



Università degli Studi di Bari - “Aldo Moro”

DIPARTIMENTO DI INFORMATICA

Corso di Laurea in

Informatica e Tecnologie per la Produzione del Software

TESI DI LAUREA
IN
PROGRAMMAZIONE II

**Url2Vec:
Clustering di pagine in un grafo Web**

Relatore:

Chiaramo Prof. Michelangelo Ceci

Correlatore:

Dott.ssa Pasqua Fabiana Lanotte

Laureando:

Christopher Piemonte

Matricola 587662

Ai miei genitori ...

Indice

1	Informazioni latenti nel Web	1
1.1	Data Mining nel Web	2
1.2	Web Structure mining	5
1.3	Rappresentazioni vettoriali di pagine Web	7
1.3.1	Apprendimento dalla struttura	8
1.3.2	Word2vec	11
1.4	Clustering	14
1.4.1	Graph Clustering	16
1.4.2	Vector Clustering	20
1.4.3	Algoritmi utilizzati	24
1.5	Data Visualization	26
2	Il Sistema Url2vec	28
2.0.1	NLP nel Web: URL Embedding	29
2.1	Web Crawling per l'estrazione dei dati	31
2.1.1	Proprietà del web crawler	33
2.1.2	Estrazione delle liste	35
2.2	Costruzione del Dataset	36
2.2.1	Random Walk	37
2.2.2	Generazione delle sequenze	40
2.2.3	Esempio di Dataset	42
2.3	Web page Clustering	45
3	Stato dell'Arte	47

3.1	Liste da sorgenti dati strutturate	49
3.2	Teoria dei Random Walk	50
4	Sperimentazione	52
4.1	Dataset	53
4.2	Metriche	54
4.3	Configurazioni	58
4.3.1	Community Detection	58
4.3.2	URL Embedding	59
4.3.3	Text Mining	62
4.3.4	Embedding e Text Mining	64
4.4	Analisi dei risultati	65
4.4.1	Community Detection	76
4.4.2	URL Embedding	79
4.4.3	Text Mining	84
4.4.4	Embedding e Text Mining	90
4.4.5	Analisi dei risultati	93
5	Conclusioni e sviluppi futuri	96

Elenco delle tabelle

4.1	My caption	59
4.2	Risultati sperimentazione del sito <code>cs.illinois.edu</code>	67
4.3	Risultati sperimentazione del sito <code>cs.stanford.edu</code>	69
4.4	Risultati sperimentazione del grafo del sito <code>eecs.mit.edu</code> . .	71
4.5	Risultati sperimentazione del sito <code>cs.princeton.edu</code>	73
4.6	Risultati sperimentazione del sito <code>cs.ox.ac.uk</code>	75
4.7	Risultati sperimentazione di partizionamento del grafo del sito <code>cs.illinois.edu</code>	77
4.8	Risultati sperimentazione di partizionamento del grafo del sito <code>cs.stanford.edu</code>	77
4.9	Risultati sperimentazione di partizionamento del grafo del sito <code>eecs.mit.edu</code>	78
4.10	Risultati sperimentazione di partizionamento del grafo del sito <code>cs.princeton.edu</code>	78
4.11	Risultati sperimentazione di partizionamento del grafo del sito <code>cs.ox.ac.uk</code>	79
4.12	Risultati sperimentazione di URL embedding del sito <code>cs.illinois.edu</code>	80
4.13	Risultati sperimentazione di URL embedding del sito <code>cs.stanford.edu</code>	81
4.14	Risultati sperimentazione di URL embedding del sito <code>eecs.mit.edu</code>	82
4.15	Risultati sperimentazione di URL embedding del sito <code>cs.princeton.edu</code>	83
4.16	Risultati sperimentazione di URL embedding del sito <code>cs.ox.ac.uk</code>	84
4.17	Risultati sperimentazione di Text Mining del sito <code>cs.illinois.edu</code>	86
4.18	Risultati sperimentazione di Text Mining del sito <code>cs.stanford.edu</code>	87
4.19	Risultati sperimentazione di Text Mining del sito <code>eecs.mit.edu</code>	88

4.20	Risultati sperimentazione di Text Mining del sito <code>cs.princeton.edu</code>	89
4.21	Risultati sperimentazione di Text Mining del sito <code>cs.ox.ac.uk</code>	90
4.22	Risultati sperimentazione di Text-Embedding del sito <code>cs.illinois.edu</code>	91
4.23	Risultati sperimentazione di Text-Embedding del sito <code>cs.stanford.edu</code>	91
4.24	Risultati sperimentazione di Text-Embedding del sito <code>eeecs.mit.edu</code>	92
4.25	Risultati sperimentazione di Text-Embedding del sito <code>cs.princeton.edu</code>	92
4.26	Risultati sperimentazione di Text-Embedding del sito <code>cs.ox.ac.uk</code>	93
4.27	Risultati sperimentazione raggruppati per algoritmo K-Means sul sito <code>cs.illinois.edu</code>	93
4.28	Risultati sperimentazione raggruppati per algoritmo K-Means sul sito <code>ecs.mit.edu</code>	94
4.29	Risultati sperimentazione raggruppati per algoritmo K-Means sul sito <code>ecs.mit.edu</code>	94

Elenco delle figure

1.1	Parole con vettori simili	9
1.2	Alcuni esempi di analogie	10
1.3	Visualizzazione con t-SNE di un Word Embedding bilingua. .	11
1.4	Vettori corrispondenti alle parole	12
1.5	Vettori corrispondenti alle frasi	12
1.6	Matrice di similarità	13
1.7	Assegna uno score utilizzando Pagerank	14
1.8	Differenze fra i vari tipi di funzioni distanza	16
1.9	Betweenness centrality score	18
1.10	Modularity score	19
1.11	Diagramma di Voronoi	21
1.12	Matrice termini-documenti. Ogni riga rappresenta un singolo termine ed ogni colonna rappresenta un singolo documento .	22
1.13	La visualizzazione dei dati è un passo fondamentale nell’analisi dei dati.	27
1.14	Riduzione da tre a due dimensioni.	27
2.1	Architettura di un web crawler	31
2.2	Differenze tra ricerca in ampiezza e ricerca in profondità . .	34
2.3	Rappresentazione visuale di 8 random walk monodimensionali.	37
2.4	Rappresentazione visuale di un random walk di 25.000 passi su due dimensioni.	38
2.5	Random walk sul grafo.	42

4.1	Rappresentazione del sito <code>cs.illinois.edu</code> , clusterizzato con K-Means.	60
4.2	Rappresentazione del sito <code>cs.illinois.edu</code> , clusterizzato con K-Means.	68
4.3	Rappresentazione del sito <code>cs.stanford.edu</code> , clusterizzato con K-Means.	70
4.4	Rappresentazione del sito <code>eecs.mit.edu</code> , clusterizzato con Fast-greedy.	72
4.5	Rappresentazione del sito <code>eecs.mit.edu</code> , clusterizzato con Fast-greedy.	74
4.6	Grafo del sito <code>cs.illinois.edu</code> etichettato manualmente.	76

Introduzione

La principale caratteristica dell’Era dell’Informazione è rappresentata dalla possibilità di generare, memorizzare, trasmettere e processare enormi quantità di dati in modo rapido ed economico.

La disponibilità di una simile quantità di dati, elaborabili automaticamente, ha consentito un forte incremento del processo di generazione e diffusione di conoscenza, utilizzabile per migliorare processi decisionali. Ad oggi, tuttavia, tali risorse non sono appieno sfruttate in tutti i campi e il loro valore potenziale riserva ancora numerose sorprese.

Questo problema è particolarmente sentito nel contesto Web. Il Web infatti può essere considerato la più grande, eterogenea e dinamica sorgente informativa pubblicamente accessibile. Tali caratteristiche rendono il processo di analisi dei dati e estrazione di nuova conoscenza un task altamente complesso e apre nuove sfide e frontiere per l’Informatica.

Un’importante sfida è rappresentata dal problema di organizzare i contenuti e la struttura dei documenti web. In particolare diversi lavori si sono concentrati sul problema del Web Clustering, ossia il processo di raggruppare pagine web in *cluster* cosicché oggetti simili possano essere raggruppati nella stessa classe e oggetti dissimili possano essere raggruppati in classi differenti. Gli obiettivi di questo processo possono essere molteplici: migliorare l’accessibilità alle informazioni e il ritrovamento delle stesse, comprendere i comportamenti di navigazione degli utenti, comprendere come le informazioni si distribuiscono su più pagine web, etc.

In quest'ottica nasce Url2vec, un sistema per il clustering di pagine web che utilizza il contenuto testuale delle pagine web e la struttura ad hyperlink del sito a cui le pagine da raggruppare appartengono, per estrarre cluster di pagine dello stesso tipo semantico (per esempio pagine web di professori, corsi, prodotti, etc.).

Differentemente dagli algoritmi di clustering di pagine web esistenti in letteratura, Url2Vec non considera un sito web come una collezione di documenti testuali indipendenti tra loro, ma cerca di combinare informazioni relative al contenuto con informazioni strutturali, in modo che due pagine web vengano considerate simili se caratterizzate da una simile distribuzione di termini e abbiano una simile correlazione nascosta all'interno dei cammini percorribili nel sito web.

Le motivazioni alla base dell'implementazione di Url2vec sono state guidate dal voler sfruttare conoscenza già immagazzinata nella risoluzione di problemi specifici in contesti differenti per il quale erano stati ideati, ricavando un trasferimento della conoscenza. Infatti Url2Vec combina algoritmi tipici dell'area del Natural Language Processing con quelli della Teoria dei Grafi al fine di utilizzare in modo innovativo algoritmi estensi nel contesto Web.

Si definisce di seguito la struttura di questo lavoro di tesi.

Nel capitolo 1 ci si occuperà di descrivere lo stato attuale, elencando le metodologie utilizzate, e di analizzare nel dettaglio le diverse problematiche da affrontare durante l'analisi dei dati. Nel capitolo 2 saranno presentati gli obiettivi principali che la metodologia presentata ed il sistema realizzato hanno seguito, descrivendo nel dettaglio le diverse tecniche utilizzate per la realizzazione delle fasi necessarie all'individuazione dei pattern latenti nella struttura del web. Nel capitolo 3 si parlerà della frontiera attuale dell'Informatica in tali campi confrontando similitudini e spunti di riflessione. Nel capitolo 4 si descriverà la sperimentazione effettuata, completa di tabelle, grafici e commenti che evidenziano punti di forza e di debolezza individuati per ciascuna delle tecniche utilizzate per le diverse fasi eseguite dal sistema. Soffermandosi sulle novità introdotte con le metodologie presentate e cercando

di confrontarle con quelle consolidate. Infine nel capitolo 5 si parlerà dei possibili miglioramenti alle tecniche ed alle metodologie proposte.

Capitolo 1

Informazioni latenti nel Web

Il progressivo aumento della dimensione del Web e le informazioni in esso contenute fanno di esso la più grande sorgente informativa pubblicamente accessibile. Il Web infatti contiene qualsiasi tipo di informazione in qualsiasi tipo di formato. La sua grande eterogeneità rende l'estrazione e il reperimento delle informazioni un task altamente complesso.

Sebbene a prima vista il Web possa sembrare un insieme disordinato e non strutturato di informazioni distribuite su molteplici pagine web, in realtà è possibile estrarre molteplici correlazioni nascoste tra informazioni all'interno della stessa pagina web e informazioni tra pagine web connesse tramite hyperlink.

Riuscire ad estrarre tali correlazioni e pattern, analizzando la struttura ad hyperlink di cui il Web si compone, permetterebbe di migliorare diverse applicazioni esistenti.

Il problema rimane l'individuazione del procedimento adatto al compito prefissato. Tecniche e metodologie per l'estrazione di conoscenza da grandi quantità di dati sono già state sviluppate nell'area del Data Mining. Trasferire questo sapere nel Web, tuttavia, può non essere semplice e immediato, date le caratteristiche che lo contraddistinguono. Di seguito sarà introdotto il contesto in cui si sviluppa la tesi, le varie aree in cui si colloca e da cui

atteggi le metodologie ed il bagaglio di conoscenza utile alla sintesi di nuovi algoritmi di estrazione di conoscenza dal Web.

1.1 Data Mining nel Web

Il Data Mining è l'insieme di tecniche e metodologie che hanno per oggetto estrazione di informazione utile, di un sapere o di una conoscenza a partire da grandi quantità di dati.

Il concetto di informazione è strettamente legato al contesto in cui si esegue un task di apprendimento, in altre parole un dato può essere interessante o trascurabile a seconda del tipo di applicazione in cui si vuole operare. La fase di estrazione di informazione dai dati per renderla direttamente utilizzabile può variare enormemente dal dominio applicativo. Le differenze sono tali da dover suddividere tali procedimenti in aree diverse, dipendenti dal tipo di dati da cui parte il processo di estrazione. Principalmente le grandi moli di dati possono variare da grandi collezioni di documenti, database, pagine Web ecc, differenziandosi molto nella struttura e nel contenuto,.

Il **Web Mining** è l'applicazione delle tecniche di Data Mining per la scoperta e l'estrazione di conoscenza o di pattern dal World Wide Web.

Le proprietà che caratterizzano il Web e che lo differenziano da altre sorgenti di dati sono:

- **Dimensione:**

Il Web è il primo mezzo di informazione dove il numero di produttori di informazioni è uguale al numero dei consumatori. La quantità dei dati da processare può essere più grande di svariati ordini di grandezza rispetto a tradizionali task di Data Mining. Diventa necessario l'utilizzo di tecniche scalabili, ossia la capacità di un sistema di "crescere" ed essere utilizzabile in funzione della crescita dei dati. Nel Data Mining tradizionale, processare un milione di record può essere considerato

sopra la media, mentre nel Web Mining dieci milioni di pagine possono non essere abbastanza.

- **Dinamicità:** Ogni secondo sono create, distrutte e modificate migliaia di pagine Web. Questo rende il Web una rete di informazioni dinamica, dove la struttura e il contenuto dell'informazione cambiano frequentemente. Monitorare questi cambiamenti rimane un problema importante per molte applicazioni.
- **Eterogeneità:** L'eterogeneità del Web può dipendere sia dal formato delle pagine che dalla contenuto testuale. Nel primo caso, l'eterogeneità è dovuta al fatto che non esiste uno standard di formato, dividendo le pagine Web in tre principali categorie: *i)* pagine non strutturate *ii)* pagine strutturate *iii)* pagine semi-strutturate.

Le pagine *non strutturate*, anche chiamate *free-text pages*, sono scritte in linguaggio naturale. Su queste possono essere applicate tecniche con un certo grado di affidabilità, denotate dall'arbitrarietà nella valutazione dei risultati.

Le pagine *strutturate* sono normalmente ottenute da sorgenti di dati strutturati (e.g. database). Le tecniche di estrazione sono applicate usando l'individuazione di regole sintattiche.

Le pagine *semi-strutturate* si posizionano al centro delle precedenti. Possiedono infatti sezioni strutturate insieme a testo libero, mostrando un certo livello di struttura nascosto nel testo. L'estrazione può avvenire cercando pattern nei tag HTML, utilizzando i metadati o identificando solo l'informazione strutturata.

In questo caso l'eterogeneità è dovuta al fatto che i contenuti Web sono creati da milioni di persone aventi differente cultura, abilità e linguaggio. Questo significa che le pagine potrebbero contenere la stessa informazione, ma presentata in maniera completamente diversa.

- **Connessione:** Il Web è generalmente rappresentato come una rete di informazioni dove i nodi sono le pagine e gli archi sono gli hyperlink. Gli hyperlink fra le pagine di uno stesso sito e quelli fra le pagine di siti diversi, hanno caratteristiche e funzionalità diverse. All'interno del sito

servono ad organizzare i contenuti, mentre fra siti diversi sono usati per trasportare autorità alle pagine di destinazione, che avranno prevalentemente contenuti simili o inerenti a quelli della pagina di partenza. In questo caso significa che come persone ci fidiamo del contenuto di queste pagine.

- **Rumore:** Differentemente da altri mezzi di informazione, la pubblicazione di contenuti è libera e non richiede approvazione. Questo contribuisce all'aumentare del volume e della diversità dell'informazione, ma anche alla creazione di contenuti ridondanti e poco informativi.
- **Società virtuale:** La vastità e la proliferazione del Web lo rendono di fatti un enorme Social Network, dove le persone possono comunicare e influenzarsi reciprocamente. Infatti non riguarda solo i dati, le informazioni e i servizi, ma anche le interazioni fra le persone, le organizzazioni o i sistemi.

Sulla base di tali caratteristiche, task di Web Mining possono utilizzare tecniche diverse ed essere finalizzate alla scoperta di informazioni differenti. Si possono distinguere principalmente tre categorie:

- *Web Usage Mining*, è l'applicazione di tecniche di Data Mining per la scoperta di pattern e informazioni utili attraverso l'analisi di log immagazzinati dai web server e contenenti click stream. Obiettivo di questo campo è l'apprendimento dell'identità, origine e comportamenti degli utenti che navigano i siti Web al fine di comprendere i loro bisogni e ad offrire loro servizi migliori attraverso una personalizzazione dell'esperienza web.
- *Web Structure Mining*, consiste nell'estrazione di relazioni sconosciute o nascoste tra pagine web attraverso l'analisi della struttura ad hyperlink di un sito web (anche chiamato “grafo web”). Questo task verrà analizzato in dettaglio nella Sezione 1.2.
- *Web Content Mining*, consiste nell'estrazione ed integrazione di informazione utile e precedentemente sconosciuta dal contenuto delle pagine

Web. Ricadono in questo campo due principali tipologie di algoritmi: *i)* algoritmi capaci di raggruppare e classificare pagine web in funzione del loro contenuto testuale o del topic descritto; *ii)* algoritmi per estrarre pattern strutturati contenuti nelle pagine Web (per esempio liste di prodotti commerciali, professori, etc.). Sebbene questi algoritmi possano sembrare molto simili ai più famosi algoritmi di Data Mining e Text Mining, le pagine Web hanno delle peculiarità che non li rendono direttamente applicabili. Un’importante branca del Web Content Mining è rappresentato dal Web Information Extraction, il cui obiettivo è quello di estrarre dati strutturati da pagine web e integrarli in tabelle relazionali. In questo contesto il Web Content Mining può essere visto quindi come reperimento e immagazzinamento di informazioni dal contenuto testuale.

1.2 Web Structure mining

Sono sempre di più le organizzazioni che disseminano informazioni in rete. Estrarre questi dati è utile in molti domini applicativi perché permette di ottenere ed integrare dati da diverse fonti, producendo così servizi migliori, informazioni personalizzate o meta-ricerche.

Tuttavia, differentemente dai documenti testuali tradizionali, il contenuto testuale delle pagine Web è arricchito da hyperlink che dividono l’informazione in molteplici ed interdipendenti pagine Web. Questi hyperlink possono essere usati per identificare le entità provenienti dal mondo reale (e.g. pagine di professori, corsi, prodotti) e le relazioni che intercorrono fra di esse costruendo il grafo del sito ed utilizzando tecniche derivanti dalla *teoria dei grafi*. Inoltre molte pagine Web si presentano come combinazione di testo non strutturato e dati strutturati, tipicamente generati dinamicamente da una sorgente sottostante come un database relazionale. Infatti i documenti Web non sono né strutturati come un database né completamente non-strutturati come do-

cumenti testuali, le tecniche tradizionali di Data Mining o Text Mining non possono essere applicate direttamente.

Nel primo caso, le tecniche di Data Mining si basano sul presupposto che i dati usati per apprendere un modello condividono uno schema comune, avente delle tabelle ben definite con attributi, colonne, tuple e vincoli. Le pagine Web non hanno questo presupposto perché contengono dati eterogenei e gli hyperlink definiscono relazioni il cui significato può variare profondamente. Inoltre le pagine Web sono codificate in HTML che, differentemente da altri linguaggi di markup, è stato progettato solo per la visualizzazione (o *rendering*) dei dati. Per questa ragione, il Web può essere considerato un moderno *legacy system*, in quanto una grande quantità di dati non è facilmente accessibile e direttamente manipolabile.

Nel secondo caso, le tecniche di Text Mining falliscono nell'apprendere modelli accurati perchè: *i*) richiedono collezioni di documenti scritti in modo consistente; *ii*) non sono in grado di gestire informazioni complesse con elementi che possiedono diversi ruoli semantici e che forniscono diverse funzionalità. Infatti differentemente dai documenti testuali, le pagine Web hanno molteplici rappresentazioni che forniscono differenti informazioni, quali la rappresentazione testuale del testo HTML e la rappresentazione visuale renderizzata da un web browser. Algoritmi di Text Mining si concentrano sulla rappresentazione testuale ed ignorano la rappresentazione visuale. Di conseguenza, esiste un forte bisogno nel campo dell'informatica di creare approcci e tecniche che usando informazioni testuali, strutturali e visuali sono capaci di estrarre uno schema da dati strutturati ed allineare i dati di conseguenza.

Il Web Structure Mining può essere diviso in due tipi:

- Estrazione di dati strutturati tra pagine web attraverso l'analisi del sito web. In questo caso un sito web è rappresentato come un grafo $G = (V, E)$ dove V è l'insieme delle pagine web e E è l'insieme degli hyperlink.
- Estrazione di dati strutturati contenuti in una pagina web, analizzan-

done la struttura ad albero basata su tag HTML ed XML.

Tra i più importanti algoritmi di web structure mining troviamo Page Rank [19] e HITS [13], i quali sfruttano la struttura ad hyperlink del Web per assegnare un rank alle pagine, ovvero per restituirle in ordine di importanza relativamente ad una determinata query.

1.3 Rappresentazioni vettoriali di pagine Web

Reperire informazione dalle pagine Web, quando queste si presentano in forma non-strutturata o semi-strutturata, necessita di passaggi preliminari per rendere i dati processabili dai tradizionali algoritmi di Machine Learning o Data Mining.

La maggior parte delle soluzioni attuali trasformano il contenuto testuale delle pagine Web in uno spazio vettoriale [30]. Questo è motivato dal fatto che task di apprendimento richiedono input che sono matematicamente e computazionalmente convenienti da elaborare, considerando solo un sottoinsieme significativo dei dati in modo da astrarre ed eliminare informazioni ritenute non pertinenti al risultato finale.

Modelli basati su spazi vettoriali sono fondamentali per task che coinvolgono il calcolo della similarità in quanto oggetti simili sono caratterizzati da rappresentazioni vettoriali simili. Il modello vettoriale *termini-documenti* è uno dei più utilizzati modelli di rappresentazione di documenti testuali nel contesto del Text Mining. In tale modello il valore dell'elemento i -esimo in un vettore documento rappresenta il numero di volte che il termine i compare nel documento stesso.

Questi modelli si basano sull'assunzione di indipendenza tra i termini all'interno di un documento e sull'assunzione di indipendenza tra i documenti stessi. Le pagine Web violano, come tutti i documenti testuali, la prima assunzione, inoltre i collegamenti ipertestuali definendo relazioni di interdipendenza tra le pagine stesse comportano la violazione della seconda. Può

accadere che pagine prive di testo, completamente vuote (e.g. con contenuti grafici) o ridotte al solo template base del sito, vengano raggruppate nello stesso cluster. Inoltre pagine relative dello stesso tipo (e.g. pagine di docenti) potrebbero essere state create da persone diverse e presentare una distribuzione dei termini notevolmente differente.

