioni tra i due approcci). Dall'altra parte (in nero) abbiamo un comportamento anomalo sfruttando la closeness; quello che salta subito all'occhio è il fatto che il comportamento della grandezza della componente più grande viene formalizzato mediante una retta (quello che effettivamente avviene è che i nodi vengono rimossi dal grado più piccolo al grado più grande). Abbastanza scettici su questo risultato abbiamo effettuato dei check sui dati e abbiamo scoperto che il range tra nodo più centrale e nodo meno centrale in termini di metriche è molto più ampio nella betweenness rispetto alla closeness.

	Betweenness	Closeness
count	26426.000000	26426.000000
mean	0.000126	0.236693
std	0.003041	0.037764
min	0.000000	0.112588
25%	0.000000	0.211128
50%	0.000000	0.237774
75%	0.000001	0.242765
max	0.335929	0.417760

Figura 17: Confronto Betweenness Centrality con Closeness Centrality.

Come possiamo vedere nella descrizione delle due metriche è evidente la differenza tra le due, è sufficiente guardare i percentili per evidenziare quanto il valore della betweenness è più considerato (almeno nella formula da noi utilizzata) rispetto a quello della closeness. Appunto per questo, rispetto alla betweenness la misura di centralità ha valore, anche se in alcuni casi viene influenzata dal tempo di riduzione del nodo a causa del grande range tra i nodi meno centrali e più centrali. Purtroppo per la closeness il range tra i valori meno centrali e più centrali è talmente ridotto che il tempo di rimozione del nodo fa da sovrano e quello che otteniamo è un andamento addirittura peggiore del random attack.

Come ultima soluzione (terza) è stato ideato un tipo di attacco in cui partendo dal nodo con degree maggiore viene simulata una reazione a catena dove si assume che ogni nodo, prima di essere rimosso, porta alla rimozione anche i suoi vicini.
L'algoritmo in primis parte da un nodo dato, nel nostro caso si è optato per quello con degree maggiore poiché solitamente durante degli attacchi è quello più mirato, e da questo nodo crea degli strati di distanza. Si parte dal nodo centrale (a cui viene assegnato il tag 0) e si computano tutti i suoi vicini a

cui vengono assegnati tag 1 (indica la distanza dal nodo di partenza), dai nodi con distanza 1 vengono computati a loro volta i vicini (escludendo quelli già incontrati nelle esplorazioni precedenti) e assegnati i tag successivi. L'idea è quindi avere una lista di nodi con associato il tag di distanza. La rimozione, infine, viene fatta partendo dalla distanza 0 fino ad arrivare in maniera incrementale alla distanza massima. Il risultato è il seguente:

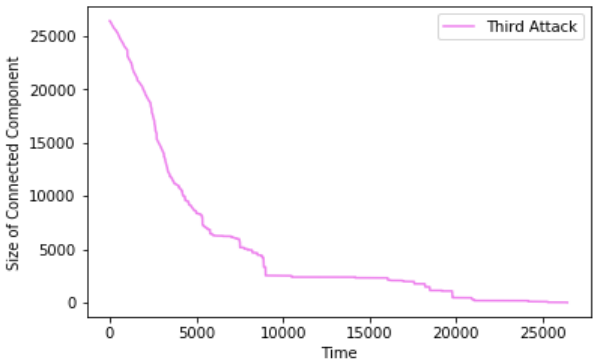

Figura 18: Variazione del Connected Component con attacco a propagazione.

Quello che si può notare è che in una prima fase questa tipologia di attacco si comporta molto vicino al primo algoritmo ideato (ma anche in maniera molto simile al target attack).
Nella seconda parte però la rimozione subisce un blocco in cui, molto probabilmente, non vengono identificati ad una determinata distanza gli hub e quindi la separazione totale della componente più grande avviene solo alla iterazione 20.000.

Tutti gli algoritmi, sia quelli portati da noi, sia quelli già affrontati a lezione, sono stati confrontati nello stesso grafico.

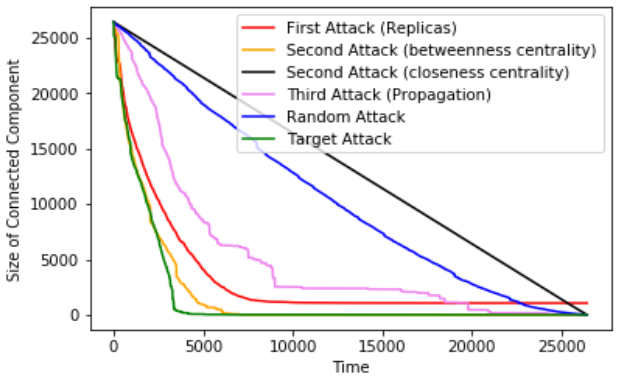

Figura 19: Grafico di confronto dei vari attacchi proposti.

Cercando l'algoritmo più efficiente (con l'obiettivo di far collassare la componente più grande), ovviamente la scelta deve ricadere su un target attack, ma, sia considerare l'opzione della propagazione, sia considerare il secondo attacco sfruttando la betweenness centrality possono rappresentare un ottimo trade off costo efficacia. Se invece si vuole il più possibile rimanere fedele alla realtà (ovvero che è più difficile rimuovere nodi con altissimo degree) la soluzione migliore è rappresentata in rosso dove, però, non rimuovendo la totalità dei nodi abbiamo un appiattimento sul valore 1054.