Di conseguenza tener conto di informazioni relative alla struttura del sito Web può arricchire di informazioni utili, linearmente indipendenti a quelle derivanti dal contenuto testuale.

1.3.1 Apprendimento dalla struttura

Gli algoritmi di apprendimento lavorano tanto bene quanto meglio i vettori ricavati rappresentino bene i dati di partenza. È necessario quindi estrarre rappresentazioni utili dai dati grezzi. Esistono molti modi per ricavare questa correlazione, come reti neurali, matrici di co-occorrenza, dimensionality reduction.

L’alternativa proposta in questa tesi consiste nell’utilizzare tecniche di Word Embedding per apprendere rappresentazioni vettoriali dei vertici all’interno del grafo, utilizzati congiuntamente ai vettori del contenuto delle pagine Web. Il Word Embedding è il nome di un insieme di tecniche per il language modeling e per il Feature Learning nel campo del Natural Language Processing (NLP)[3], utilizzate in collezioni di documenti, dove ad ogni parola viene associato un vettore anche chiamato *Feature Vector*.

Il Word Embedding può essere visto come una funzione parametrizzata

$$W : words \rightarrow \mathbb{R}^k \quad (1.1)$$

che associa una parola in un dato linguaggio ad un vettore multidimensionale. Un esempio potrebbe essere:

$$W("cat") = (0.2, -0.4, 0.7, \dots) \quad (1.2)$$

A parole simili corrispondono un vettori simili. Se si cambia una parola con un sinonimo, la validità della frase in esame non cambia (e.g. molti cantano bene → tanti cantano bene). Questo permette di generalizzare da una frase ad una classe di frasi simili o di capire se una frase è valida, ovvero se è formulata correttamente.

Questo non significa solo poter scambiare una parola con un sinonimo, ma anche di cambiare una parola con una altra in una classe simile (eg. il muro è rosso → il muro è blu) [7]. Questo può essere appreso analizzando il contesto della parola da analizzare. Ad esempio ci saranno molti casi in cui sono state osservate frasi valide di questo tipo, quindi cambiando la parola “rosso” con la parola “blu” porterebbe alla creazione una frase ugualmente valida.

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Figura 1.1: Parole con vettori simili

Da questo potrebbe sembrare necessario osservare esempi relativi ad ogni parola per permetterci di generalizzarla. Comprendi tutte le parole che hai già visto, ma non hai già visto tutte le frasi che riesci a capire. Questo è l’approccio delle reti neurali.

Analogie Il Word Embedding mostra un'altra proprietà interessante anche se molto controversa: le analogie. Le analogie tra parole sembrano essere nascoste nella differenza dei loro rispettivi vettori [18].

$$W("woman") - W("man") \simeq W("aunt") - W("uncle") \quad (1.3)$$

Da questo si evince che c'è una correlazione tra delle parole e le rispettive forme del genere opposto in quanto appariranno in contesti simili, differenti solo per alcuni dettagli come pronomi o articoli. Stessa cosa per tra singolare e plurale [18]. Queste proprietà possono essere considerate effetti collaterali.

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Figura 1.2: Alcuni esempi di analogie

Non si è cercato di far apprendere il modello in modo da avere parole simili vicine fra loro. Questo sembra essere un punto di forza delle reti neurali nell'apprendere features che rappresentano bene i dati in modo automatico. Invece di apprendere un rappresentazione dei dati specifica ed usarla per diversi task, è possibile apprendere un metodo per associare diversi tipi di dati in una singola rappresentazione. Queste tecniche sono note come Transfer Learning, metodi per applicare la conoscenza già appresa in contesti simili.

Un esempio può essere il Word Embedding di parole linguaggi diversi. Dato che parole simili saranno associate a vettori simili, parole con significato simile in una lingua e nell'altra finiranno vicine tra loro, così come i loro sinonimi. È possibile notare che anche parole di cui non si conosceva la traduzione o che avessero significati simili, sono finite vicine tra loro [33].

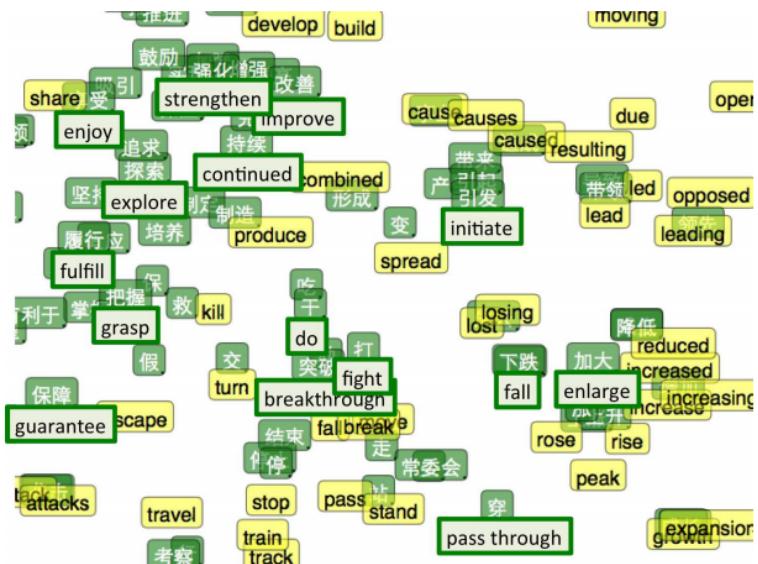


Figura 1.3: Visualizzazione con t-SNE di un Word Embedding bilingua.

1.3.2 Word2vec

Un algoritmo molto famoso di word embedding è il recente word2vec. L'algoritmo usa i documenti per far apprendere una rete neurale, massimizzando la probabilità condizionata del contesto data una parola, applicando il modello appreso ad ogni parola per ricavare il vettore corrispondente e calcolando il vettore della frase facendo la media dei vettori delle parole, costruisce la matrice di similarità delle frasi ed usa PageRank per classificare le frasi nel grafo.

$$\arg \max_{\theta} \prod_{(w,c) \in D} p(c|w; \theta) \quad (1.4)$$

L'obiettivo è di ottimizzare il parametro (θ) massimizzando la probabilità condizionata del contesto (c) data la parola (w). D è l'insieme di tutte le coppie (w, c) . Per esempio: "ho mangiato un _____ al McDonald ieri sera", molto probabilmente restituirà "Big Mac".

Applicare il modello di ogni parola per ottenere il suo vettore corrispondente (Figura 1.4)

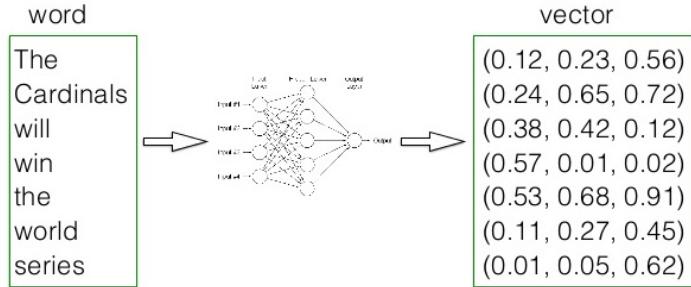


Figura 1.4: Vettori corrispondenti alle parole

Calcolare il vettore delle frasi facendo la media del vettore delle loro parole (Figura 1.5)

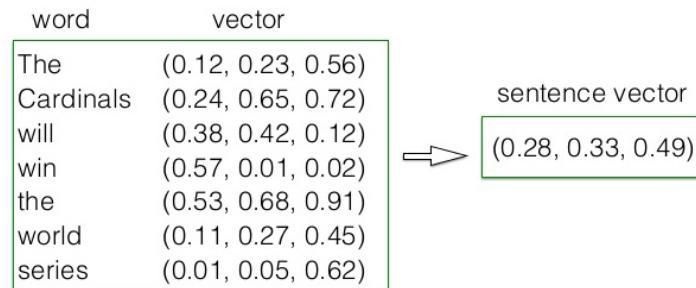
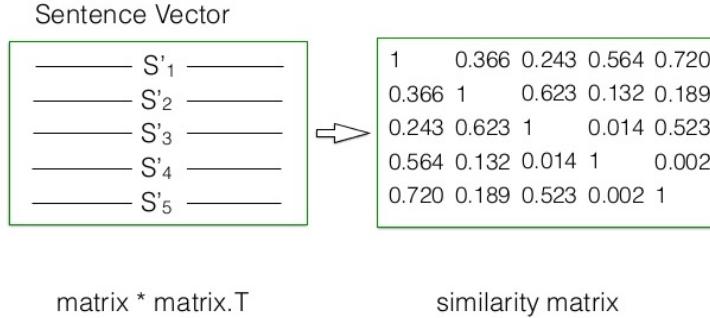


Figura 1.5: Vettori corrispondenti alle frasi

Costruire la matrice di similarità delle frasi (Figura 1.6)

Infine usare PageRank per classificare le frasi nel grafo. (Figura 1.7)

Word2vec è una rete neurale a due layer, sebbene non sia profonda (Deep Neural Network) come spesso definita, trasforma il testo in modo che altre reti neurali possano comprenderlo. Prende in input un corpus di documenti e genera un insieme di vettori: Feature Vectors per ogni parola del corpus. I


Figura 1.6: Matrice di similarità

vettori restituiti sono rappresentazioni numeriche del contesto della singola parola.

Dati abbastanza dati, utilizzo e contesti, Word2vec può apprendere rappresentazioni delle parole altamente accurate, basate sulle apparizioni della parola nei diversi contesti. Queste rappresentazioni possono essere usate per trovare associazioni fra parole o per raggruppare documenti e classificarli per argomento. La similarità fra i vettori può essere misurata attraverso la coseño similarità, dove nessuna similarità è espressa come un angolo di 90 gradi, mentre una similarità totale è data da un angolo di 0 gradi tra i vettori. Ad esempio il vettore relativo a “Sweden” è uguale al vettore “Sweden” mentre il vettore “Norway” ha una distanza di similarità di 0.760124.

Word2vec può apprendere rappresentazioni principalmente in due modi, o usando il contesto per predire la parola data (metodo conosciuto come “continuous bag of word”, o **CBOW**), o usando una parola per predire il contesto (**skip-gram**). Per oggetti simili risulteranno rappresentazioni simili. Questo infatti costituisce uno dei primi passi da compiere per effettuare task di Machine Learning o Data Mining come ad esempio il raggruppamento di vettori simili in cluster attraverso una qualche funzione di similarità. La traduzione in vettori è necessaria per rendere le informazioni facilmente processabili.

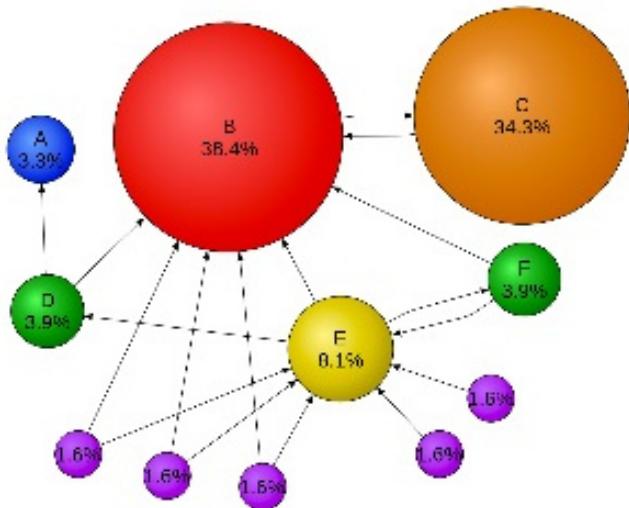


Figura 1.7: Assegna uno score utilizzando PageRank

1.4 Clustering

Più precisamente, il clustering consiste in un insieme di tecniche di analisi multivariata dei dati volte alla selezione e raggruppamento di elementi omogenei in un insieme di dati [28]. Le tecniche di clustering si basano su misure relative alla somiglianza tra gli elementi. In molti approcci questa similarità (o dissimilarità) è concepita in termini di distanza in uno spazio multidimensionale. La bontà delle analisi ottenute dagli algoritmi di clustering dipende molto dalla scelta della metrica, e quindi da come è calcolata la distanza. Gli algoritmi di clustering raggruppano gli elementi sulla base della loro distanza reciproca, e quindi l'appartenenza o meno ad un insieme dipende da quanto l'elemento preso in esame è distante dall'insieme stesso dividendo gli elementi in più cluster(soft/fuzzy clustering) o in un solo cluster(hard cluster). Le tecniche di clustering si possono basare principalmente su due filosofie: partizionale e gerarchico.

Partizionale

Gli algoritmi di clustering di questa famiglia creano una partizione delle osservazioni minimizzando una certa funzione di costo:

$$\sum_{j=1}^k E(C_j) \quad (1.5)$$

dove k è il numero desiderato di cluster, C_j è il j -esimo cluster e $E : C \rightarrow \mathbb{R}^+$ è la funzione di costo associata al singolo cluster. L'algoritmo più famoso appartenente a questa famiglia è il k-means, chiamato così da MacQueen nel 1967 [16].

Gerarchico

Nel clustering gerarchico, invece, è necessario individuare il cluster da suddividere in due sottogruppi. Per questa ragione sono necessarie funzioni che misurino la compattezza del cluster, la densità o la distanza dei punti assegnati ad un cluster. Le funzioni normalmente utilizzate nel caso divisivo sono:

Single-link proximity Calcola la distanza tra i due cluster come la distanza minima tra elementi appartenenti a cluster diversi:

$$D(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y) \quad (1.6)$$

Average-link proximity Questa funzione calcola la distanza tra i due cluster come la media delle distanze elementi:

$$D(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} d(x, y) \quad (1.7)$$

Complete-link proximity Questa funzione valuta la distanza massima tra due punti interni ad un cluster. Tale valore è noto anche come 'diametro del cluster': più tale valore è basso, più il cluster è compatto:

$$D(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y) \quad (1.8)$$

Distanza tra centroidi Questa funzione valuta la distanza massima tra due punti interni ad un cluster. Tale valore è noto anche come 'diametro del cluster': più tale valore è basso, più il cluster è compatto:

$$D(C_i, C_j) = d(\hat{c}_i, \hat{c}_j) \quad (1.9)$$

Nei casi precedenti, $d(x, y)$ indica una qualsiasi funzione distanza su uno spazio metrico.

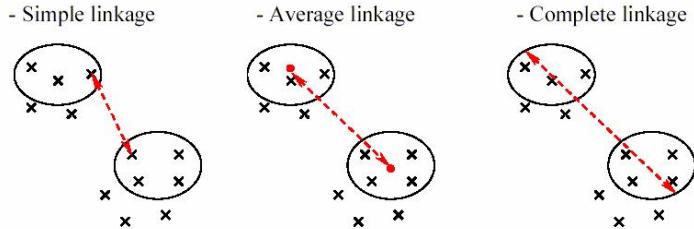


Figura 1.8: Differenze fra i vari tipi di funzioni distanza

1.4.1 Graph Clustering

Un grafo è una coppia ordinata $G = (V, E)$ di insiemi, con V insieme dei nodi ed E insieme degli archi, tali che gli elementi di E siano coppie di elementi da V da $E \subseteq V \times V$ segue in particolare che $|E| \leq |V|^2$.

I grafi sono oggetti discreti che permettono di schematizzare una grande varietà di situazioni e di processi e spesso di consentirne delle analisi in termini quantitativi e algoritmici.

Nello studio di reti complesse, è possibile trovare gruppi di nodi fortemente connessi, che possono essere raggruppati in comunità (potenzialmente sovrapposte). Questa disomogeneità di connessioni suggerisce che esiste una certa divisione naturale all'interno della rete. Nel caso particolare di strutture non sovrapposte, la ricerca di comunità implica la divisione della rete in gruppi di nodi con fortemente connessi internamente e connessioni sparse fra i gruppi. Una definizione più generale è basata sul principio che coppie di nodi sono più probabilmente connessi se fanno parte della stessa comunità, e meno probabilmente connessi se non condividono la stessa comunità.

Le comunità sono molto comuni all'interno delle reti. Le reti sociali includono gruppi di comunità che condividono la posizione, gli interessi, l'occupazione ecc. Essere in grado di individuare queste sotto-strutture all'interno di una rete può fornire indizi su come funziona la rete in considerazione o la topologia che influenza i nodi. Questi indizi possono essere utili per implementare algoritmi sui grafi. Molti metodi di community detection sono stati sviluppati con diversi livelli di successo.

Minimum-cut method

In questo metodo, la rete è divisa in un numero predeterminato di parti, generalmente della stessa grandezza, scelte in modo che il numero degli archi tra i gruppi è minimizzato. Questo metodo funziona bene in molte applicazioni per le quali è stato ideato, ma non è la scelta migliore per scovare comunità in reti generali, dato che troverà comunità indistintamente dal fatto che queste ci siano o meno e troverà solo un numero fissato di comunità.

Hierachical-clustering

Un altro metodo per scovare sotto-strutture concesse nelle reti viene effettuato tramite algoritmi di clustering gerarchico. Con questo approccio si definisce

una misura di similarità fra coppie di nodi. Misure comunemente usate sono la coseno similarità, l'indice di Jaccard e la distanza di Hamming fra le righe della matrice di adiacenza. Poi i gruppi di nodi simili vengono raggruppati in comunità.

Girvan-newman algorithm

Un altro algoritmo molto utilizzato è quello di Girvan–Newman. Questo algoritmo identifica all'interno della rete, gli archi che uniscono community diverse e li rimuove, isolandole. L'identificazione di tali archi è effettuata applicando una misura nota della teoria dei grafi: la **betweenness centrality**. Questa assegna un valore ad ogni arco, che è alto tanto più l'arco è attraversato nel cammino più breve (geodesico) che collega due qualsiasi nodi della rete.

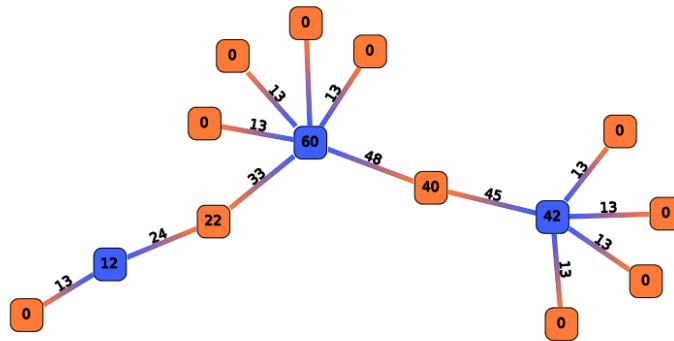


Figura 1.9: Betweenness centrality score

Modularity maximization

La modularità è una misura che viene attribuita al grafo. Questa compara la densità all'interno dei cluster con la densità fra di essi. Indica una certa divisione intrinseca e viene utilizzata per conoscere “quanto” un grafo è separato.

Nonostante i suoi svantaggi uno dei metodi più utilizzati per il community detection è la massimizzazione della modularità. Questo approccio scava strutture connesse tramite la ricerca della miglior divisione di una rete in modo che la modularità risulti massimizzata.

Dato che effettuare un confronto su tutte le possibili combinazioni è solitamente impraticabile, gli algoritmi di questa famiglia si basano su metodi di ottimizzazione approssimati quali algoritmi greedy, cioè che cercano di ottenere una soluzione ottima da un punto di vista globale attraverso la scelta della soluzione considerata migliore ad ogni passo locale. Un famoso approccio di questo tipo è il metodo Louvain, che ottimizza le community locali iterativamente, fin quando la modularità globale non può più essere migliorata.

L'accuratezza di questi algoritmi, comunque, è dibattuta, in quanto è stato dimostrato che molte volte fallisce nell'individuare cluster più piccola di una certa soglia, dipendente dalla grandezza della rete.

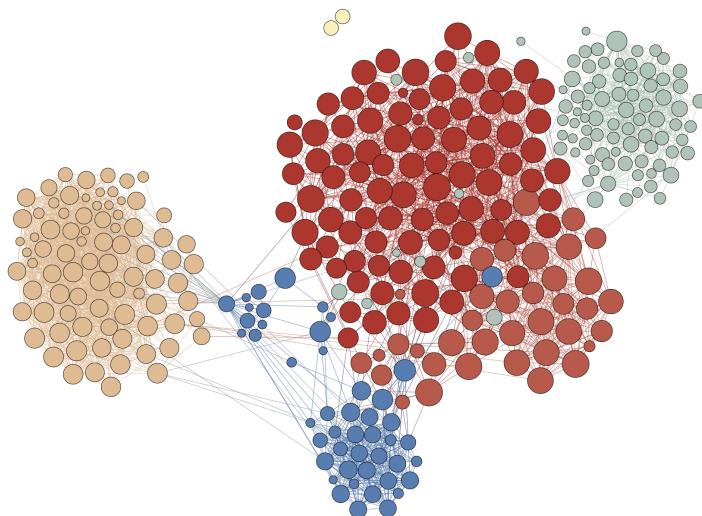


Figura 1.10: Modularity score

Clique-based methods

Le cricche (cliques) sono sottografi dove ogni nodo è collegato con ogni altro nodo nella cricca. Dato che i nodi non possono essere più connessi di così, non è sorprendente che ci siano molti metodi in community detection che si basano su questo approccio. È da notare che dato che un nodo può far parte di più di una cricca, quindi può far parte di più community contemporaneamente, questi metodi restituiscono strutture sovrapposte. Un approccio consiste nel trovare cricche tali che non siano sottografi di altre cricche. Un classico algoritmo per scovare tali strutture è quello di Bron-Kerbosch.

1.4.2 Vector Clustering

Gli algoritmi di clustering in uno spazio vettoriale seguono un altro approccio. Qui viene preso in considerazione la vicinanza (o la distanza) degli elementi rappresentati come punti su un iperpiano. Il feature vector associato può avere grandi dimensioni e può essere ottenuto utilizzando diverse metodologie, dipendentemente dal tipo di dato elaborato.

Hierarchical Clustering

Anche qui il clustering gerarchico è molto utilizzato, seguendo un approccio divisivo (top-down) o agglomerativo (bottom-up), l'idea è quella di unire (o separare) elementi in base alla loro vicinanza, seguendo uno degli approcci già descritti, costruendo così una struttura chiamata dendrogramma che raggruppa gli elementi ad ogni livello. La differenza risiede nel come viene calcolata la distanza.

Centroid-based clustering

Nell'approccio basato sui centroidi, i cluster sono rappresentati da un vettore centrale, che non è necessariamente un membro del dataset. Quando il nu-

mero dei cluster è prefissato ad un numero k , la seguente definizione formale può essere applicata: vengono definiti k centroidi e si prosegue assegnando ogni elemento al centroide più vicino, tale che il quadrato delle distanze dal cluster è minimizzato. Molti algoritmi di questa famiglia richiedono che il parametro k sia stabilito in precedenza, che è il loro più grande svantaggio. Inoltre solitamente vengono trovati cluster di grandezza simile, dato che verrà assegnato un elemento al centroide più vicino. Questi metodi partizionano lo spazio dei dati in una struttura conosciuta come diagramma di Voronoi. Nonostante questo, rimangono tra degli approcci più utilizzati ed efficaci.

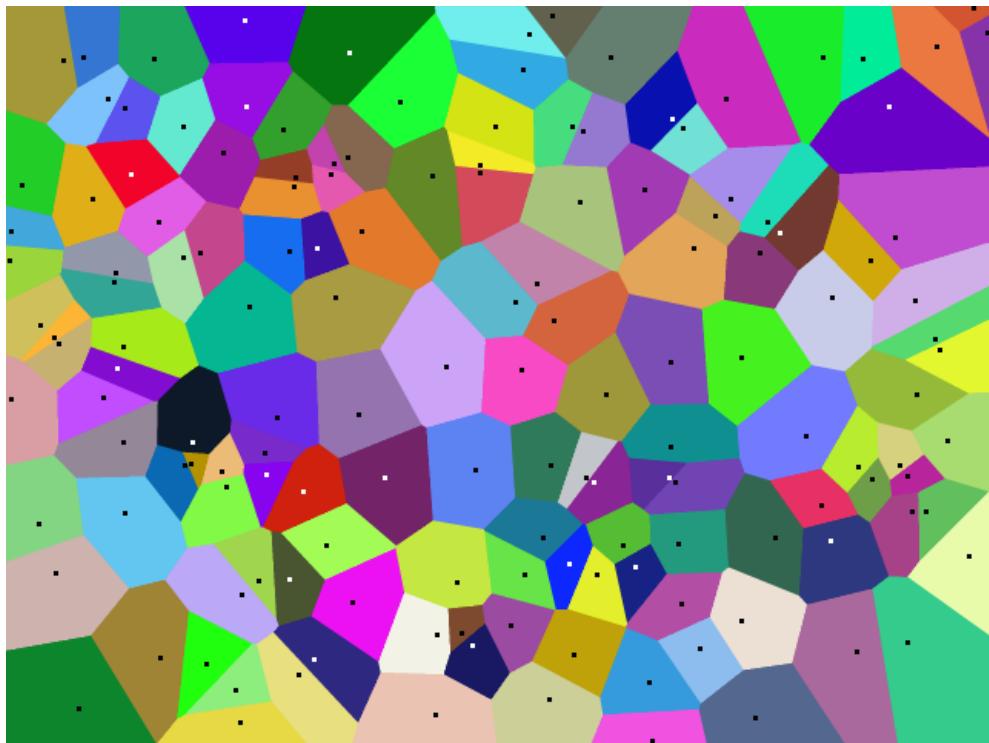


Figura 1.11: Diagramma di Voronoi

Da notare anche la somiglianza concettuale con l'algoritmo di classificazione KNN (k nearest neighbor).

Distribution-based clustering

Il raggruppamento avviene analizzando l'intorno di ogni punto dello spazio. In particolare, viene considerata la densità di punti in un intorno di raggio fissato. Si basano sul considerare collegati due punti che si trovano all'interno di una certa distanza limite. I cluster sono definiti come aree con più alta densità rispetto al resto del dataset. Elementi in un area meno denso sono spesso considerati rumore, quindi come non facenti parte di nessun cluster.

Uno degli svantaggi di questi algoritmi è che si aspettano un certo tipo di densità comune a tutti i cluster. Inoltre non eccellono nel scovare cluster presenti in molti dati del mondo reale.

Document Clustering

Agisce sempre nello spazio vettoriale, si differenzia principalmente nelle operazioni di pre-processing finalizzate ad ottenere un feature vector utilizzabile. Un approccio efficace consiste nel rappresentare i documenti come vettori dove ogni dimensione rappresenta la frequenza di occorrenza di una parola del vocabolario, un insieme di parole precedentemente costruito utilizzando tutte le parole del corpus.

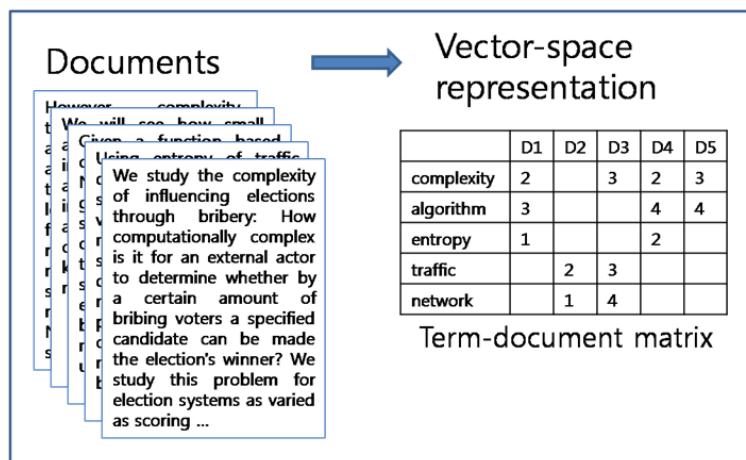


Figura 1.12: Matrice termini-documenti. Ogni riga rappresenta un singolo termine ed ogni colonna rappresenta un singolo documento

Term-Frequency (TF) misura quante volte un termine appare in un documento. Dato che ogni documento ha lunghezza differente, è possibile che un termine possa apparire molte più volte nei documenti più lunghi che in quelli più corti. Quindi può essere necessario dividere la frequenza dei termini per documenti aventi la stessa lunghezza.

Inverse-Document-Frequency (IDF) un altro aspetto da considerare è la frequenza di un termine in un documento relativamente alla sua presenza globale in tutto il corpus. Tenendo in considerazione solo la frequenza di occorrenza tutti i termini sono considerati ugualmente importanti. Termini che appaiono molte volte in un documento ma meno volte in tutto il corpus potrebbero essere molto più significativi per quel specifico documento e portare molta più informazione, quindi tendono ad essere più importanti. Così come i termini che appaiono molte volte in tutti i documenti del corpus sono spesso poco rilevanti e considerati inutili (stopwords).

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|} \quad (1.10)$$

dove $|D|$ è il numero di documenti nella collezione, mentre il denominatore è il numero di documenti che contengono il termine t_i . Tale funzione aumenta proporzionalmente al numero di volte che il termine è contenuto nel documento, ma cresce in maniera inversamente proporzionale con la frequenza del termine nella collezione. L'idea alla base di questo comportamento è di dare più importanza ai termini che compaiono nel documento, ma che in generale sono poco frequenti.

Altre tecniche utilizzate nel clustering sui documenti sono

LSI il **Latent Semantic Indexing** è un metodo di indicizzazione e recupero che usa una tecnica matematica chiamata decomposizione a valori singolari (SVD) per identificare pattern nelle relazioni tra i termini e i concetti contenuti in una collezione non strutturata di testo. La LSI è basa-

ta sul principio che parole che sono usate nello stesso contesto tendono ad avere significato simile. Chiamata così per la sua abilità di correlare semanticamente termini correlati che sono nascosti (latenti) in grandi collezioni testuali. La SVD può venire troncata per task di dimensionality reduction, in modo da diminuire la dimensione del vettore mantenendo comunque il significato.

Coseno similarità una euristica per la misurazione della similitudine tra due vettori effettuata calcolando il coseno tra di loro. Dati due vettori di attributi numerici, A e B , il livello di similarità tra di loro è espresso utilizzando la formula

$$\text{similarity} = \cos \theta = \frac{A \cdot B}{\|A\| \|B\|} \quad (1.11)$$

In base alla definizione del coseno, dati due vettori si otterrà sempre un valore di similitudine compreso tra -1 e $+1$, dove -1 indica una corrispondenza esatta ma opposta (ossia un vettore contiene l'opposto dei valori presenti nell'altro) e $+1$ indica due vettori uguali. Nel caso dell'analisi dei testi, poiché le frequenze dei termini sono sempre valori positivi, si otterranno valori che vanno da 0 a $+1$, dove $+1$ indica che le parole contenute nei due testi sono le stesse (ma non necessariamente nello stesso ordine) e 0 che non c'è nessuna parola che appare in entrambi.

1.4.3 Algoritmi utilizzati

Vengono riportati di seguito gli algoritmi di clustering testati sul dataset generato. L'obiettivo della tesi comunque non è verificare la validità di questi ma verificare se la soluzione proposta rappresenti un miglioramento ed una possibile alternativa alle soluzioni più diffuse e consolidate nell'ambito del clustering di pagine Web.

- **WalkTrap** È un approccio basato su Random Walk. L'idea generale è che se vengono generati dei Random Walk sul grafo, i percorsi rimarranno probabilmente all'interno della stessa comunità perché ci sono meno

archi che congiungono comunità diverse.

L'algoritmo esegue piccoli Random Walk (dipendente da un parametro in input) è usa i risultati per fondere comunità diverse in maniera bottom-up. Tagliando il dendrogramma risultante ad una certa altezza è possibile ricevere il numero di cluster desiderati.

- **Fastgreedy** È un approccio gerarchico bottom-up. Cerca di ottimizzare una funzione di modularità in maniera greedy, euristica attuata effettuando la scelta migliore con le informazioni in possesso ad ogni iterazione. Inizialmente ogni nodo è una comunità separata e ricorsivamente si procede ad unire i nodi in modo che la fusione porti al massimo aumento di modularità rispetto al valore corrente.

L'algoritmo finisce quando non è più possibile aumentare la modularità. È un metodo veloce, solitamente usato come primo approccio perchè non ha parametri da modificare.

- **K-Means** Divide il dataset in un numero prefissato k di cluster. Inizialmente vengono scelti casualmente k punti, non necessariamente facenti parte del dataset, chiamati centroidi. Si procede assegnando ogni data point al centroide più vicino e ricalcolando il centroide sulla media aritmetica dei punti contiene.

Questo processo di assegnazione dei data point e ricalcolo dei centroidi continua fino a quando non avvengono più assegnazioni. I k cluster risultanti saranno quelli restituiti. Anche questo algoritmo rappresenta spesso il punto di partenza nell'analisi di un dataset, in quanto molto spesso porta a buoni risultati ma ha come svantaggio il dover sapere a priori il numero di cluster desiderati.

- **DBSCAN** Deriva da *Density-Based Spatial Clustering of Applications with Noise* è un algortimo basato sulla densità, connettendo regioni di punti con densità sufficientemente alta. Fondamentalmente, un punto q è direttamente raggiungibile da un punto p se non viene superata una data distanza ϵ e se p è circondato da un numero sufficiente di punti, allora p e q possono essere considerati parti di un cluster.

Si può affermare che q è density-reachable da p se c'è una sequenza

p_1, p_2, \dots, p_n di punti con $p_1 = p$ e $p_n = q$ dove ogni p_{i+1} è density-reachable direttamente da p_i . Da notare che la relazione density-reachable non è simmetrica (dato che q potrebbe situarsi su una periferia del cluster, avendo un numero insufficiente di vicini per considerarlo un elemento genuino del cluster). Di conseguenza la nozione density-connected diventa: due punti p e q sono density-connected se c'è un punto o tale che sia o e p che o e q sono density-reachable.

Un cluster, che è un sotto-insieme dei punti del database, soddisfa due proprietà:

- i) Tutti i punti all'interno del cluster sono mutualmente density-connected.
- ii) Se un punto è density-connected a un altro punto del cluster, anch'esso è parte del cluster.

- **HDBSCAN** Deriva da *Hierarchical Density-Based Spatial Clustering of Applications with Noise* [5]. Applica DBSCAN variando il valore dell' ϵ ed integra i risultati restituendo cluster che stabilizzano meglio tale valore.

Questo permette ad HDBSCAN di trovare cluster con densità diversa, principale svantaggio di DBSCAN.

1.5 Data Visualization

Una nota sulla visualizzazione dei dati, campo in crescita data la corrispondente crescita su economie basate sull'informazione e sulla crescita dei dati generati (big data) portata avanti anche da campi relativamente nuovi nel campo dell'analisi dei dati, come Business Analytics, Business Intelligence, Data Science etc.

Tale disciplina è indirizzata a comunicare informazioni in modo chiaro e comprensibile, attraverso grafici, tavole, diagrammi ecc. La visualizzazione può spesso aiutare ad analizzare e ragionare sui dati, rendendo dati complessi molto più accessibili ed usabili.

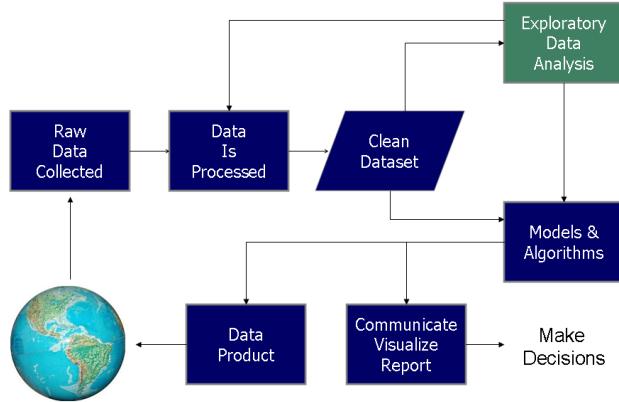


Figura 1.13: La visualizzazione dei dati è un passo fondamentale nell’analisi dei dati.

Immaginare rappresentazioni di dati in spazi multidimensionali non è intuitivo. Un modo efficace per raggiungere tale scopo può essere effettuato tramite *dimensionality reduction*. Queste consistono nel ridurre la dimensione dei vettori un uno spazio a due o tre dimensioni per poterle rappresentare in modo comprensibile all’occhio umano, che possono avvenire attraverso trasformazioni lineari [20] o non lineari [31].

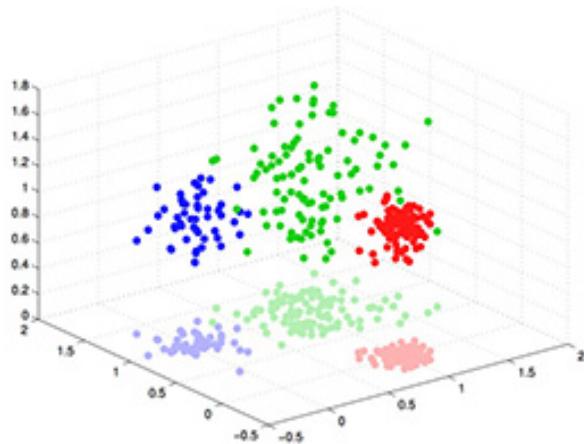


Figura 1.14: Riduzione da tre a due dimensioni.

Capitolo 2

Il Sistema Url2vec

I grafi possono essere utilizzati per modellare innumerevoli fenomeni reali, da quelli scientifici a quelli sociali. Questi rappresentano le entità non come singole unità descritte appieno dalle loro caratteristiche, ma piuttosto dai collegamenti che intercorrono tra di esse. Queste "relazioni" codificano la maggior parte delle informazioni latenti in un grafo, caratterizzandolo in maniera univoca.

In questa tesi si approfondiscono tecniche e metodologie per le estrazione di queste informazioni dal grafo del Web, più precisamente di un sito Web, indirizzate alla divisione delle pagine in cluster omogenei. Le pagine Web tuttavia non sono divisibili in classi ben definite, non ci sono informazioni esplicite che rappresentano univocamente i professori, gli studenti o i corsi. Data questa assenza, non è possibile generare un modello accurato di classificazione. Si è rivelato necessario risolvere il problema con un altro approccio.

A dispetto della diversità delle pagine Web nella rete, quelle che risiedono all'interno di una particolare organizzazione, spesso, condividono una certa struttura. Questa può essere appresa attraverso l'analisi del peso, della densità o della direzione degli hyperlink presenti nelle pagine Web, ovvero gli archi del grafo costruito su di esse. Ad esempio, il sito Web di un dipartimen-

to di informatica conterrà pagine riguardanti i docenti, gli studenti, i corsi, la ricerca che saranno accessibili attraverso determinati *percorsi* del grafo. L’analisi dei grafi comporta anche degli svantaggi, gli algoritmi della teoria dei grafi sono NP-completi [11], e questo non può essere trascurato considerando le dimensioni del Web. Nella metodologia presentata viene proposta l’alternativa di apprendere rappresentazioni vettoriali dei vertici all’interno del grafo, utilizzandole insieme alle informazioni estratte dal contenuto di una pagina Web, così da processare vettori linearmente indipendenti. Queste rappresentazioni vettoriali sono apprese attraverso la generazione di percorsi attraverso gli archi del grafo, creando delle sequenze di vertici che saranno trattate come frasi, dove le parole sono pagine Web.

2.0.1 NLP nel Web: URL Embedding

In questo modo è possibile sfruttare i recenti sviluppi nel campo del Word Embedding. Questi algoritmi riescono a considerare sempre più fattori e restituire vettori sempre più accurati. Vengono incluse informazioni riguardanti il contesto di una parola, ovvero gli altri termini contenuti nella frase in cui compare la parola analizzata. Nel Web questo implica considerare le pagine ”più vicine”, ovvero le pagine che puntano o sono puntate direttamente dalla pagina in esame, privilegiando quelle che hanno più collegamenti con questa. Questa informazione riesce ad estrarre le informazioni latenti nella struttura del sito, ed a raggruppare le pagine similmente ad un algoritmo di partizionamento di un grafo. Queste informazioni hanno rivelato proprietà peculiari quando applicate su grandi testi. Il caso più noto e controverso è quello delle analogie.

È stato osservato che correlazioni nascoste possono trovarsi nella differenza tra coppie di vettori [18], come nel caso di parole simili ma con leggere differenze come il genere o il numero, infatti esse appaiono in frasi tendenzialmente simili, ma con delle piccole differenze. In questo caso l’informazione può essere interpretata come il genere o il numero.

Informazioni simili possono essere trovate anche nel campo del Web. È ormai consolidata la questione dell'autorità di alcune pagine [13] e di come queste abbiano un maggior numero di link in entrata. Alcune pagine tuttavia saranno accessibili prevalentemente attraverso alcuni percorsi e si troveranno quindi in un contesto con elementi simili. Il problema resta dare un senso a queste correlazioni ed un significato utilizzabile. Nello sviluppo di Url2vec, tuttavia, sono emerse corrispondenze abbastanza naturali, come le pagine dei professori con i relativi corsi insegnati o i laboratori di afferenza.

Queste relazioni sono dovute al fatto che il grafo di un sito Web è fortemente connesso, ma la maggior parte delle volte in cui appare un determinato corso di studio appare anche il relativo docente e viceversa.

Differenza tra parole ed URL

La sola analisi delle sequenze comunque può risultare limitata. Le pagine Web non sono parole, ad esse è associata un ulteriore informazione, ovvero il contenuto testuale. Questa proprietà si è rivelata molto utile, infatti combinando le informazioni ricavate dall'esame di tutti e due gli aspetti, il grado di accuratezza del processo di clustering è salito in modo significativo. Il contenuto di una pagina infatti fornisce informazioni fondamentali su di essa, ma queste possono variare significativamente anche tra pagine considerate dello stesso tipo, presentando ad esempio una diversa distribuzione dei termini. Analizzando la struttura connessa delle pagine Web, si è cercato costruire delle rappresentazioni più significative. Saper sfruttare tale struttura può agevolare notevolmente i task di Web Mining. Esso infatti estrae conoscenza strutturata e facilmente processabile.

Nelle sottosezioni che seguono si descrive il sistema realizzato, oltre che le tecniche e gli algoritmi utilizzati per la realizzazione degli obiettivi preposti. Sono state realizzate tre macro componenti.

- *Crawling delle pagine Web.* Questo è regolato da alcuni parametri che rendono il processo flessibile e altamente modificabile.

- *Costruzione del Dataset*, ovvero la strutturazione delle informazioni ottenute nel processo precedente, secondo alcuni criteri necessari per le successive elaborazioni.
- *Clustering delle pagine Web* attraverso la combinazione dell’analisi del contenuto testuale e delle relazioni tra le pagine.

2.1 Web Crawling per l'estrazione dei dati

Le proprietà che caratterizzano le pagine Web rendono complicato il processo di estrazione di informazioni, soprattutto nel caso in cui i contenuti vengono generati dinamicamente.

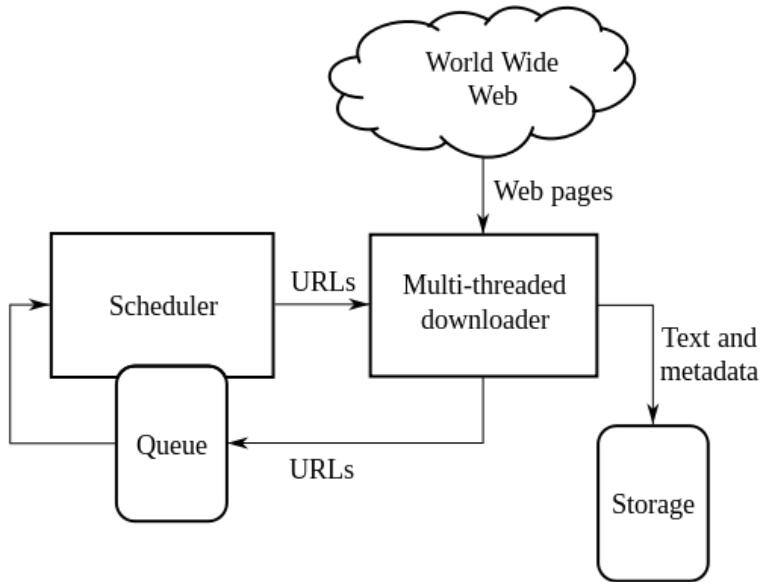


Figura 2.1: Architettura di un web crawler

Un Web crawler, chiamato anche web spider o web robot, è un componente software. I suoi obiettivi principali sono:

- raccogliere il più velocemente ed efficientemente possibile pagine utili, insieme alla struttura ad hyperlink che le collega;

- copiare tutte le pagine che visita per elaborazioni future, per poi indicizzarle così che gli utenti possano trovarle più velocemente;
- validare gli hyperlink e il codice HTML;
- eseguire il Web Scraping delle pagine Web.

Esso esplora una pagina alla volta, analizzandone la struttura e gli hyperlink contenuti. Questi sono immagazzinati nella "frontiera" che inizialmente vuota, conserva tutte le pagine ancora da esplorare. L'ordine di esplorazione e le politiche di filtraggio degli hyperlink possono variare in base al risultato desiderato.

Algorithm 1 Crawling BFS

```

url ← homepage; //Pagina da cui iniziare
queue ← empty //Coda per la BFS
analyzedVertex.add(url); //Insieme di url già visitati
maxDepthurl; //Massima profondità di esplorazione

Begin
while queue ≠ empty do
    urlToAnalyze ← queue.dequeue()
    if urlToAnalyze.depth ≤ maxDepth then
        outlinks ← urlToAnalyze.getOutlinks()
    else
        urlToAnalyze
    end if
    if outlinks ≠ null then
        outlinks ← urlToAnalyze.getOutlinks()
        analyzedVertex.add(outlinks)
        serialize(urlToAnalyze)
        for each link in outlinks do
            serialize(link)
            queue.enqueue(link, urlToAnalyze + 1)
        end for
    end if
end while
End

```

2.1.1 Proprietà del web crawler

Nel processo di Crawling, data la natura non strutturata del Web, è necessaria l'applicazione di numerose tecniche e metodologie per un corretto funzionamento.

Normalizzazione degli URL

Il termine di normalizzazione, chiamato anche canonicalizzazione di URL, si riferisce al processo di modifica e standardizzazione di un URL in una maniera consistente, ad esempio alcuni siti Web mettono a disposizione gli stessi file o i medesimi contenuti attraverso URL differenti.

```
http://domain.com/products/page.php?product=smartphone  
http://domain.com/products/smartphone.php
```

```
http://www.domain.edu/courses  
http://www.domain.edu/courses/index.html
```

Le due coppie di URL nell'esempio puntano agli stessi contenuti. Altri casi sono URL che differiscono solo per il protocollo ("http://" o "https://") o l'omissione della stringa "www". Effettuando la normalizzazione, si sceglie un URL come formato di riferimento per accedere ad un determinato contenuto. Ci sono diversi tipi di normalizzazione che possono essere usati, come la conversione in minuscolo, rimuovere i ".." e "." portando gli URL relativi ad URL assoluti, aggiungere slash finali al componente di percorso non vuoto.

Una soluzione consiste nella creazione di un dizionario che ha come chiavi gli hashcode del contenuto testuale delle pagine e come valori una lista di tutti i diversi URL che hanno quel contenuto. In questo modo si riduce il problema ad una sola operazione così da annullare le relazioni e i vari di pattern da scovare nell'analisi degli URL per capire se sono la stessa pagina o meno.

Ricerca all'interno dello stesso dominio

Per estrarre informazione e per il successivo processo di clustering delle pagine web, è necessario che le pagine estratte si riferiscano allo stesso dominio, o in altri termini, che appartengano allo stesso sito web.

Questo viene effettuato per sfruttare la struttura gerarchica del dito web rivolto principalmente riuscire a raccogliere le informazioni nascoste nel grafo e soprattutto negli hyperlink.

Dimensione della ricerca

Anche limitando la ricerca ad un solo dominio, la dimensione delle pagine da esplorare, e quindi la grandezza della frontiera, può aumentare considerevolmente o addirittura portare il processo di crawling a divergere.

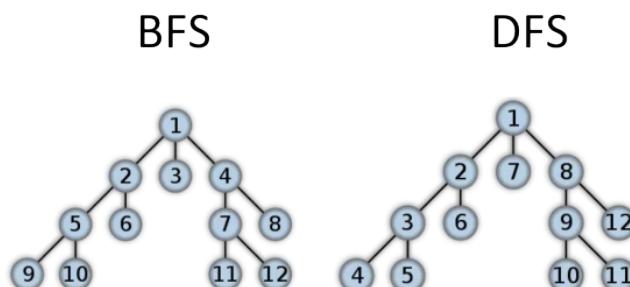


Figura 2.2: Differenze tra ricerca in ampiezza e ricerca in profondità

È stata utilizzata una ricerca in ampiezza con un limite variabile di profondità. Ponendo un limite all'esplorazione del grafo delle pagine si garantisce la terminazione del processo di crawling e utilizzando una ricerca in ampiezza si dà priorità alle pagine vicine al nodo radice (comunemente l'homepage). Questa scelta è stata dettata anche dal cercare di evitare le cosiddette "spider traps". Queste sono dei meccanismi utilizzati, intenzionalmente o involontariamente, dai server oggetto di crawling, che possono portare ad una generazione dinamica e potenzialmente infinita in URL univoci, e quindi considerati come pagine diverse. Questo può essere evitato non aggiungendo

alla frontiera URL che contengono il carattere "?", ma questo non è sempre efficace.

Inoltre è stato osservato che nei siti web entity-oriented, le informazioni ricercate si trovano quasi sempre nei primi livelli di gerarchia [12].

Restrizione dell'esplorazione

La restrizione può essere effettuata sulle pagine da esplorare o dalla frequenza di richieste che è possibile effettuare.

Il robots exclusion protocol è uno standard che consiste in un file (robots.txt), posto alla radice della gerarchia di un sito Web. In pratica, il file indica le regole utilizzate dai crawler per applicare restrizioni di analisi sulle pagine di un sito web.

Molte volte i crawler non sono ben accetti, in quanto possono rallentare pesantemente la navigazione. Se server effettua dei controlli sulle richieste ricevute e queste hanno una frequenza troppo elevata o seguono uno schema riconducibile ad una macchina queste possono essere ignorate. Può capitare che richieste continue e non curanti delle restrizioni imposte portino a bandire l'indirizzo IP del servizio trasgressore.

Per evitare tali conseguenze è opportuno seguire una certa etica nell'operazione di crawling.

2.1.2 Estrazione delle liste

Riuscire a strutturare dati non strutturati può rivelarsi ostico. Molti tentativi, più o meno efficaci, sono stati effettuati a tale proposito.

In questa tesi uno degli obiettivi è stato quello di prendere in considerazione i collegamenti all'interno delle pagine web, seguendo i percorsi generati dalla concatenazione di più hyperlink. Questa scelta è stata effettuata sulla base

dei recenti progressi nel campo del natural language processing (NLP) [29]. Esplorando la struttura del grafo del web, data la forte connessione che esiste fra i suoi nodi, estrarre informazione può rivelarsi un'operazione tutt'altro che banale.

È stato introdotto il concetto di **lista**. Per permettere una migliore visualizzazione dell'informazione descritta, quasi tutte le pagine web vengono formattate utilizzando regole CSS. Di conseguenza, per poter effettuare correttamente questo task, bisogna prima elaborare la pagina Web con tutte le informazioni grafiche sui nodi HTML e solo successivamente si può procedere alla loro estrazione. Grazie alle informazioni ricavate dall'HTML della pagina Web e dalla posizione e dimensione dei singoli nodi, è possibile stabilire una struttura gerarchica ad albero dei nodi che la compongono. Questa struttura gerarchica ad albero permette di scoprire i record, ovvero dati strutturati (ad esempio provenienti da database) che sono allineati orizzontalmente, verticalmente e anche strutturalmente (elementi HTML ul, li, . . .); permette quindi di individuare dei gruppi di record, ovvero delle liste di record.

L'individuazione delle liste di raggiungere pagine che contengono elementi simili a quelli contenuti nella pagina che si sta visualizzando. In questo modo le pagine dovrebbero essere accessibili solo attraverso percorsi predeterminati e non in maniera fortemente connessa. I nodi e gli archi risultanti risultano così un sottoinsieme di quelli originali.

2.2 Costruzione del Dataset

I dati estratti nel processo vanno organizzati e ampliati in modo da garantire l'accesso e l'elaborazione in maniera agevole.

Il processo di crawling restituisce il grafo delle pagine web e il contenuto testuale di ogni pagina esplorata, a queste va aggiunta la generazione delle sequenze attraverso la teoria dei *Random Walk*, come spiegato in 2.2.1. Inoltre per un minor spreco di risorse si è optato per la conversione degli URL in codici, ovvero una associazione univoca fra un URL ed un codice (e.g. un

numero) molto più corto, così da risparmiare tempo di elaborazione e spazio di archiviazione.

2.2.1 Random Walk

Per la generazione delle sequenze è stato utilizzata la teoria dei *Random Walk*. In matematica, un Random Walk è la formalizzazione dell'idea di prendere passi successivi in direzioni casuali. Matematicamente parlando, è il processo stocastico più semplice, il processo markoviano. Qui sono utilizzati per ricavare percorsi pseudo casuali da attraversamento del grafo del web per ottenere sequenze di URL collegati semanticamente.

In una passeggiata aleatoria monodimensionale si studia il moto di una particella puntiforme vincolata a muoversi lungo una retta nelle due direzioni consentite. Ad ogni movimento essa si sposta (a caso) di un passo a destra (con una probabilità fissata p) o a sinistra con una probabilità $1 - p$, ed ogni passo è di lunghezza uguale e indipendente dagli altri.

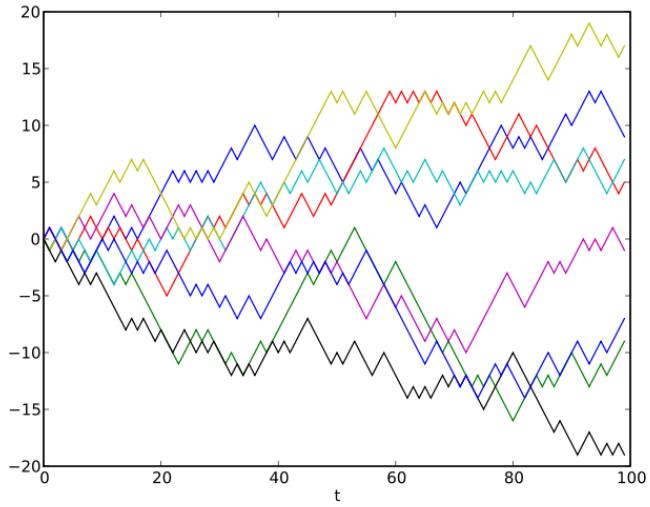


Figura 2.3: Rappresentazione visuale di 8 random walk monodimensionali.

In una passeggiata aleatoria bidimensionale si studia il moto di una particella vincolata a muoversi sul piano spostandosi casualmente ad ogni passo a destra o a sinistra con probabilità $\frac{1}{2}$, verso l'alto o verso il basso con probabilità $p = \frac{1}{2}$. In pratica ad ogni passo può compiere un movimento lungo una delle quattro diagonali con probabilità $\frac{1}{4}$. Ci si chiede con che probabilità la particella tornerà al punto di partenza.

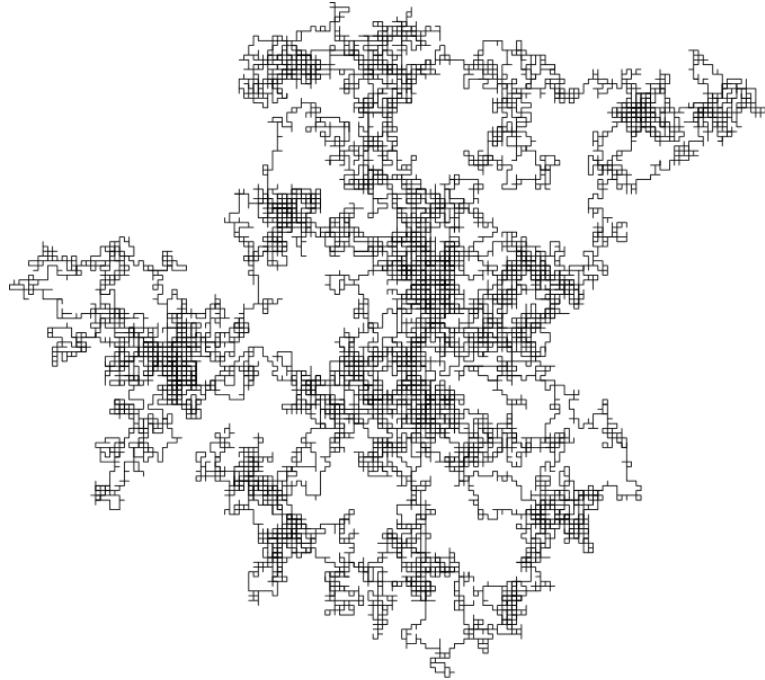


Figura 2.4: Rappresentazione visuale di un random walk di 25.000 passi su due dimensioni.

Queste passeggiate aleatorie possono trovare effettivi riscontri in natura come il traiettoria percorso da una particella in un liquido o in un gas, il tragitto di un animale affamato o anche il prezzo di un titolo fluttuante o la situazione finanziaria di un giocatore d'azzardo. Tutti questi esempi possono essere espressi come random walk, anche se in natura potrebbero non essere veramente casuali.

Un popolare modello di random walk è quello su un reticolo regolare, dove ad ogni passo si segue un determinato percorso basandosi su una qualche

distribuzione di probabilità. Nel caso più semplice si può solo “saltare” sul sito vicino. In un semplice random walk simmetrico in un reticolo localmente finito, le probabilità di saltare su ognuno dei siti vicini è la stessa.

Per definire un random walk formalmente, prese indipendenti variabili casuali Z_1, Z_2, Z_3, \dots dove ogni variabile è o 1 o -1, con una probabilità del 50% per ognuno dei due casi, e dato

$$S_0 = 0 \quad (2.1)$$

$$S_n = \sum_{j=1}^n Z_j \quad (2.2)$$

la serie $\{S_n\}$ è chiamata random walk semplice in \mathbb{Z} .

Si consideri un attraversatore casuale della rete (random surfer) che, partendo da un pagina web, esegue un passo alla volta in questo modo: ad ogni iterazione, dalla pagina corrente, procede verso una pagina a caso tale che esiste un link che dalla pagina corrente punta a questa.

Procedendo da pagina a pagina, visiterò alcuni nodi, più spesso di altri; intuitivamente, saranno nodi con molti link entranti da altri nodi frequentemente visitati. Ci sono alcuni problemi. Che succede se si arriva ad una pagina che non ha link in uscita? In questo caso è necessario introdurre un'altra operazione: il teletrasporto. In questa operazione l'attraversatore casuale salta dal nodo corrente ad un qualsiasi altro nodo sulla rete. Se N è il numero totale dei nodi nel grafo, l'operazione di teletrasporto porta l'attraversatore verso ogni nodo con probabilità $\frac{1}{N}$. Potrebbe anche trasportarsi sulla posizione corrente con probabilità di $\frac{1}{N}$. Questa operazione è chiamata quando si arriva ad un non senza nodi in uscita o con una probabilità d data, con $0 < d < 1$.

2.2.2 Generazione delle sequenze

Le sequenze rappresentano un percorso che un attraversatore casuale della rete seguirebbe cliccando su un hyperlink a caso fra tutti quelli disponibili nella pagina corrente (o eventualmente nelle liste). Questi percorsi, chiamati **Random Walk** (o passeggiate aleatorie), sono stati largamente utilizzati in molti algoritmi sui grafi e sul Web [1] in quanto buone approssimazioni di comportamenti casuali. Il problema di questa tecnica applicata al Web si presenta quando l'attraversatore casuale arriva ad una pagina priva di hyperlink. La soluzione più diffusa consiste nell'effettuare un "salto" verso una qualsiasi altra pagina quando non ci sono outlink da seguire.

Qui il problema non si pone, in quanto le sequenze generate hanno una lunghezza fissata prima dell'esecuzione e se la generazione dovesse bloccarsi, la sequenza risultante sarà solo più piccola. Questa scelta è dovuta dal fatto che l'informazione cercata scaturisce da percorsi reali di navigazione e non necessita una lunghezza obbligatoria da rispettare, in quanto le sequenze possono essere viste come frasi di un testo, dove le parole sono gli URL.

Per motivi di sperimentazione sono stati implementati tre tipi diversi di Random Walk, utilizzabili modificando i parametri di esecuzione dell'algoritmo.

Algorithm 2 Generazione delle sequenze

```
numRandomWalks; //Numero di Random Walk da generare  
lengthRandomWalks //Lunghezza dei Random Walk  
  
Begin  
node ← RANDOMNODE  
for  $i \leftarrow 0 \rightarrow numRandomWalks$  do  
    sequence.add(node)  
    while sequence.length ≤ lengthRandomWalks do  
        if node.hasOutlinks() then  
            node ← node.getOutlinks(RandomIndex)  
            sequence.add(node)  
        else  
            break  
        end if  
    end while  
    serialize(sequence)  
end for  
End
```

Random Walk standard

Questo è il caso standard, ovvero si parte da un nodo casuale del grafo e si segue ogni volta un arco a caso fra quelli disponibili, fino al raggiungimento della lunghezza prefissata o all'impossibilità di proseguire.

Da notare che questo processo e il precedente non sono completamente separati, in quanto la scelta di un nodo casuale e la consecutiva traiettoria, possono portare all'esplorazione di pagine non precedentemente visitate. È quindi necessario mantenere aggiornato il grafo immagazzinato.

Random Walk con partenza fissa

Qui l'unica differenza consiste nel punto di partenza del cammino. Infatti si può partire in un nodo prefissato del grafo (generalmente l'homepage di un sito web), in modo da esplorare più percorsi possibili avente quel nodo come origine.

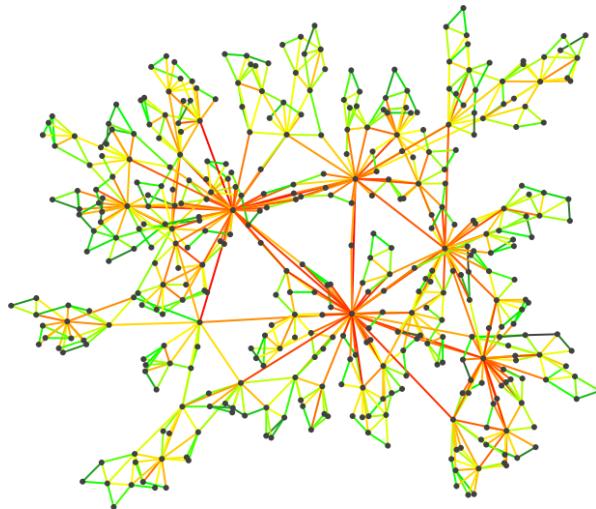


Figura 2.5: Random walk sul grafo.

Random Walk attraverso le Liste

Qui invece si può seguire uno dei due approcci precedenti, ma con il vincolo delle liste, quindi limitando la camminata ad un sottoinsieme di quella precedente.

2.2.3 Esempio di Dataset

Di seguito sono elencati file generati nel processo di crawling e di generazione delle sequenze. Gli esempi riportati di seguito sono stati ottenuti analizzando il sito del dipartimento di informatica di Urbana, IL: www.cs.illinois.edu

urlsMap.txt

Contiene la associazioni fra gli URL e il relativo codice identificativo. Questo è dovuto dalle ragioni spiegate in precedenza, ovvero ridurre i tempi di elaborazione e spazio di archiviazione.

<http://cs.illinois.edu>,3

```
http://cs.illinois.edu/prospective-students,4  
http://cs.illinois.edu/current-students,5  
http://cs.illinois.edu/courses,6  
http://cs.illinois.edu/alumni,7  
http://cs.illinois.edu/research,8  
http://cs.illinois.edu/news,9  
http://cs.illinois.edu/partners,10  
http://cs.illinois.edu/about-us,11  
. . .
```

vertex.txt

Contiene il contenuto testuale di ogni pagina esplorata. Ogni riga è quindi formata da il codice identificativo di un URL e il relativo contenuto.

```
1 department of computer science at illinois engineering at ...  
2 prospective students department of computer science at ...  
3 current students department of computer science at ...  
4 courses department of computer science at illinois ...  
5 alumni department of computer science at illinois ...  
6 research department of computer science at illinois ...  
7 news department of computer science illinois engineering ...  
8 partners department of computer science at illinois ...  
9 about us department of computer science at illinois ...  
. . .
```

edges.txt

Questo è il file principale per la generazione delle sequenze, qui sono immagazzinate tutte le relazioni fra le pagine, ovvero gli archi che le collegano.

```
1 1  
1 2  
1 3  
1 4  
1 5  
1 6  
1 7  
1 8  
1 9  
. . .
```

sequencesIDs.txt

Contiene le sequenze generate. I codici relativi alle pagine web sono separati da ” -1 ” e la linea finisce con un ” -2 ”. Da notare che sono riportate le sequenze che partono da un nodo casuale del grafo.

```
137 -1 2 -1 27 -1 8 -1 52 -1 53 -1 8 -1 8 -1 10 -1 13 -1 -2  
506 -1 5 -1 14 -1 11 -1 6 -1 2 -1 27 -1 114 -1 111 -1 11 -1 -2  
424 -1 4 -1 12 -1 6 -1 8 -1 53 -1 4 -1 7 -1 12 -1 8 -1 -2  
616 -1 5 -1 6 -1 8 -1 8 -1 9 -1 1 -1 21 -1 6 -1 3 -1 -2  
51 -1 7 -1 7 -1 38 -1 38 -1 25 -1 103 -1 27 -1 113 -1 12 -1 -2  
429 -1 10 -1 3 -1 6 -1 4 -1 11 -1 8 -1 9 -1 9 -1 3 -1 -2  
783 -1 421 -1 5 -1 9 -1 7 -1 5 -1 2 -1 8 -1 2 -1 24 -1 -2  
506 -1 5 -1 8 -1 52 -1 53 -1 25 -1 40 -1 13 -1 11 -1 13 -1 -2  
638 -1 63 -1 153 -1 62 -1 63 -1 152 -1 63 -1 155 -1 13 -1 7 -1 -2  
. . .
```

sequencesIDsFromHomepage.txt

Contiene le sequenze generate che partono da uno stesso nodo. La generazione di questo file avviene esplicitando il nodo di origine di ogni sequenza nella fase di generazione.

```
1 -1 8 -1 2 -1 14 -1 2 -1 2 -1 10 -1 14 -1 66 -1 3 -1 -2
1 -1 18 -1 39 -1 8 -1 8 -1 24 -1 1 -1 4 -1 7 -1 25 -1 -2
1 -1 23 -1 -2
1 -1 16 -1 10 -1 3 -1 29 -1 29 -1 4 -1 25 -1 97 -1 108 -1 -2
1 -1 20 -1 20 -1 1 -1 11 -1 3 -1 2 -1 9 -1 10 -1 13 -1 -2
1 -1 25 -1 48 -1 48 -1 44 -1 42 -1 4 -1 7 -1 38 -1 38 -1 -2
1 -1 25 -1 115 -1 4 -1 11 -1 11 -1 1 -1 2 -1 26 -1 13 -1 -2
1 -1 24 -1 4 -1 9 -1 60 -1 56 -1 6 -1 25 -1 113 -1 116 -1 -2
1 -1 21 -1 25 -1 135 -1 32 -1 13 -1 4 -1 25 -1 149 -1 59 -1 -2
. . .
```

2.3 Web page Clustering

Il clustering delle pagine Web all'interno di uno stesso dominio può venire effettuato sfruttando la struttura connessa del sito o trattando le singole pagine Web come documenti. Nel primo caso si tratta principalmente di utilizzare algoritmi e tecniche derivanti dalla teoria dei grafi per partizionare il grafo, in modo da isolare i cluster di pagine "simili". Nel secondo caso invece viene analizzato unicamente il contenuto testuale visivo della pagina, si costruisce una collezione di documenti, un vocabolario dei termini e si considera ogni pagina indipendentemente dalle altre.

Gli hyperlink tra le pagine di uno stesso sito sono tipicamente utilizzati per organizzare i contenuti e puntano a contenuti inerenti ma non per forza simili. Mentre il contenuto testuale, data l'ambiguità del linguaggio naturale, può

fornire indizi sbagliati e considerare diverse pagine correlate solo per una differente distribuzione dei termini.

In questa tesi si è scelto un approccio diverso, la rappresentazione di ogni pagina infatti viene creata unendo le informazioni sulla struttura della pagina all'interno del grafo con le informazioni sul contenuto testuale. È importante notare che non si applicano tecniche di partizionamento del grafo, le relazioni tra le pagine vengono apprese mediante i Random Walk generati nella fase precedente e codificate in uno spazio vettoriale. Questo comporta numerosi vantaggi computazionali. Inoltre in questo modo è possibile unire queste rappresentazioni con quelle basate sulla frequenza dei termini. In particolare le sequenze generate attraverso la teoria dei Random Walk, vengono trattate come frasi ed analizzate con tecniche di Word Embedding derivanti dal campo del Natural Language Processing. La rappresentazione vettoriale di ogni pagina non è umanamente comprensibile come la rappresentazione termini-documenti, ma è conveniente da elaborare. Queste sono state le premesse alla base del lavoro svolto, ovvero unire tecniche consolidate in aree specifiche e trasferire la conoscenza in altri domini applicativi.

L'algoritmo implementato prende in input l'insieme delle pagine, con il relativo contenuto, e le sequenze di Random Walk generate, restituendo rappresentazioni vettoriali per ogni pagina. La fase di apprendimento può comunque essere opportunamente personalizzata considerando maggiormente uno dei due aspetti o gli algoritmi utilizzati per ognuno. Questo è dovuto all'eterogeneità del Web. Infatti è stato notato che non tutti i siti tendono a formare cluster sferici, o ancora che l'organizzazione interna di un sito non sia sufficientemente informativa.

Capitolo 3

Stato dell'Arte

Il clustering di pagine Web non è un nuovo ambito di ricerca. Nei diversi anni si sono susseguite in letteratura una serie di tecniche e metodologie per il raggruppamento ed il reperimento di pagine Web [2, 8, 4, 6, 10].

Tuttavia gli sforzi si sono concentrati prevalentemente su pagine provenienti da diversi siti Web. Relativamente poco è stato il lavoro svolto sul clustering di un specifico sito di una determinata organizzazione. L'analisi degli hyperlink, infatti, ha significati diversi in relazione al dominio di destinazione. Se la pagina puntata si trova nello stesso dominio, il collegamento avrà funzione di organizzazione dei contenuti, mentre se il collegamento punta ad una pagina esterna, questo sarà indirizzato alla conferma del contenuto mostrato e presenterà molto probabilmente argomenti simili [].

Gli algoritmi di clustering di pagine web possono essere classificati in quattro grandi categorie in base alle informazioni che questi utilizzano per raggruppare le pagine web:

- **Algoritmi di clustering basati sul contenuto testuale.** Ricadono in questa categoria gli algoritmi che analizzano la distribuzione dei termini delle singole pagine web e raggruppano le stesse in funzione dei topic descritti. I cluster così ottenuti sono composti da pagine non necessariamente collegate tra loro, ma trattanti argomenti simili. In [6]

gli autori trattano le pagine come documenti e propongono un metodo per rilevare la pertinenza di una pagina ad un topic e raggruppare le pagine in funzione dei topic estratti. In questo modo, tuttavia, si assume l'indipendenza tra le pagine web analizzate non considerando le relazioni che intercorrono.

Come proposto in [8], questo approccio suggerisce l'utilizzo di informazioni semantiche per migliorare il processo di estrazione, come ad esempio meta-informationi sulla struttura e/o sulla gerarchia delle pagine web o euristiche quali la conoscenza del tool utilizzato per la creazione delle pagine stesse (e.g. CMS). Tuttavia queste informazioni non sono sempre disponibili.

- **Algoritmi di clustering basati sul Web Log.** Ricadono in questa classe gli algoritmi che analizzano ed estraggono informazioni a partire da web log. Tali algoritmi possono essere utilizzati in applicazioni per raggruppare pagine web in funzione di pattern comportamentali degli utenti o raggruppare utenti aventi simili comportamenti durante la navigazione sulle pagine web. In [26] viene considerata la cronologia di navigazione ed il tempo di visualizzazione di ogni pagina per raggruppare i profili utenti. I cluster così estratti possono essere utilizzati per migliorare l'esperienza di navigazione degli utenti [9]. Per esempio nel sito web di un dipartimento, i professori effettueranno percorsi di navigazione differenti da quelli degli studenti. Tuttavia raggruppare le pagine web in base ai differenti profili utente può presentare delle difficoltà e ad ogni profilo utente potrebbero corrispondere cluster differenti di pagine web.
- **Algoritmi di clustering basati sulla struttura HTML.** La struttura interna alle pagine web può essere analizzata per scovare pattern ricorrenti nei tag. Infatti le pagine HTML presentano tag innestati in maniera gerarchica che formano una struttura ad albero, solitamente denominata DOM (Document Object Model) [17]. Questa può essere utilizzata per costruire una funzione di similarità tra pagine che presentano la stessa porzione di sottoalbero.

In [10] gli autori propongono un algoritmo di estrazione di pagine web dello stesso tipo semantico (e.g. professori, prodotti, libri, etc.). Gli autori si basano sull'assunzione che pagine web dello stesso tipo semantico sono caratterizzate da una simile struttura. Tuttavia, la soluzione proposta necessita di almeno una pagina campione per ogni cluster da estrarre. Questo approccio quindi richiede una fase di etichettamento delle pagine campione.

In SiteMap Generator [14] viene affrontato un argomento diverso, ovvero la generazione della sitemap di un sito web. Tuttavia anche in questo caso vengono considerati alcuni sotto-alberi del DOM delle pagine web all'interno di uno stesso dominio, per identificare gli hyperlink necessari a determinare la sitemap.

- **Algoritmi di clustering basati sulla struttura ad hyperlink.** Come introdotto precedentemente, la struttura del Web suggerisce l'applicazione di algoritmi capaci di analizzare il grafo web ricavato a partire dagli hyperlink. In [15] gli autori propongono una soluzione di clustering di pagine web attraverso tecniche di partizionamento del grafo web. Obiettivo dell'algoritmo è quello di dividere il grafo iniziale in sotto-grafi, tramite una funzione che minimizza il numero di archi tra cluster e massimizza il numero degli archi tra i nodi di un cluster. Tuttavia il problema del partizionamento dei grafi ha complessità NP-completa [11]. Per superare questo problema in [2] è proposto l'algoritmo Spectral Clustering per partizionare in modo efficiente il grafo in una sequenza di sotto-grafi sempre più piccoli.

Ci sono delle limitazioni nell'utilizzo di algoritmi di partizionamento, in quanto considerano unicamente la struttura connessa del grafo.

3.1 Liste da sorgenti dati strutturate

L'estrazione delle liste di hyperlink dalle pagine web per ricavare dati strutturati è stata già affrontata in letteratura. Liste web sono collezioni strutturate

di hyperlink si ripetono in molte pagine all'interno di uno stesso domino e provengono da sorgenti di dati strutturati, come ad esempio un database relazionale. Queste si rivelano molto utili per filtrare la navigazione web o per identificare i blocchi di tag HTML con alto contenuto informativo.

In [14] viene diviso il DOM di ogni pagina web in sotto-alberi, chiamati blocchi, in seguito classificati in base al contenuto. I blocchi candidati a formare la sitemap del sito vengono identificati come quelli che presentano una alta percentuale di hyperlink.

In [10] gli autori affermano che gli hyperlink contenuti nelle liste web (chiamate "link collections") puntano in generale a pagine dello stesso tipo semantico e che liste web distribuite su diverse pagine web ed aventi simile struttura e rendering punteranno a pagine semanticamente simili. Gli autori utilizzano tale assunzione per migliorare il clustering di pagine web.

3.2 Teoria dei Random Walk

Le informazioni derivanti dalla struttura connessa degli hyperlink di un sito web cela un elevato contenuto informativo. Tuttavia le dimensioni che il Web può raggiungere possono scoraggiare l'utilizzo di metodologie derivanti dalla teoria dei grafi, in quanto queste utilizzano tecniche selettive, ovvero che esplorano tutte le possibili opzioni per avere la sicurezza di arrivare, prima o poi, alla soluzione. Questi algoritmi possono essere difficilmente computabili in quanto ricadono nella classe di complessità NP-completa [11].

Nell'ambito della Social Networks Analysis, DeepWalk [21] propone una metodologia interessante. Dato un grafo, vengono generati Random Walk (riportati in sezione 2.2.1) di piccola lunghezza. Questi vengono poi trattati come frasi, e applicando tecniche di Natural Language Processing viene stimata la verosimiglianza che specifiche sequenze di parole (in questo caso i nodi del grafo) appaiano nel corpus, ovvero l'insieme dei Random Walk generati. I nodi del grafo vengono quindi rappresentati in uno spazio vettoriale. DeepWalk è applicato nell'ambito delle reti sociali per l'identificazione di

gruppi sociali correlati, tecnica che viene chiamata "Community Detection". In [32] viene presentato un modello gerarchico dei documenti, basato sui topic riscontrati. Nei livelli più alti saranno presenti documenti riguardanti topic più generali, che saranno specializzati nei documenti discendenti nell'albero gerarchico. L'estrazione del modello è realizzata attraverso la nozione di autorità, rilevata attraverso l'utilizzo di Random Walk con restart alla homepage. Su questi si assume che l'elevata autorità corrisponda al trattamento di argomenti più generali.

Capitolo 4

Sperimentazione

In questo capitolo si descriverà la progettazione e l'esecuzione della sperimentazione.

Obiettivo di questo capitolo è quello di valutare l'efficacia dell'approccio realizzato al fine di comprendere se e come i dati strutturati, di cui un sito web si compone, possano essere combinati con dati non strutturati, come quelli testuali, al fine di migliorare il processo di clustering e valutare in dettaglio le cause di un eventuale successo o insuccesso delle tecniche utilizzate. A tal fine si confronteranno le performance degli algoritmi di clustering basati su rappresentazione a grafo con algoritmi di clustering basati su rappresentazione vettoriale contenente informazioni sulla struttura del grafo stesso.

Questa tesi rappresenta un lavoro preliminare utile a comprendere la bontà della soluzione proposta e quindi decidere se proseguire gli studi in questa direzione o definire strategie alternative.

Si descrivono di seguito i dataset e le configurazioni degli algoritmi di clustering su cui sono realizzate le sperimentazioni e le metriche utilizzate per valutare la qualità dei cluster estratti. Infine verranno mostrati una serie di grafici e tavole utili a comparare i risultati ottenuti al variare delle configurazioni.

4.1 Dataset

La sperimentazione è stata effettuata sfruttando i dati provenienti da siti web appartenenti a tre importanti dipartimenti di Computer Science:

- Dipartimento di Computer Science dell'università di Urbana, IL:
www.cs.illinois.edu
Il crawling è stato lanciato con profondità massima 10, e immagazzinando un massimo di 10.000 termini per ogni pagina esplorata. Il grafo web estratto si compone di 728 pagine web e 16993 hyperlink. Dal grafo sono state poi generate 10.000 sequenze di lunghezza massima 10.
- Dipartimento di Computer Science dell'università di Stanford, CA:
cs.stanford.edu
Il crawling è stato effettuato con profondità massima 10, e immagazzinando un massimo di 10.000 termini per ogni pagina esplorata. Il grafo web estratto si compone di 1458 pagine web e 99686 hyperlink. Dal grafo sono state poi generate 10.000 sequenze di lunghezza massima 10.
- Dipartimento di Computer Science del Massachusetts Institute of Technology a Cambridge, MA: eecs.mit.edu
Il crawling è stato lanciato con profondità massima 10, e immagazzinando un massimo di 10.000 termini per ogni pagina esplorata. Il grafo web estratto si compone di 1745 pagine web e 63937 hyperlink. Dal grafo sono state poi generate 10.000 sequenze di lunghezza massima 10.
- Dipartimento di Computer Science dell'università di Princeton, NJ:
cs.princeton.edu
Il crawling è stato lanciato con profondità massima 7, e immagazzinando un massimo di 10.000 termini per ogni pagina esplorata. Il grafo web estratto si compone di 16378 pagine web e 206985 hyperlink. Dal grafo sono state poi generate 10.000 sequenze di lunghezza massima 10.
- Dipartimento di Computer Science dell'università di Oxford, UK:
cs.ox.ac.uk
Il crawling è stato lanciato con profondità massima 5, e immagazzinando un massimo di 10.000 termini per ogni pagina esplorata. Il grafo web estratto si compone di 1000 pagine web e 10000 hyperlink. Dal grafo sono state poi generate 10.000 sequenze di lunghezza massima 5.

do un massimo di 10.000 termini per ogni pagina esplorata. Il grafo web estratto si compone di 4183 pagine web e 27954 hyperlink. Dal grafo sono state poi generate 10.000 sequenze di lunghezza massima 10.

La scelta dei precedenti siti è dovuta a ragioni di confidenza con il dominio applicativo. Infatti la valutazione dei risultati sperimentali ha richiesto, oltre che il crawling dei siti web sopracitati anche una fase di etichettatura delle pagine web al fine di generare una ground truth.

Inoltre per ogni sito web sono stati estratti due differenti grafi web:

- **Web graph no-constraint (nc).** Il grafo web $G_{nc} = (V, E)$ rappresenta fedelmente il sito web. In particolare V rappresenta l'insieme delle pagine web appartenenti al sito ed E è l'insieme di tutte le coppie (i, j) per i quali la pagina con url i contiene un outlink alla pagina con url j .
- **Web graph list-constraint (lc).** Il grafo web $G_{ls} = (V, E)$ è filtrato eliminando tutti quegli archi $(i, j) \in E$ per i quali l'url j non è contenuto in nessuna lista web nella pagina con url i (vedi Sezione [metti sezione](#)).

4.2 Metriche

Valutare le performance di un algoritmo di clustering non è semplice come contare il numero di errori o calcolare metriche quali *Precision* e *Recall* di un algoritmo di apprendimento supervisionato. In particolare, le metriche di valutazione non dovrebbero prendere in considerazione gli specifici valori delle label. Queste dovrebbero piuttosto considerare se il raggruppamento generato dall'algoritmo definisce una separazione dei dati similmente a quanto fornito nella *ground truth*, ovvero il vero valore delle label, o soddisfare qualche assunzione, come ad esempio che membri dello stesso cluster siano più simili rispetto a quelli di cluster differenti, utilizzando una data funzione

di similarità. Di seguito si elencano le metriche utilizzate per valutare la qualità dei cluster prodotti.

- **Omogeneità:** nota la ground truth (ossia l'insieme dei cluster ideali), questo valore rappresenta quanto ogni cluster appreso sia omogeneo, ossia quanto gli elementi dello stesso cluster appartengono alla stessa classe. Formalmente l'omogeneità può essere definita come segue:

$$h = 1 - \frac{H(C|K)}{H(C)}; \quad (4.1)$$

dove $H(C|K)$ è l'entropia condizionata delle classi date le assegnazioni dei cluster e $H(C)$ è l'entropia delle classi, ossia:

$$H(C|K) = - \sum_{c=1}^{|C|} \sum_{k=1}^{|K|} \frac{n_{c,k} n}{\cdot} \log \left(\frac{n_{c,k}}{n_k} \right) \quad (4.2)$$

$$H(C) = \sum_{c=1}^{|C|} \frac{n_c}{n} \cdot \log \left(\frac{n_c}{n} \right) \quad (4.3)$$

con n il numero totale delle pagine, e n_c ed n_k il numero delle pagine appartenenti alla classe c o al cluster k , ed infine $n_{c,k}$ sono le pagine della classe c assegnate al cluster k . In una situazione ideale l'omogeneità assume valore 1. [23]

- **Completezza:** è una misura simmetrica all'Omogeneità. In modo da soddisfare i criteri di completezza un algoritmo di clustering deve assegnare tutti gli elementi di una data classe ad un unico cluster. Per valutare la completezza è necessario esaminare la distribuzione degli assegnamenti di ogni classe. In particolare in una soluzione perfetta ogni distribuzione di classe dovrebbe convergere in un cluster. Formalmente la *Completezza* può essere così formulata:

$$c = 1 - \frac{H(C|K)}{H(K)} \quad (4.4)$$

dove $H(C|K)$ è definita nell'equazione 4.2 e $H(K)$ è l'entropia dei cluster. Come nel caso dell'Omogeneità la Completezza di una soluzione ideale assume valore 1.

- **V-Measure:** è una misura basata sull'entropia che esplicitamente calcola quanto i criteri di omogeneità e completezza sono stati soddisfatti. Essa è computata come la media armonica fra Omogeneità e Completezza, così come Precisione e Richiamo sono combinati nel calcolo del F-Measure:

$$v = 2 \cdot \frac{h \cdot c}{h + c} \quad (4.5)$$

- **Adjusted Rand Index (ARI):** nota la ground truth e gli assegnamenti restituiti da un algoritmo di clustering, l'ARI calcola la similarità tra i due insiemi ignorando le permutazioni [25].

Da un punto di vista matematico l'ARI misura l'accuratezza del clustering, ossia la percentuale di coppie di oggetti per i quali due algoritmi di clustering concordano sull'assegnazione. In particolare, dato C l'insieme degli assegnamenti ideali (ground truth) e K l'insieme delle label predette, allora è possibile calcolare il **Random Index**:

$$RI = \frac{a + b}{C_2^n} \quad (4.6)$$

Dove:

- a è il numero di coppie di elementi che si trovano sia in c che in K
- b è il numero di coppie di elementi che si trovano in insiemi diversi in C e in insiemi diversi in K .
- C_2^n è il numero totale di tutte le possibili coppie nel dataset.

Un problema del RI è che cluster cumputati in modo casuale non assumono un indice costante (per esempio 0).

A tal fine è definito l'Adjusted Random Index:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]} \quad (4.7)$$

Questa misura garantisce che cluster generati in modo casuale abbiano un valore ARI prossimi allo 0. Differentemente dalle precedenti misure l'ARI assume valori tra -1 e 1 .

- **Mutual Information (MI):** nota la ground truth e le assegnazioni dell'algoritmo di clustering, la *Mutual Information* è una funzione che misura la corrispondenza delle due informazioni, ignorando le permutazioni.

Anche qui, una assegnazione uniforme (o casuale) dei cluster avrà valori prossimi allo 0, evidenziando l'indipendenza delle due informazioni, mentre valori prossimi a 1 indicano una corrispondenza significativa.

Dati due assegnamenti di pari lunghezza, U e V , la loro entropia è la quantità di incertezza per una partizione, definita da:

$$H(U) = \sum_{i=1}^{|U|} P(i) \log(P(i)) \quad (4.8)$$

Dove $P(i) = \frac{|U_i|}{N}$ è la probabilità che un oggetto preso a caso da U appartenga alla classe U_i . Similmente per V :

$$H(V) = \sum_{j=1}^{|V|} P'(j) \log(P'(j)) \quad (4.9)$$

Con $P'(j) = \frac{|V_j|}{N}$. La Mutual Information p definita come:

$$MI(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P(i, j) \log \left(\frac{P(i, j)}{P(i)P'(j)} \right) \quad (4.10)$$

Per le metriche riportate è degno di nota che nessuna assunzione viene fatta sulla struttura dei cluster, quindi possono essere utilizzate con algoritmi che

identificano cluster di forma diversa.

Queste, ad eccezione dell'ARI, essendo misure probabilistiche i valori possibili spaziano nell'intervallo (0, 1).

Silhouette : misura la forza di ogni cluster [24]. Essa può essere usata per calcolare la qualità dei cluster anche quando la Ground Truth non è nota. Formalmente la Silhouette può essere così definita:

$$s = \frac{b - a}{\max(a, b)} \quad (4.11)$$

dove:

- a = la distanza media tra un vettore e tutti gli altri nella stessa classe
- b = la distanza media tra un vettore e tutti gli altri nella classe più vicina

La Silhouette può variare da -1 , per cluster non definiti, a 1 per cluster densamente connessi. Un valore intorno allo 0 indica cluster sovrapposti. In generale cluster convessi o molto densi come quelli ottenuti con DBSCAN, assumono alti valori.

4.3 Configurazioni

Si descrivono di seguito le configurazioni utilizzate per gli algoritmi di clustering utilizzati durante la fase di sperimentazione.

4.3.1 Community Detection

Per gli algoritmi di clustering basati su grafo sono stati utilizzati *Walk Trap* e *Fastgreedy*. Questi rappresentano gli algoritmi più comunemente utilizzati in applicazioni come *Social Network Mining* per estrarre Web Community.

Tabella 4.1: My caption

Web site	ran walk len	dendr.cut	num. real clusters
cs.illinois.edu	3	15	16
cs.stanford.edu	3	10	13
eeecs.mit.edu	3	15	16
cs.princeton	4	20	29
cs.ox.ac.uk	4	20	26

Fastgreedy non richiede parametri specifici, tuttavia per ogni sito web è stata definita un'altezza entro cui tagliare il dendogramma in modo tale che il numero dei cluster restituiti sia simile al numero dei cluster della ground truth. WalkTrap invece richiede in input la lunghezza dei Random Walk.

La tabella 4.1 mostra per ogni sito web oggetto di analisi i parametri utilizzati.

4.3.2 URL Embedding

Considerando i Random Walk generati sul grafo come frasi, è possibile applicare algoritmi di Word Embedding per raggruppare le pagine sulla base del contesto in cui appaiono, ovvero le pagine che più verosimilmente appariranno insieme nelle sequenze.

Le sequenze di Random Walk sono state usate per apprendere rappresentazioni vettoriali delle pagine Web. La fase di URL embedding è stata effettuata utilizzando l'algoritmo Word2vec, [22] (esaminato in Sezione 1.3.2) modificando alcuni parametri e lasciando invariato altri.

I parametri personalizzati sono:

- **min-count**: tutte le parole (o URL) con frequenza di occorrenza minore di questo valore vengono ignorate.
- **window** rappresenta la distanza massima tra l'URL corrente e quello predetto all'interno di una frase.

- **negative** Nella fase di embedding di una URL, viene calcolato il rapporto tra la similarità del contesto con la parola e la sommatoria di tutte le similarità tra la parola e gli altri contesti. Più precisamente:

$$\frac{v_c \cdot v_w}{\sum_{c \in C} v_c \cdot v_w} \quad (4.12)$$

Questa operazione può essere molto lenta. Per accelerare il processo possono venir scelti n contesti casuali da confrontare. Questo parametro, se maggiore di 0, rappresenta il numero di vettori da confrontare.

- **sg** definisce l'algoritmo di apprendimento, di default viene usato *CBOW*, mentre se impostato a 1 utilizza *skip-gram* [18]. Per lo scopo della sperimentazione si imposterà questo valore a CBOW.

Sulle rappresentazioni vettoriali estratte sono stati applicati gli algoritmi HDBSCAN e K-MEANS. Di seguito si definiscono i parametri utilizzati per ogni sito web:

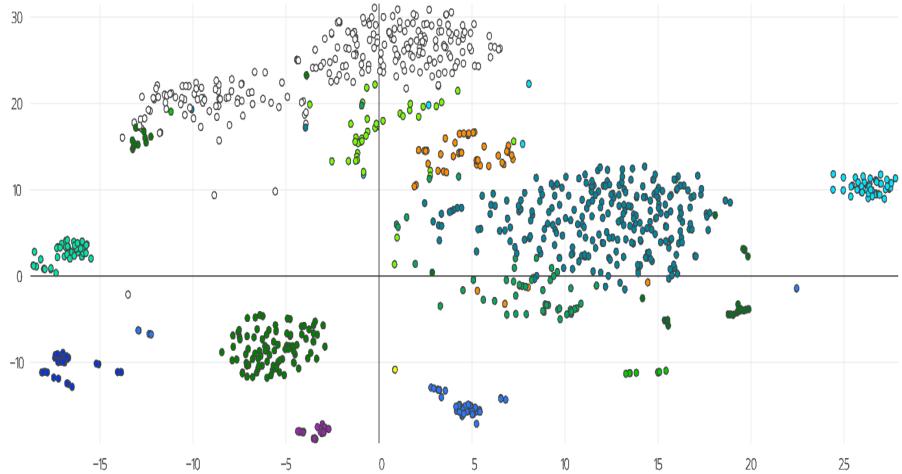


Figura 4.1: Rappresentazione del sito cs.illinois.edu, clusterizzato con K-Means.

- **cs.illinois.edu** è stata impostato un *min-count* pari a 1, *window* con valore 5 ed è stato utilizzato *skip-gram* con 5 *negative sampling*. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 4; HDBSCAN con *min-cluster-size*= 6; K-Means con numero di cluster pari a 15.
- **cs.stanford.edu** è stata impostato un *min-count* pari a 1, *window* con valore 5 ed è stato utilizzato *skip-gram* con 5 *negative sampling*. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 7; HDBSCAN con *min-cluster-size*= 10; K-Means con numero di cluster pari a 10. Per il dataset delle liste: DBSCAN con $\epsilon = 1.6$ ed *min-samples*= 3; HDBSCAN con *min-cluster-size*= 5; K-Means con numero di cluster pari a 10.
- **eecs.mit.edu** è stata impostato un *min-count* pari a 1, *window* con valore 5 ed è stato utilizzato *skip-gram* con 5 *negative sampling*. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 1.0$ ed *min-samples*= 10; HDBSCAN con *min-cluster-size*= 10; K-Means con numero di cluster pari a 15. Per il dataset delle liste: DBSCAN con $\epsilon = 1.5$ ed *min-samples*= 5; HDBSCAN con *min-cluster-size*= 13; K-Means con numero di cluster pari a 15.
- **cs.princeton.edu** è stata impostato un *min-count* pari a 1, *window* con valore 5 ed è stato utilizzato *skip-gram* con 5 *negative sampling*. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.8$ ed *min-samples*= 7; HDBSCAN con *min-cluster-size*= 10; K-Means con numero di cluster pari a 30. Per il dataset delle liste: DBSCAN con $\epsilon = 0.7$ ed *min-samples*= 5; HDBSCAN con *min-cluster-size*= 8; K-Means con numero di cluster pari a 25.
- **cs.ox.ac.uk** è stata impostato un *min-count* pari a 1, *window* con valore 5 ed è stato utilizzato *skip-gram* con 5 *negative sampling*. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.8$ ed *min-samples*= 7; HDBSCAN con *min-cluster-size*= 10; K-Means con numero di cluster pari a 30. Per il dataset delle liste: DBSCAN con $\epsilon = 0.7$ ed *min-*

samples= 5; HDBSCAN con *min-cluster-size*= 8; K-Means con numero di cluster pari a 25.

4.3.3 Text Mining

Sono state utilizzate tecniche di Text Mining per il clustering basato sul contenuto testuale. I contenuti all'interno di uno stesso sito web avranno una struttura e termini comuni, differenziandosi al variare dell'argomento trattato. La struttura gerarchica di un sito web organizza solitamente le pagine in sezioni simili. Questa metodologia tuttavia, considera solo l'informazione testuale, assumendo che i termini all'interno del sito web siano indipendenti l'uno dall'altro così come i documenti, ignorando le relazioni interdipendenti tra questi. Il web si discosta dall'analisi classica dei documenti proprio per le relazioni che intercorrono tra le pagine, tuttavia l'analisi testuale rimane molto importante.

Nella fase di sperimentazione è stata utilizzata una rappresentazione vettoriale della frequenza dei termini all'interno dell'insieme delle pagine web, calcolata con la tecnica della *frequency-inverse document frequency* (tf-idf).

I parametri personalizzati per la costruzione della matrice documenti-termini con funzione di peso *idf* sono stati:

- **max-df**: questo valore rappresenta la massima frequenza, all'interno dei documenti, che un termine può avere per essere utilizzato nella matrice tf-idf. Se un termine appare molte volte nel corpus, molto probabilmente avrà poco significato.
- **min-df**: indica il numero minimo di documenti in cui un termine dovrà apparire per essere considerato.
- **ngram-range**: vengono presi in considerazioni gli n-grammi di lunghezza compresa nell'intervallo specificato in questo parametro. Un n-gramma è una sottosequenza di n elementi di un'altra.

I cluster estratti sono stati ottenuti nel seguente modo:

- **cs.illinois.edu** Sul dataset del sito, costituito da 728 pagine e 433 termini, il corpus è stato ripulito delle stopword, stemmatizzato ed è stato impostato il *max-df* all'80%, il *min-df* a 0.1 e sono stati considerati solo uni-grammi, bi-grammi e tri-grammi. Se i termini appaiono in più dell'80% dei documenti, probabilmente avrà poco significato, lo stesso se appare troppe poche volte. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 4; HDBSCAN con *min-cluster-size*= 4; K-Means con numero di cluster pari a 15.
 Sul Dataset costruito estraendo le liste: DBSCAN con $\epsilon = 0.7$ ed *min-samples*= 4; HDBSCAN con *min-cluster-size*= 7; K-Means con numero di cluster pari a 15.
- **cs.stanford.edu** Sul dataset del sito, costituito da 1458 pagine e 843 termini, il corpus è stato ripulito delle stopword, stemmatizzato ed è stato impostato il *max-df* all'80%, il *min-df* a 0.1 e sono stati considerati solo uni-grammi, bi-grammi e tri-grammi. Se i termini appaiono in più dell'80% dei documenti, probabilmente avrà poco significato, lo stesso se appare troppe poche volte. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.3$ ed *min-samples*= 5; HDBSCAN con *min-cluster-size*= 15; K-Means con numero di cluster pari a 10.
 Sul Dataset costruito estraendo le liste: DBSCAN con $\epsilon = 0.5$ ed *min-samples*= 3; HDBSCAN con *min-cluster-size*= 5; K-Means con numero di cluster pari a 10.
- **eecs.mit.edu** Sul dataset del sito, costituito da 1745 pagine e 354 termini, il corpus è stato ripulito delle stopword, stemmatizzato ed è stato impostato il *max-df* all'80%, il *min-df* a 0.1 e sono stati considerati solo uni-grammi, bi-grammi e tri-grammi. Se i termini appaiono in più dell'80% dei documenti, probabilmente avrà poco significato, lo stesso se appare troppe poche volte. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 9; HDBSCAN con *min-cluster-size*= 15; K-Means con numero di cluster pari a 15.
 Sul Dataset costruito estraendo le liste: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 6; HDBSCAN con *min-cluster-size*= 9; K-Means con numero

di cluster pari a 15.

- **cs.princeton.edu** Sul dataset del sito, costituito da 16378 pagine e 470 termini, il corpus è stato ripulito delle stopword, stemmatizzato ed è stato impostato il *max-df* all'80%, il *min-df* a 0.1 e sono stati considerati solo uni-grammi, bi-grammi e tri-grammi. Se i termini appaiono in più dell'80% dei documenti, probabilmente avrà poco significato, lo stesso se appare troppe poche volte. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.3$ ed *min-samples*= 5; HDBSCAN con *min-cluster-size*= 15; K-Means con numero di cluster pari a 20.

Sul Dataset costruito estraendo le liste: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 6; HDBSCAN con *min-cluster-size*= 15; K-Means con numero di cluster pari a 20.

- **cs.ox.ac.uk** Sul dataset del sito, costituito da 3951 pagine e 363 termini, il corpus è stato ripulito delle stopword, stemmatizzato ed è stato impostato il *max-df* all'80%, il *min-df* a 0.1 e sono stati considerati solo uni-grammi, bi-grammi e tri-grammi. Se i termini appaiono in più dell'80% dei documenti, probabilmente avrà poco significato, lo stesso se appare troppe poche volte. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.3$ ed *min-samples*= 5; HDBSCAN con *min-cluster-size*= 15; K-Means con numero di cluster pari a 20.

Sul Dataset costruito estraendo le liste: DBSCAN con $\epsilon = 0.7$ ed *min-samples*= 7; HDBSCAN con *min-cluster-size*= 7; K-Means con numero di cluster pari a 20.

4.3.4 Embedding e Text Mining

La terza ed ultima configurazione analizzata è rappresentata dalla combinazione nella rappresentazione vettoriale di informazioni riguardanti il contenuto testuale delle pagine web e la struttura del sito web. Questo è realizzato attraverso la combinazione di rappresentazioni vettoriali ottenute con algoritmi di Word Embedding e rappresentazioni vettoriali ottenute con algoritmi derivanti dall'analisi del contenuto testuale.

Di seguito si definiscono i parametri e gli algoritmi utilizzati in questa configurazione:

- **cs.illinois.edu** I vettori di word2vec sono stati generati di dimensione 48, mentre i vettori-riga documenti sono stati ridotti con *TruncateSVD* a dimensione 50. Gli algoritmi testati sono stati: HDBSCAN con *min-cluster-size*= 7; K-Means con numero di cluster pari a 15.
- **cs.stanford.edu** I vettori di word2vec sono stati generati di dimensione 48, mentre i vettori-riga documenti sono stati ridotti con *TruncateSVD* a dimensione 50. Gli algoritmi testati sono stati: HDBSCAN con *min-cluster-size*= 10; K-Means con numero di cluster pari a 15.
- **eecs.mit.edu** I vettori di word2vec sono stati generati di dimensione 48, mentre i vettori-riga documenti sono stati ridotti con *TruncateSVD* a dimensione 50. Gli algoritmi testati sono stati: HDBSCAN con *min-cluster-size*= 14; K-Means con numero di cluster pari a 15.
- **cs.princeton.edu** I vettori di word2vec sono stati generati di dimensione 48, mentre i vettori-riga documenti sono stati ridotti con *TruncateSVD* a dimensione 50. Gli algoritmi testati sono stati: HDBSCAN con *min-cluster-size*= 14; K-Means con numero di cluster pari a 20.
- **cs.ox.ac.uk** I vettori di Word2vec sono stati generati di dimensione 48, mentre i vettori-riga documenti sono stati ridotti con *TruncateSVD* a dimensione 50. Gli algoritmi testati sono stati: HDBSCAN con *min-cluster-size*= 14; K-Means con numero di cluster pari a 20.

4.4 Analisi dei risultati

Di seguito si confrontano i risultati degli algoritmi di clustering basati su grafo con quelli basati su rappresentazioni vettoriali al variare del grafo web (list-constraint vs no-constraint). Inoltre per gli algoritmi di clustering basati su vettori si analizzano le prestazioni degli algoritmi al varia-

re delle rappresentazioni vettoriali (Url-embedding, Tf-idf, Url-embeddig + tf-idf).

La divisione del grafo in Community può essere effettuata seguendo diversi approcci, tuttavia la rete costruita su un sito web è caratterizzata solitamente da un numero elevato di collegamenti tra pagine, il quale suggerisce l'utilizzo di alcune tipologie piuttosto che altre. Infatti misure come la betweenness non hanno restituito risultati significativi. Dalla scelta di apprendere le rappresentazioni vettoriali delle relazioni invece di utilizzare algoritmi di Community Detection del grafo, sono derivati dei vantaggi. Innanzitutto gli algoritmi di partizionamento dei grafi hanno complessità NP-completa, ovvero necessitano di un tempo superpolinomiale nella dimensione dell'input. Nel contesto del Web Mining la dimensione del dataset può crescere enormemente ed avere soluzioni più efficienti costituisce senz'altro una priorità.

In tutti i casi di seguito riportati sono stati generati grafi sia in modalità classica, che attraverso l'estrazione delle liste. Nel primo caso i dataset saranno chiamati **"nc"** (no-constraint, senza vincoli), mentre nel secondo caso **"lc"** (list-constraint, con il vincolo delle liste (vedere sezione 2.1.2)). Dove omessa, la configurazione è rimasta invariata.

`cs.illinois.edu`

Illinois	Hom	Com	V-M	ARI	MI	Silh
G-nc WT	0.6471	0.6585	0.6527	0.4363	0.6281	//
G-nc FG	0.5518	0.8563	0.6711	0.5764	0.5354	//
G-lc WT	0.5093	0.4892	0.4991	0.2762	0.4722	//
G-lc FG	0.5522	0.6035	0.5767	0.3656	0.5382	//
E-nc dbSCAN	0.5553	0.6579	0.6023	0.4487	0.5234	0.2588
E-nc hdbSCAN	0.5759	0.6720	0.6203	0.5282	0.5525	0.2573
E-nc Kmeans	0.8238	0.7575	0.7892	0.7883	0.7423	0.3131
E-lc dbSCAN	0.4163	0.5922	0.4889	0.2250	0.3935	0.1320
E-lc hdbSCAN	0.4760	0.5067	0.4908	0.2275	0.4515	0.1054
E-lc Kmeans	0.8095	0.6593	0.7267	0.6189	0.6473	0.2281
T-nc dbSCAN	0.5601	0.5962	0.5776	0.4078	0.5346	0.1242
T-nc hdbSCAN	0.5152	0.6029	0.5556	0.3862	0.4858	0.0881
T-nc Kmeans	0.7619	0.5814	0.6596	0.3184	0.5586	0.1767
T-lc dbSCAN	0.5710	0.7042	0.6306	0.5197	0.5566	0.1406
T-lc hdbSCAN	0.4501	0.5104	0.4783	0.1938	0.4297	0.1018
T-lc Kmeans	0.8061	0.5892	0.6808	0.4296	0.5748	0.2002
ET-nc hdbSCAN	0.7327	0.7534	0.7429	0.7204	0.7186	0.2070
ET-nc Kmeans	0.8812	0.8069	0.8424	0.8299	0.7949	0.3198
ET-lc hdbSCAN	0.6541	0.6129	0.6328	0.3249	0.5992	0.1203
ET-lc Kmeans	0.8548	0.6885	0.7627	0.6488	0.6773	0.2573

Tabella 4.2: Risultati sperimentazione del sito `cs.illinois.edu`

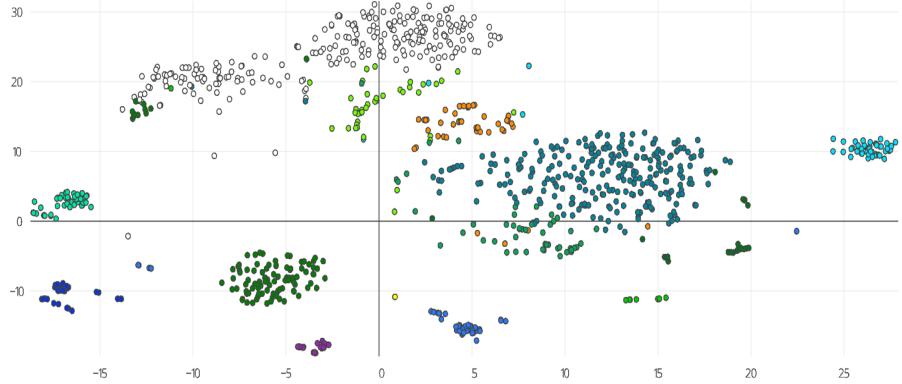


Figura 4.2: Rappresentazione del sito `cs.illinois.edu`, clusterizzato con K-Means.

I risultati hanno evidenziato che l’analisi singola, sia della correlazione tra le pagine sia del contenuto testuale, può non bastare a codificare esaustivamente la conoscenza che una pagina web può offrire. Entrambe le informazioni sono rilevanti ed andrebbero processate insieme in quanto le informazioni codificate nelle due tipologie di vettori sono complementari. Le sperimentazioni effettuate sul grafo intero (**nc**) e sul grafo costruito unicamente con gli hyperlink nelle liste (**lc**), hanno restituito risultati migliori delle altre configurazioni. Tuttavia le liste in questo caso non hanno migliorato la qualità dei Random Walk. Le pagine hanno formato cluster sferici e abbastanza separati, rendendo l’algoritmo K-Means il più performante. Ad un’analisi più dettagliata, è emerso che pagine etichettate diversamente sono state raggruppate insieme secondo alcuni criteri ragionevoli. Ad esempio pagine relative agli studenti triennali, ma riguardanti sezioni diverse, sono state raggruppate nello stesso cluster, così come quelle relative agli studenti magistrali. Questo è stato dovuto alla presenza di simili distribuzioni dei termini ed alla presenza negli stessi Random Walk. Infine, erano presenti diverse versioni della stessa pagina, ad esempio relative agli stessi professori. Queste presentavano una diversa distribuzione di termini ed apparivano in percorsi diversi e sono state raggruppate in cluster diversi.

cs.stanford.edu

Stanford	Hom	Com	V-M	ARI	MI	Silh
G-nc WT	0.1628	0.3967	0.2308	0.1127	0.1340	//
G-nc FG	0.3568	0.3894	0.3724	0.2439	0.3357	//
G-lc WT	0.3087	0.6314	0.4146	0.0502	0.1911	//
G-lc FG	0.4854	0.6740	0.5644	0.2999	0.3892	//
E-nc dbSCAN	0.3453	0.4031	0.3720	0.2577	0.3252	0.4058
E-nc hdbSCAN	0.4720	0.4418	0.4564	0.3094	0.4210	0.4030
E-nc Kmeans	0.3810	0.3844	0.3827	0.2276	0.3574	0.5659
E-lc dbSCAN	0.4830	0.5798	0.5270	0.0976	0.3363	0.0938
E-lc hdbSCAN	0.6506	0.6299	0.6353	0.2466	0.5014	0.1762
E-lc Kmeans	0.5853	0.6838	0.6308	0.2423	0.4786	0.2513
T-nc dbSCAN	0.2584	0.3435	0.2949	0.0208	0.2403	0.0981
T-nc hdbSCAN	0.3015	0.2784	0.2895	0.0089	0.2521	0.0916
T-nc Kmeans	0.6014	0.4652	0.5246	0.2639	0.4489	0.2704
T-lc dbSCAN	0.0876	0.3924	0.1432	0.0102	0.0327	-0.1188
T-lc hdbSCAN	0.1827	0.3802	0.2468	0.0687	0.1326	0.0841
T-lc Kmeans	0.5436	0.5726	0.5577	0.2167	0.4157	0.1732
ET-nc hdbSCAN	0.3495	0.5272	0.4203	0.3335	0.3307	0.3469
ET-nc Kmeans	0.3490	0.3629	0.3558	0.2061	0.3238	0.4336
ET-lc hdbSCAN	0.2343	0.7045	0.3517	0.1012	0.2033	0.2342
ET-lc Kmeans	0.7508	0.7422	0.7465	0.5123	0.6656	0.2375

Tabella 4.3: Risultati sperimentazione del sito cs.stanford.edu

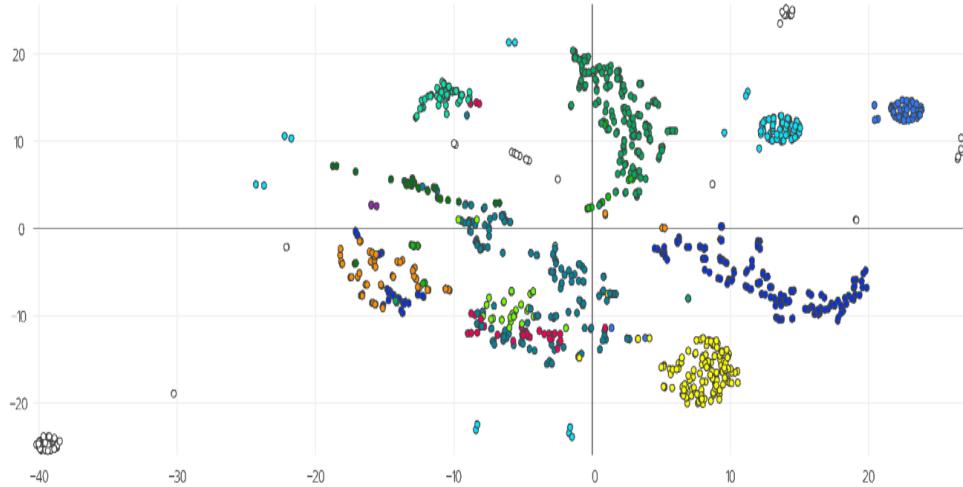


Figura 4.3: Rappresentazione del sito `cs.stanford.edu`, clusterizzato con K-Means.

Nonostante il dataset del grafo completo presenti un maggiore coefficiente di Silhouette, l'applicazione delle liste hanno restituito risultati migliori. In questo caso hanno filtrato in modo consistente le pagine esplorate in quanto il DOM del sito includeva la maggior parte degli hyperlink in sezione nascoste, prevalentemente menù dropdown. Questo ha garantito una diversificazione marcata dei percorsi. Inoltre i risultati dell'analisi testuale si attestano al di sotto di quelli ottenuti attraverso l'URL embedding.

Anche qui i cluster si sono formati prevalentemente con forma sferica, favorendo il K-Means sugli altri.

eeecs.mit.edu

MIT	Hom	Com	V-M	ARI	MI	Silh
G-nc WT	0.1792	0.6729	0.2830	0.1762	0.1648	//
G-nc FG	0.5769	0.5215	0.5478	0.5514	0.5119	//
G-lc WT	0.0856	0.5013	0.1462	0.0450	0.0664	//
G-lc FG	0.5617	0.7127	0.6282	0.6851	0.5497	//
E-nc dbSCAN	0.4097	0.4236	0.4164	0.3779	0.3869	0.3264
E-nc hdbSCAN	0.5834	0.4413	0.5025	0.5811	0.4231	0.3141
E-nc Kmeans	0.6422	0.5129	0.5703	0.5141	0.5011	0.3914
E-lc dbSCAN	0.1727	0.5736	0.2655	0.1455	0.1550	0.1750
E-lc hdbSCAN	0.7882	0.4456	0.5693	0.5217	0.4061	0.2288
E-lc Kmeans	0.7165	0.5283	0.6081	0.3625	0.5146	0.2646
T-nc dbSCAN	0.2071	0.1908	0.1986	0.1011	0.1759	0.1133
T-nc hdbSCAN	0.2363	0.2679	0.2611	0.0918	0.2207	0.0845
T-nc Kmeans	0.4175	0.2381	0.3032	0.0917	0.2290	0.1958
T-lc dbSCAN	0.3470	0.3466	0.3467	0.2071	0.3326	0.1180
T-lc hdbSCAN	0.4402	0.4147	0.4271	0.2806	0.3995	0.1423
T-lc Kmeans	0.5857	0.4198	0.4890	0.2597	0.4085	0.2296
ET-nc hdbSCAN	0.5573	0.3748	0.4482	0.2845	0.3600	0.1607
ET-nc Kmeans	0.5505	0.4344	0.4856	0.2687	0.4252	0.1699
ET-lc hdbSCAN	0.6076	0.4329	0.5056	0.1910	0.4181	0.1464
ET-lc Kmeans	0.6679	0.5332	0.5930	0.3499	0.5235	0.2311

Tabella 4.4: Risultati sperimentazione del grafo del sito eeecs.mit.edu

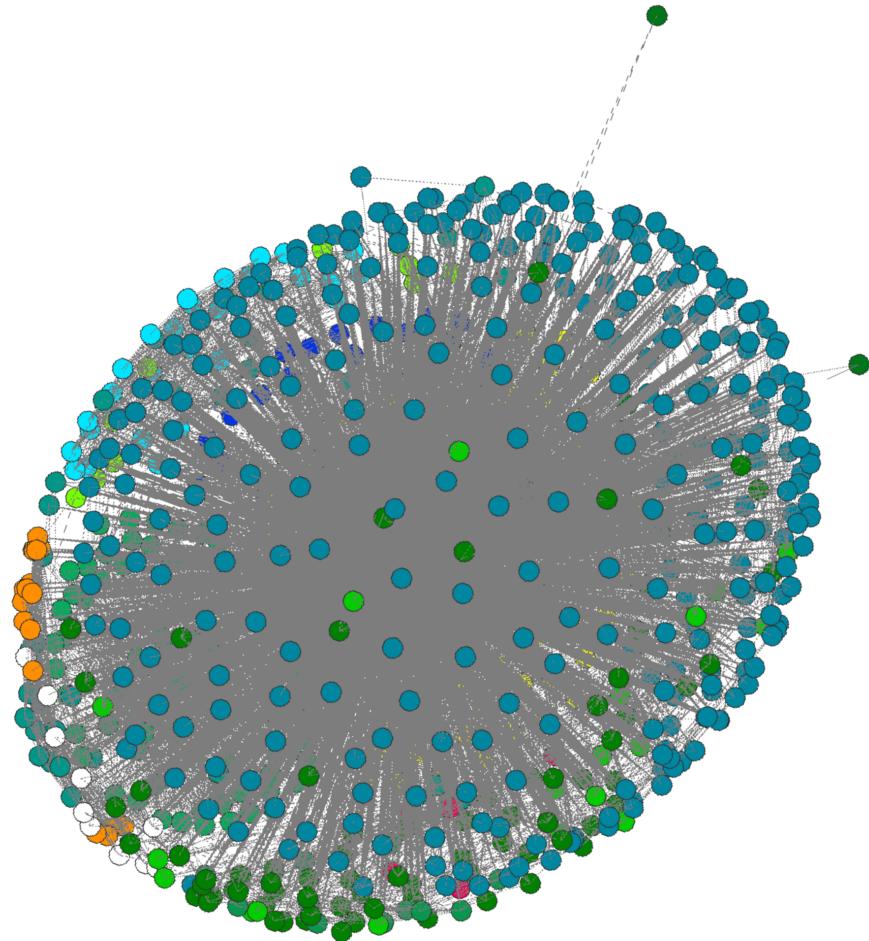


Figura 4.4: Rappresentazione del sito `eecs.mit.edu`, clusterizzato con Fastgreedy.

In questo caso il partizionamento del grafo si è rivelato più efficace. L’embedding degli URL ha comunque restituito una omogeneità dei cluster maggiore. In questo caso i Random Walk, sia nell’algoritmo WalkTrap sul grafo, sia quelli utilizzati per apprendere rappresentazioni vettoriali, non sono riusciti a codificare abbastanza informazioni. La modularità del grafo, utilizzata in Fastgreedy, ha evidenziato una forte divisione fra le Community presenti. Questo è stato dovuto dalle maggiori connessioni tra le pagine all’interno dello stesso cluster, rispetto alle connessioni fra pagine di cluster diversi. Le liste hanno contribuito a filtrare i cammini percorribili.

cs.princeton.edu

Princeton	Hom	Com	V-M	ARI	MI	Silh
G-nc WT	0.0259	0.1568	0.0444	-0.0221	0.0197	//
G-nc FG	0.4900	0.5702	0.5270	0.4594	0.4848	//
G-lc WT	0.1543	0.2983	0.2034	0.1935	0.1375	//
G-lc FG	0.4230	0.3817	0.4013	0.2168	0.3686	//
E-nc dbSCAN	0.0955	0.2417	0.1369	0.0622	0.0886	0.2656
E-nc hdbSCAN	0.2302	0.3292	0.2710	0.1651	0.2195	0.1687
E-nc Kmeans	0.4311	0.3637	0.3946	0.3404	0.3522	0.3877
E-lc dbSCAN	0.5941	0.5696	0.5816	0.7706	0.5579	0.2269
E-lc hdbSCAN	0.6531	0.5416	0.5922	0.7708	0.5275	0.1789
E-lc Kmeans	0.7611	0.4448	0.5615	0.4571	0.4304	0.2282
T-nc dbSCAN	0.4231	0.3935	0.4078	0.3441	0.3773	0.4627
T-nc hdbSCAN	0.3385	0.3082	0.3227	0.2662	0.2941	0.3442
T-nc Kmeans	0.4402	0.3369	0.3817	0.3891	0.3313	0.5111
T-lc dbSCAN	0.3352	0.2267	0.2704	0.0513	0.1926	0.1077
T-lc hdbSCAN	0.3099	0.2081	0.2490	0.1101	0.1772	0.1092
T-lc Kmeans	0.6906	0.3715	0.4832	0.2795	0.3602	0.2115
ET-nc hdbSCAN	0.1663	0.1963	0.1801	-0.0134	0.1541	0.0379
ET-nc Kmeans	0.4006	0.3326	0.3634	0.2929	0.3223	0.1974
ET-lc hdbSCAN	0.3024	0.3582	0.3279	0.1848	0.2913	0.0325
ET-lc Kmeans	0.7387	0.3921	0.5122	0.3175	0.3782	0.1307

Tabella 4.5: Risultati sperimentazione del sito cs.princeton.edu

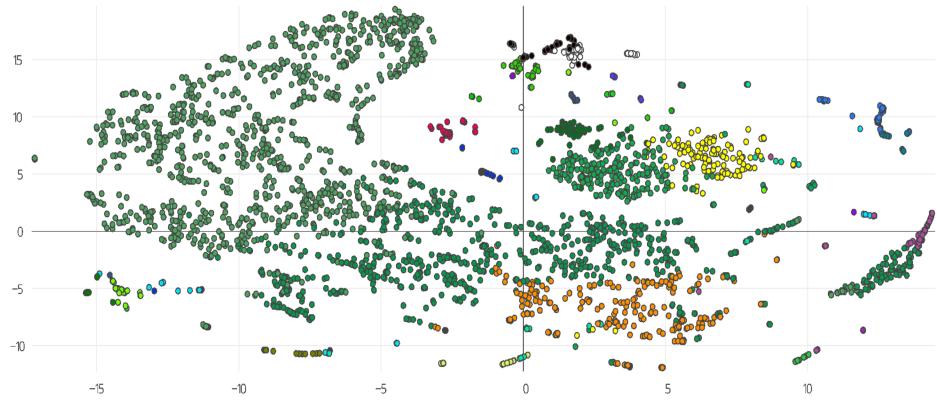


Figura 4.5: Rappresentazione del sito cs.princeton.edu, clusterizzato con Fastgreedy.

I cluster generati dalle rappresentazioni vettoriali delle pagine hanno formato cluster densi. L'algoritmo HDBSCAN è riuscito a distinguere meglio le pagine presentano la più alta V-measure e Adjusted Random Index. Anche qui le liste hanno migliorato notevolmente i risultati ottenuti.

cs.ox.ac.uk

Oxford	Hom	Com	V-M	ARI	MI	Silh
G-nc WT	0.2264	0.8330	0.3560	0.1133	0.2149	//
G-nc FG	0.4416	0.5005	0.4692	0.2617	0.4302	//
G-lc WT	0.1888	0.5069	0.2751	0.0034	0.1753	//
G-lc FG	0.4445	0.4444	0.4444	0.1758	0.4322	//
E-nc dbSCAN	0.2289	0.4673	0.3073	0.1777	0.2126	0.5214
E-nc hdbSCAN	0.2391	0.4616	0.3150	0.1811	0.2213	0.5241
E-nc Kmeans	0.3966	0.4649	0.4281	0.2516	0.3746	0.3991
E-lc dbSCAN	0.2401	0.5187	0.3282	0.2619	0.2319	0.5037
E-lc hdbSCAN	0.3007	0.5382	0.3859	0.2965	0.2880	0.4503
E-lc Kmeans	0.4247	0.4010	0.4125	0.2630	0.3788	0.2372
T-nc dbSCAN	0.2014	0.5083	0.2885	0.0658	0.1813	-0.1190
T-nc hdbSCAN	0.3421	0.4731	0.3971	0.1062	0.3260	0.0598
T-nc Kmeans	0.7021	0.4976	0.5825	0.2714	0.4821	0.1555
T-lc dbSCAN	0.4961	0.4837	0.4898	0.2616	0.4689	0.1319
T-lc hdbSCAN	0.4613	0.4744	0.4677	0.1219	0.4418	0.0526
T-lc Kmeans	0.7314	0.4946	0.5901	0.2623	0.4813	0.1600
ET-nc hdbSCAN	0.1343	0.1762	0.1524	-0.0357	0.1012	-0.0826
ET-nc Kmeans	0.3681	0.3052	0.3338	0.1337	0.2846	0.1322
ET-lc hdbSCAN	0.1537	0.1741	0.1632	-0.0383	0.1248	-0.0319
ET-lc Kmeans	0.3869	0.2836	0.3273	0.1277	0.2638	0.1386

Tabella 4.6: Risultati sperimentazione del sito cs.ox.ac.uk

QUESTA PARTE METTILA NELLA PARTE DI DISCUSSIONE DEI RISULTATI.

4.4.1 Community Detection

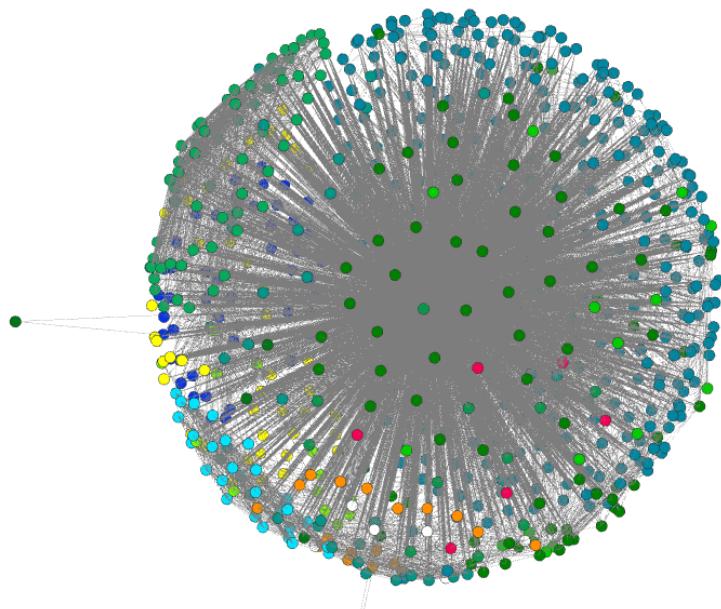


Figura 4.6: Grafo del sito `cs.illinois.edu` etichettato manualmente.

`cs.illinois.edu` L'algoritmo *Fastgreedy* non richiede parametri specifici ma è stato tagliato il dendrogramma ad altezza 15, il numero di cluster nella operazione di raggruppamento manuale. Ugualmente per *WalkTrap*, che però necessitava della lunghezza dei Random Walk da effettuare. I risultati migliori sono stati osservati con percorsi di lunghezza 3.

G - Illinois	Homog	Compl	V-Measure	ARI	MI
nc WalkTrap	0.6471	0.6585	0.6527	0.4363	0.6281
nc Fastgreedy	0.5518	0.8563	0.6711	0.5764	0.5354
lc WalkTrap	0.5093	0.4892	0.4991	0.2762	0.4722
lc Fastgreedy	0.5522	0.6035	0.5767	0.3656	0.5382

Tabella 4.7: Risultati sperimentazione di partizionamento del grafo del sito `cs.illinois.edu`

`cs.stanford.edu` Il grafo del sito presenta 1458 nodi e 99686 archi. Il dendrogramma restituito dall'algoritmo *Fastgreedy* è stato troncato ad altezza 10. Anche qui la scelta è ricaduta su Random Walk di lunghezza 3 per l'algoritmo *WalkTrap*.

G - Stanford	Homog	Compl	V-Measure	ARI	MI
nc WalkTrap	0.1628	0.3967	0.2308	0.1127	0.1340
nc Fastgreedy	0.3568	0.3894	0.3724	0.2439	0.3357
lc WalkTrap	0.3087	0.6314	0.4146	0.0502	0.1911
lc Fastgreedy	0.4854	0.6740	0.5644	0.2999	0.3892

Tabella 4.8: Risultati sperimentazione di partizionamento del grafo del sito `cs.stanford.edu`

`eeecs.mit.edu` Il grafo del sito presenta 1745 nodi e 63937 archi. L'algoritmo *Fastgreedy* ha costruito un dendrogramma che è stato tagliato ad altezza 15, il numero di cluster nella operazione di raggruppamento manuale. Ugualemente per *WalkTrap*, che però necessitava della lunghezza dei Random Walk da effettuare. I risultati migliori sono stati osservati con percorsi di lunghezza 3.

G - MIT	Homog	Compl	V-Measure	ARI	MI
nc WalkTrap	0.1792	0.6729	0.2830	0.1762	0.1648
nc Fastgreedy	0.5769	0.5215	0.5478	0.5514	0.5119
lc WalkTrap	0.0856	0.5013	0.1462	0.0450	0.0664
lc Fastgreedy	0.5617	0.7127	0.6282	0.6851	0.5497

Tabella 4.9: Risultati sperimentazione di partizionamento del grafo del sito `eecs.mit.edu`

`cs.princeton.edu` Il grafo del sito presenta 16378 nodi e 206985 archi. L'algoritmo *Fastgreedy* ha costruito un dendrogramma che è stato tagliato ad altezza 20, il numero di cluster nella operazione di raggruppamento manuale. Ugualmente per *WalkTrap*, che però necessitava della lunghezza dei Random Walk da effettuare. I risultati migliori sono stati osservati con percorsi di lunghezza 4.

G - Princeton	Homog	Compl	V-Measure	ARI	MI
nc WalkTrap	0.0259	0.1568	0.0444	-0.0221	0.0197
nc Fastgreedy	0.4900	0.5702	0.5270	0.4594	0.4848
lc WalkTrap	0.1543	0.2983	0.2034	0.1935	0.1375
lc Fastgreedy	0.4230	0.3817	0.4013	0.2168	0.3686

Tabella 4.10: Risultati sperimentazione di partizionamento del grafo del sito `cs.princeton.edu`

`cs.ox.ac.uk` Il grafo del sito presenta 4183 nodi e 27954 archi. L'algoritmo *Fastgreedy* ha costruito un dendrogramma che è stato tagliato ad altezza 20, il numero di cluster nella operazione di raggruppamento manuale. Ugualmente per *WalkTrap*, che però necessitava della lunghezza dei Random Walk da

effettuare. I risultati migliori sono stati osservati con percorsi di lunghezza 4.

G - Oxford	Homog	Compl	V-Measure	ARI	MI
nc WalkTrap	0.2264	0.8330	0.3560	0.1133	0.2149
nc Fastgreedy	0.4416	0.5005	0.4692	0.2617	0.4302
lc WalkTrap	0.1888	0.5069	0.2751	0.0034	0.1753
lc Fastgreedy	0.4445	0.4444	0.4444	0.1758	0.4322

Tabella 4.11: Risultati sperimentazione di partizionamento del grafo del sito `cs.ox.ac.uk`

4.4.2 URL Embedding

Considerando i Random Walk generati sul grafo come frasi, è possibile applicare algoritmi di Word Embedding per raggruppare le pagine sulla base del contesto in cui appaiono, ovvero le pagine che più verosimilmente appariranno insieme nelle sequenze.

Le sequenze di Random Walk sono state usate per apprendere rappresentazioni vettoriali delle pagine Web. La fase di URL embedding è stata effettuata utilizzando l'algoritmo Word2vec, [22] (esaminato in 1.3.2) modificando alcuni parametri e lasciando invariato altri.

I parametri personalizzati sono:

- **min-count:** tutte le parole (o URL) con frequenza di occorrenza minore di questo valore vengono ignorate.
- **window** rappresenta la distanza massima tra l'URL corrente e quello predetto all'interno di una frase.
- **negative** Nella fase di embedding di una URL, viene calcolato il rapporto tra la similarità del contesto con la parola e la sommatoria di

tutte le similarità tra la parola e gli altri contesti. Più precisamente:

$$\frac{v_c \cdot v_w}{\sum_{c \in C} v_c \cdot v_w} \quad (4.13)$$

Questa operazione può essere molto lenta. Per accelerare il processo possono venir scelti n contesti casuali da confrontare. Questo parametro, se maggiore di 0, rappresenta il numero di vettori da confrontare.

- **sg** definisce l'algoritmo di apprendimento, di default viene usato *CBOW*, mentre se impostato a 1 utilizza *skip-gram* [18]. Per lo scopo della sperimentazione si imposterà questo valore a CBOW.

I migliori risultati sono stati osservati con:

`cs.illinois.edu` è stata impostato un *min-count* pari a 1, *window* con valore 5 ed è stato utilizzato *skip-gram* con 5 *negative sampling*. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 4; HDBSCAN con *min-cluster-size*= 6; K-Means con numero di cluster pari a 15.

E - Illinois	Homog	Compl	V-Meas	ARI	MI	Silh
nc dbSCAN	0.5553	0.6579	0.6023	0.4487	0.5234	0.2588
nc hdbscan	0.5759	0.6720	0.6203	0.5282	0.5525	0.2573
nc Kmeans	0.8238	0.7575	0.7892	0.7883	0.7423	0.3131
lc dbSCAN	0.4163	0.5922	0.4889	0.2250	0.3935	0.1320
lc hdbscan	0.4760	0.5067	0.4908	0.2275	0.4515	0.1054
lc Kmeans	0.8095	0.6593	0.7267	0.6189	0.6473	0.2281

Tabella 4.12: Risultati sperimentazione di URL embedding del sito `cs.illinois.edu`

`cs.stanford.edu` è stata impostato un *min-count* pari a 1, *window* con valore 5 ed è stato utilizzato *skip-gram* con 5 *negative sampling*. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 7; HDBSCAN con *min-cluster-size*= 10; K-Means con numero di cluster pari a 15. Per il dataset delle liste: DBSCAN con $\epsilon = 1.6$ ed *min-samples*= 3; HDBSCAN con *min-cluster-size*= 5; K-Means con numero di cluster pari a 15.

E - Stanford	Homog	Compl	V-Meas	ARI	MI	Silh
nc dbSCAN	0.3453	0.4031	0.3720	0.2577	0.3252	0.4058
nc hdbSCAN	0.4720	0.4418	0.4564	0.3094	0.4210	0.4030
nc Kmeans	0.3810	0.3844	0.3827	0.2276	0.3574	0.5659
lc dbSCAN	0.4830	0.5798	0.5270	0.0976	0.3363	0.0938
lc hdbSCAN	0.6506	0.6299	0.6353	0.2466	0.5014	0.1762
lc Kmeans	0.5853	0.6838	0.6308	0.2423	0.4786	0.2513

Tabella 4.13: Risultati sperimentazione di URL embedding del sito `cs.stanford.edu`

`eecs.mit.edu` è stata impostato un *min-count* pari a 1, *window* con valore 5 ed è stato utilizzato *skip-gram* con 5 *negative sampling*. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 1.0$ ed *min-samples*= 10; HDBSCAN con *min-cluster-size*= 10; K-Means con numero di cluster pari a 10. Per il dataset delle liste: DBSCAN con $\epsilon = 1.5$ ed *min-samples*= 5; HDBSCAN con *min-cluster-size*= 13; K-Means con numero di cluster pari a 10.

E - MIT	Homog	Compl	V-Meas	ARI	MI	Silh
nc dbSCAN	0.4097	0.4236	0.4164	0.3779	0.3869	0.3264
nc HDBSCAN	0.5834	0.4413	0.5025	0.5811	0.4231	0.3141
nc Kmeans	0.6422	0.5129	0.5703	0.5141	0.5011	0.3914
lc dbSCAN	0.1727	0.5736	0.2655	0.1455	0.1550	0.1750
lc HDBSCAN	0.7882	0.4456	0.5693	0.5217	0.4061	0.2288
lc Kmeans	0.7165	0.5283	0.6081	0.3625	0.5146	0.2646

Tabella 4.14: Risultati sperimentazione di URL embedding del sito `eecs.mit.edu`

`cs.princeton.edu` è stata impostato un *min-count* pari a 1, *window* con valore 5 ed è stato utilizzato *skip-gram* con 5 *negative sampling*. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.8$ ed *min-samples= 7*; HDBSCAN con *min-cluster-size= 10*; K-Means con numero di cluster pari a 30. Per il dataset delle liste: DBSCAN con $\epsilon = 0.7$ ed *min-samples= 5*; HDBSCAN con *min-cluster-size= 8*; K-Means con numero di cluster pari a 25.

E - Princeton	Homog	Compl	V-Meas	ARI	MI	Silh
nc dbSCAN	0.0955	0.2417	0.1369	0.0622	0.0886	0.2656
nc HDBSCAN	0.2302	0.3292	0.2710	0.1651	0.2195	0.1687
nc Kmeans	0.4311	0.3637	0.3946	0.3404	0.3522	0.3877
lc dbSCAN	0.5941	0.5696	0.5816	0.7706	0.5579	0.2269
lc HDBSCAN	0.6531	0.5416	0.5922	0.7708	0.5275	0.1789
lc Kmeans	0.7611	0.4448	0.5615	0.4571	0.4304	0.2282

Tabella 4.15: Risultati sperimentazione di URL embedding del sito `cs.princeton.edu`

`cs.ox.ac.uk` è stata impostato un *min-count* pari a 1, *window* con valore 5 ed è stato utilizzato *skip-gram* con 5 *negative sampling*. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.8$ ed *min-samples= 7*; HDBSCAN con *min-cluster-size= 10*; K-Means con numero di cluster pari a 30. Per il dataset delle liste: DBSCAN con $\epsilon = 0.7$ ed *min-samples= 5*; HDBSCAN con *min-cluster-size= 8*; K-Means con numero di cluster pari a 25.

E - Oxford	Homog	Compl	V-Meas	ARI	MI	Silh
nc dbscan	0.2289	0.4673	0.3073	0.1777	0.2126	0.5214
nc hdbSCAN	0.2391	0.4616	0.3150	0.1811	0.2213	0.5241
nc Kmeans	0.3966	0.4649	0.4281	0.2516	0.3746	0.3991
lc dbscan	0.2401	0.5187	0.3282	0.2619	0.2319	0.5037
lc hdbSCAN	0.3007	0.5382	0.3859	0.2965	0.288052	0.4503
lc Kmeans	0.4247	0.4010	0.4125	0.2630	0.3788	0.2372

Tabella 4.16: Risultati sperimentazione di URL embedding del sito `cs.ox.ac.uk`

4.4.3 Text Mining

Sono state utilizzate tecniche di Text Mining per il clustering basato sul contenuto testuale. I contenuti all'interno di uno stesso sito web avranno una struttura e termini comuni, differenziandosi al variare dell'argomento trattato. La struttura gerarchica di un sito web organizza solitamente le pagine in sezioni simili. Questa metodologia tuttavia, considera solo l'informazione testuale, assumendo che i termini all'interno del sito web siano indipendenti l'uno dall'altro così come i documenti, ignorando le relazioni interdipendenti tra questi. Il web si discosta dall'analisi classica dei documenti proprio per le relazioni che intercorrono tra le pagine, tuttavia l'analisi testuale rimane molto importante.

Nella fase di sperimentazione è stata utilizzata una rappresentazione vettoriale della frequenza dei termini all'interno dell'insieme delle pagine web, calcolata con la tecnica della *frequency-inverse document frequency* (tf-idf).

I parametri personalizzati per la costruzione della matrice documenti-termini con funzione di peso *idf* sono stati:

- **max-df**: questo valore rappresenta la massima frequenza, all'interno dei documenti, che un termine può avere per essere utilizzato nella matrice tf-idf. Se un termine appare molte volte nel corpus, molto probabilmente avrà poco significato.
- **min-df**: indica il numero minimo di documenti in cui un termine dovrà apparire per essere considerato.
- **ngram-range**: vengono presi in considerazioni gli n-grammi di lunghezza compresa nell'intervallo specificato in questo parametro. Un n-gramma è una sottosequenza di n elementi di un'altra.

I risultati mostrati sono stati ottenuti nel seguente modo:

`cs.illinois.edu` Sul dataset del sito, costituito da 728 pagine e 433 termini, il corpus è stato ripulito delle stopword, stemmatizzato ed è stato impostato il $\max\text{-}df$ all'80%, il $\min\text{-}df$ a 0.1 e sono stati considerati solo uni-grammi, bi-grammi e tri-grammi. Se i termini appaiono in più dell'80% dei documenti, probabilmente avrà poco significato, lo stesso se appare troppe poche volte. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.9$ ed $\min\text{-}samples = 4$; HDBSCAN con $\min\text{-}cluster\text{-}size = 4$; K-Means con numero di cluster pari a 15.

Sul Dataset costruito estraendo le liste: DBSCAN con $\epsilon = 0.7$ ed $\min\text{-}samples = 4$; HDBSCAN con $\min\text{-}cluster\text{-}size = 7$; K-Means con numero di cluster pari a 15.

T - Illinois	Homog	Compl	V-Meas	ARI	MI	Silh
nc dbscan	0.5601	0.5962	0.5776	0.4078	0.5346	0.1242
nc hdbSCAN	0.5152	0.6029	0.5556	0.3862	0.4858	0.0881
nc Kmeans	0.7619	0.5814	0.6596	0.3184	0.5586	0.1767
lc dbscan	0.5710	0.7042	0.6306	0.5197	0.5566	0.1406
lc hdbSCAN	0.4501	0.5104	0.4783	0.1938	0.4297	0.1018
lc Kmeans	0.8061	0.5892	0.6808	0.4296	0.5748	0.2002

Tabella 4.17: Risultati sperimentazione di Text Mining del sito `cs.illinois.edu`

`cs.stanford.edu` Sul dataset del sito, costituito da 1458 pagine e 843 termini, il corpus è stato ripulito delle stopword, stemmatizzato ed è stato impostato il *max-df* all'80%, il *min-df* a 0.1 e sono stati considerati solo unigrammi, bi-grammi e tri-grammi. Se i termini appaiono in più dell'80% dei documenti, probabilmente avrà poco significato, lo stesso se appare troppe poche volte. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.3$ ed *min-samples*= 5; HDBSCAN con *min-cluster-size*= 15; K-Means con numero di cluster pari a 15.

Sul Dataset costruito estraendo le liste: DBSCAN con $\epsilon = 0.5$ ed *min-samples*= 3; HDBSCAN con *min-cluster-size*= 5; K-Means con numero di cluster pari a 15.

T - Stanford	Homog	Compl	V-Meas	ARI	MI	Silh
nc dbscan	0.2584	0.3435	0.2949	0.0208	0.2403	0.0981
nc hdbSCAN	0.3015	0.2784	0.2895	0.0089	0.2521	0.0916
nc Kmeans	0.6014	0.4652	0.5246	0.2639	0.4489	0.2704
lc dbscan	0.0876	0.3924	0.1432	0.0102	0.0327	-0.1188
lc hdbSCAN	0.1827	0.3802	0.2468	0.0687	0.1326	0.0841
lc Kmeans	0.5436	0.5726	0.5577	0.2167	0.4157	0.1732

Tabella 4.18: Risultati sperimentazione di Text Mining del sito `cs.stanford.edu`

eeecs.mit.edu Sul dataset del sito, costituito da 1745 pagine e 354 termini, il corpus è stato ripulito delle stopword, stemmatizzato ed è stato impostato il *max-df* all'80%, il *min-df* a 0.1 e sono stati considerati solo uni-grammi, bi-grammi e tri-grammi. Se i termini appaiono in più dell'80% dei documenti, probabilmente avrà poco significato, lo stesso se appare troppe poche volte. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 9; HDBSCAN con *min-cluster-size*= 15; K-Means con numero di cluster pari a 10.

Sul Dataset costruito estraendo le liste: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 6; HDBSCAN con *min-cluster-size*= 9; K-Means con numero di cluster pari a 10.

T - MIT	Homog	Compl	V-Meas	ARI	MI	Silh
nc dbscan	0.2071	0.1908	0.1986	0.1011	0.1759	0.1133
nc hdbSCAN	0.2363	0.2679	0.2611	0.0918	0.2207	0.0845
nc Kmeans	0.4175	0.2381	0.3032	0.0917	0.2290	0.1958
lc dbscan	0.3470	0.3466	0.3467	0.2071	0.3326	0.1180
lc hdbSCAN	0.4402	0.4147	0.4271	0.2806	0.3995	0.1423
lc Kmeans	0.5857	0.4198	0.4890	0.2597	0.4085	0.2296

Tabella 4.19: Risultati sperimentazione di Text Mining del sito `eeecs.mit.edu`

`cs.princeton.edu` Sul dataset del sito, costituito da 16378 pagine e 470 termini, il corpus è stato ripulito delle stopword, stemmatizzato ed è stato impostato il *max-df* all'80%, il *min-df* a 0.1 e sono stati considerati solo unigrammi, bi-grammi e tri-grammi. Se i termini appaiono in più dell'80% dei documenti, probabilmente avrà poco significato, lo stesso se appare troppe poche volte. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.3$ ed *min-samples*= 5; HDBSCAN con *min-cluster-size*= 15; K-Means con numero di cluster pari a 10.

Sul Dataset costruito estraendo le liste: DBSCAN con $\epsilon = 0.9$ ed *min-samples*= 6; HDBSCAN con *min-cluster-size*= 15; K-Means con numero di cluster pari a 25.

T - Princeton	Homog	Compl	V-Meas	ARI	MI	Silh
nc dbSCAN	0.4231	0.3935	0.4078	0.3441	0.3773	0.4627
nc HDBSCAN	0.3385	0.3082	0.3227	0.2662	0.2941	0.3442
nc Kmeans	0.4402	0.3369	0.3817	0.3891	0.3313	0.5111
lc dbSCAN	0.3352	0.2267	0.2704	0.0513	0.1926	0.1077
lc HDBSCAN	0.3099	0.2081	0.2490	0.1101	0.1772	0.1092
lc Kmeans	0.6906	0.3715	0.4832	0.2795	0.3602	0.2115

Tabella 4.20: Risultati sperimentazione di Text Mining del sito cs.princeton.edu

cs.ox.ac.uk Sul dataset del sito, costituito da 3951 pagine e 363 termini, il corpus è stato ripulito delle stopword, stemmatizzato ed è stato impostato il *max-df* all'80%, il *min-df* a 0.1 e sono stati considerati solo uni-grammi, bi-grammi e tri-grammi. Se i termini appaiono in più dell'80% dei documenti, probabilmente avrà poco significato, lo stesso se appare troppe poche volte. Gli algoritmi testati sono stati: DBSCAN con $\epsilon = 0.3$ ed *min-samples*= 5; HDBSCAN con *min-cluster-size*= 15; K-Means con numero di cluster pari a 10.

Sul Dataset costruito estraendo le liste: DBSCAN con $\epsilon = 0.7$ ed *min-samples*= 7; HDBSCAN con *min-cluster-size*= 7; K-Means con numero di cluster pari a 25.

T - Oxford	Homog	Compl	V-Meas	ARI	MI	Silh
nc dbSCAN	0.2014	0.5083	0.2885	0.0658	0.1813	-0.1190
nc HDBSCAN	0.3421	0.4731	0.3971	0.1062	0.3260	0.0598
nc Kmeans	0.7021	0.4976	0.5825	0.2714	0.4821	0.1555
lc dbSCAN	0.4961	0.4837	0.4898	0.2616	0.4689	0.1319
lc HDBSCAN	0.4613	0.4744	0.4677	0.1219	0.4418	0.0526
lc Kmeans	0.7314	0.4946	0.5901	0.2623	0.4813	0.1600

Tabella 4.21: Risultati sperimentazione di Text Mining del sito cs.ox.ac.uk

4.4.4 Embedding e Text Mining

Sono stati quindi considerati come un unico vettore. Il vantaggio di associare entrambe le relazioni in uno spazio vettoriale offre il vantaggio usare la stessa rappresentazione e quindi di unire i vettori derivanti dagli algoritmi di Word Embedding con quelli derivanti dall'analisi di contenuto testuale.

Così facendo è possibile dare più importanza ad una tipologia di informazione piuttosto che ad un'altra, andando a modificare il rapporto tra le dimensioni dei vettori.

cs.illinois.edu I vettori di word2vec sono stati generati di dimensione 48, mentre i vettori-riga documenti sono stati ridotti con *TruncateSVD* a dimensione 50. Gli algoritmi testati sono stati: HDBSCAN con *min-cluster-size= 7*; K-Means con numero di cluster pari a 15.

ET - Illinois	Homog	Compl	V-Meas	ARI	MI	Silh
nc hdbSCAN	0.7327	0.7534	0.7429	0.7204	0.7186	0.2070
nc Kmeans	0.8812	0.8069	0.8424	0.8299	0.7949	0.3198
lc hdbSCAN	0.6541	0.6129	0.6328	0.3249	0.5992	0.1203
lc Kmeans	0.8548	0.6885	0.7627	0.6488	0.6773	0.2573

 Tabella 4.22: Risultati sperimentazione di Text-Embedding del sito cs.illinois.edu

cs.stanford.edu I vettori di word2vec sono stati generati di dimensione 48, mentre i vettori-riga documenti sono stati ridotti con *TruncateSVD* a dimensione 50. Gli algoritmi testati sono stati: HDBSCAN con *min-cluster-size*= 10; K-Means con numero di cluster pari a 15.

ET - Stanford	Homog	Compl	V-Meas	ARI	MI	Silh
nc hdbSCAN	0.3495	0.5272	0.4203	0.3335	0.3307	0.3469
nc Kmeans	0.3490	0.3629	0.3558	0.2061	0.3238	0.4336
lc hdbSCAN	0.2343	0.7045	0.3517	0.1012	0.2033	0.2342
lc Kmeans	0.7508	0.7422	0.7465	0.5123	0.6656	0.2375

 Tabella 4.23: Risultati sperimentazione di Text-Embedding del sito cs.stanford.edu

eeecs.mit.edu I vettori di word2vec sono stati generati di dimensione 48, mentre i vettori-riga documenti sono stati ridotti con *TruncateSVD* a dimensione 50. Gli algoritmi testati sono stati: HDBSCAN con *min-cluster-size*= 14; K-Means con numero di cluster pari a 10.

ET - MIT	Homog	Compl	V-Meas	ARI	MI	Silh
nc hdbSCAN	0.5573	0.3748	0.4482	0.2845	0.3600	0.1607
nc Kmeans	0.5505	0.4344	0.4856	0.2687	0.4252	0.1699
lc hdbSCAN	0.6076	0.4329	0.5056	0.1910	0.4181	0.1464
lc Kmeans	0.6679	0.5332	0.5930	0.3499	0.5235	0.2311

Tabella 4.24: Risultati sperimentazione di Text-Embedding del sito eecs.mit.edu

cs.princeton.edu I vettori di word2vec sono stati generati di dimensione 48, mentre i vettori-riga documenti sono stati ridotti con *TruncateSVD* a dimensione 50. Gli algoritmi testati sono stati: HDBSCAN con *min-cluster-size*= 14; K-Means con numero di cluster pari a 10.

ET - Princeton	Homog	Compl	V-Meas	ARI	MI	Silh
nc hdbSCAN	0.1663	0.1963	0.1801	-0.0134	0.1541	0.0379
nc Kmeans	0.4006	0.3326	0.3634	0.2929	0.3223	0.1974
lc hdbSCAN	0.3024	0.3582	0.3279	0.1848	0.2913	0.0325
lc Kmeans	0.7387	0.3921	0.5122	0.3175	0.3782	0.1307

Tabella 4.25: Risultati sperimentazione di Text-Embedding del sito cs.princeton.edu

cs.ox.ac.uk I vettori di word2vec sono stati generati di dimensione 48, mentre i vettori-riga documenti sono stati ridotti con *TruncateSVD* a dimensione 50. Gli algoritmi testati sono stati: HDBSCAN con *min-cluster-size*= 14; K-Means con numero di cluster pari a 10.

ET - Oxford	Homog	Compl	V-Meas	ARI	MI	Silh
nc hdbSCAN	0.1343	0.1762	0.1524	-0.0357	0.1012	-0.0826
nc Kmeans	0.3681	0.3052	0.3338	0.1337	0.2846	0.1322
lc hdbSCAN	0.1537	0.1741	0.1632	-0.0383	0.1248	-0.0319
lc Kmeans	0.3869	0.2836	0.3273	0.1277	0.2638	0.1386

Tabella 4.26: Risultati sperimentazione di Text-Embedding del sito cs.ox.ac.uk

4.4.5 Analisi dei risultati

Il processo di crawling è stato svolto su 3 siti, sui quali sono state generate sequenze di 2 tipi diversi, ossia Random Walk senza vincoli e Random Walk con i vincoli delle liste. Su di esse sono state poi effettuate 4 metodologie diverse: grafo, Word Embedding, Text Mining e Word Embedding & Text Mining. Per ogni metodologia, inoltre, sono stati usati 3 algoritmi di apprendimento diversi, ovvero K-Means, HDBSCAN e DBSCAN. Infine, su ogni algoritmo sono state ricavate 6 metriche. In tutto sono 432 valori.

Di seguito sono riportati i risultati più interessanti per una migliore comprensione.

K-Means nc - cs.illinois.edu

Type	Homog	Compl	V-Meas	ARI	MI	Silh
Embedding	0.5553	0.6579	0.6023	0.4487	0.5234	0.2588
Text Mining	0.5759	0.6720	0.6203	0.5282	0.5525	0.2573
Emb + Text	0.8238	0.7575	0.7892	0.7883	0.7423	0.3131

Tabella 4.27: Risultati sperimentazione raggruppati per algoritmo K-Means sul sito cs.illinois.edu

In questo caso la combinazione delle due informazioni ha portato un miglioramento complessivo del clustering. Nel terzo caso si è verificato un miglioramento in media di 22.0% rispetto al primo, e di 18.8% rispetto al secondo.

K-Means lc - cs.stanford.edu

Type	Homog	Compl	V-Meas	ARI	MI	Silh
Embedding	0.5853	0.6838	0.6308	0.2423	0.4786	0.2513
Text Mining	0.5857	0.4198	0.4890	0.2597	0.4085	0.2296
Emb + Text	0.7508	0.7422	0.7465	0.5123	0.6656	0.2375

Tabella 4.28: Risultati sperimentazione raggruppati per algoritmo K-Means sul sito ecs.mit.edu

Anche in questo caso l'utilizzo combinato di Text Mining e Word Embedding ha portato miglioramento. In media di 16% rispetto all'Embedding e di 21.2% rispetto al Text Mining.

K-Means lc - eecs.mit.edu

Type	Homog	Compl	V-Meas	ARI	MI	Silh
Embedding	0.7165	0.5283	0.6081	0.3625	0.5146	0.2646
Text Mining	0.5436	0.5726	0.5577	0.2167	0.4157	0.1732
Emb + Text	0.6679	0.5332	0.5930	0.3499	0.5235	0.2311

Tabella 4.29: Risultati sperimentazione raggruppati per algoritmo K-Means sul sito ecs.mit.edu

Qui, invece, è risultata migliore l'applicazione dell'embedding, con un miglioramento di 1.2% rispetto all'Word Embedding & Text Mining, mentre di 8.4% rispetto al solo Text Mining.

Lo scopo della tesi non era valutare l'efficacia dei vari algoritmi riportati, ma verificare un eventuale miglioramento nei risultati ottenuti.

Valutare i risultati di un algoritmo di clustering non è un'operazione semplice. Infatti l'assegnazione manuale delle etichette denota una certa arbitrarietà. Inoltre, l'analisi dei percorsi ha fatto notare come certe classi, idealmente raggruppate insieme in quanto stessa entità (e.g. docenti), possano invece essere divise in fase di apprendimento per motivi ragionevoli. Ad esempio, nel sito *cs.illinois.edu*, erano presenti molteplici pagine relative agli stessi professori. Questo era dovuto al fatto che durante gli anni erano state pubblicate diverse edizioni del sito. Questo ha portato le diverse versioni della stessa pagina (concettuale) ad essere presenti con diversa frequenza in percorsi diversi. O ancora ad avere anche testo differente. Infatti anche l'analisi testuale fra le diverse versioni era differente.

Un altro esempio era il raggruppamento di pagine con poco testo oppure, diverse pagine relative agli studenti "undergraduates" etichettate in classi diverse, sono state raggruppate nello stesso cluster, probabilmente dovuto al fatto che apparivano in percorsi simili ed avevano testi simili.

In conclusione il problema del clustering di pagine web può rivelarsi ostico e dare risultati diversi da quelli desiderati ma comunque sensati. Considerare più aspetti può essere rivelarsi utile in molti contesti applicativi.

Capitolo 5

Conclusioni e sviluppi futuri

In questo lavoro di tesi è stato trattato il clustering di pagine Web, proponendo una nuova metodologia. La trattazione effettuata è incentrata sull'utilizzo dei Random Walk per apprendere rappresentazioni vettoriali delle pagine, unitamente al loro contenuto testuale. Il lavoro non pretende di essere esaustivo, ma piuttosto un punto di partenza per ulteriori sviluppi, sia teorici che sperimentali.

I risultati sperimentali prodotti si sono rivelati discreti e incentivano a proseguire gli studi in questa direzione in modo da individuare nuove tecniche che permettano di migliorare i risultati raggiunti in termini di qualità. In particolare è stato osservato come la forma dei cluster vari in funzione dal Dataset. Quindi, sarebbe opportuno utilizzare l'algoritmo più appropriato in base al contesto.

Da notare come le "analogie" estraibili dall'utilizzo di *Word2vec* su collezioni di documenti, abbiano avuto un riscontro nel Web.

Bibliografía

- [1] David Aldous and James Allen Fill. Reversible markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at <http://www.stat.berkeley.edu/\sim\aldous/RWG/book.html>.
- [2] Raul Baños, Consolación Gil, Julio Ortega, and Francisco G. Montoya. Multilevel heuristic algorithm for graph partitioning. In *Proceedings of the 2003 International Conference on Applications of Evolutionary Computing*, EvoWorkshops'03, pages 143–153, Berlin, Heidelberg, 2003. Springer-Verlag.
- [3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.
- [4] Bettina Berendt, Bamshad Mobasher, Myra Spiliopoulou, and John Wiltschire. Measuring the accuracy of sessionizers for web usage analysis. In *Int. SIAM Workshop on Web Mining*, Apr. 2001.
- [5] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. Knowl. Discov. Data*, 10(1):5:1–5:51, July 2015.
- [6] I-Jen Chiang, Charles Chih-Ho Liu, Yi-Hsin Tsai, and Ajit Kumar. Discovering latent semantics in web documents using fuzzy clustering. *Fuzzy Systems, IEEE Transactions on*, 23(6):2122–2134, 2015.

- [7] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.
- [8] Robert Cooley. The use of web structure and content to identify subjectively interesting web usage patterns. *ACM Trans. Internet Technol.*, 3(2):93–116, May 2003.
- [9] Daniel Crabtree, Peter Andreea, and Xiaoying Gao. Query directed web page clustering. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, WI ’06, pages 202–210, Washington, DC, USA, 2006. IEEE Computer Society.
- [10] Valter Crescenzi, Paolo Merialdo, and Paolo Missier. Clustering web pages based on their structure. *Data Knowl. Eng.*, 54(3):279–299, September 2005.
- [11] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.
- [12] Yeye He, Dong Xin, Venkatesh Ganti, Sriram Rajaraman, and Nirav Shah. Crawling deep web entity pages. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM ’13, pages 355–364, New York, NY, USA, 2013. ACM.
- [13] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.
- [14] Shian-Hua Lin, Kuan-Pak Chu, and Chun-Ming Chiu. Automatic sitemaps generation: Exploring website structures using block extraction and hyperlink analysis. *Expert Syst. Appl.*, 38(4):3944–3958, April 2011.
- [15] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, December 2007.
- [16] MacQueen. Some methods for classification and analysis of multivariate observation. In *Proceedings of the Fifth Berkeley Symposium on*

- Mathematical Statistics and Probability.* University of California Press, 1967.
- [17] Joe Marini. *Document Object Model.* McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2002.
 - [18] Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics, May 2013.
 - [19] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.
 - [20] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.
 - [21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’14, pages 701–710, New York, NY, USA, 2014. ACM.
 - [22] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. <http://is.muni.cz/publication/884893/en>.
 - [23] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.
 - [24] Peter Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, November 1987.

- [25] Jorge M. Santos and Mark Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *Proceedings of the 19th International Conference on Artificial Neural Networks: Part II*, ICANN '09, pages 175–184, Berlin, Heidelberg, 2009. Springer-Verlag.
- [26] C. Shahabi, A. M. Zarkesh, J. Adibi, and V. Shah. Knowledge discovery from users web-page navigation. In *Proceedings of the 7th International Workshop on Research Issues in Data Engineering (RIDE '97) High Performance Database Management for Large-Scale Applications*, RIDE '97, pages 20–, Washington, DC, USA, 1997. IEEE Computer Society.
- [27] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Random walk with restart: Fast solutions and applications. *Knowl. Inf. Syst.*, 14(3):327–346, March 2008.
- [28] R.C. Tryon and D.E. Bailey. *Cluster Analysis;corelation Profile an Orthometric (factor) Analysis for the Isolation of Unities in Mind and Personality, by Robert Choate Tryon ... Ann Arbor, Mich., Edwards Brothers, Inc., Lithoprinters and Publishers, 1939.* McGraw-Hill, 1970.
- [29] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [30] Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, January 2010.
- [31] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
- [32] Tim Weninger, Yonatan Bisk, and Jiawei Han. Document-topic hierarchies from document graphs. In *Proceedings of the 21st ACM Interna-*

- tional Conference on Information and Knowledge Management*, CIKM '12, pages 635–644, New York, NY, USA, 2012. ACM.
- [33] Will Y. Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398. ACL, 2013.