Corso di laurea magistrale in Scienze Statistiche
Metodi informatici per statistica e data science
2023/2024

Università degli Studi di Padova

Mini-progetto

LSH near duplicate document detection

Pietro Stangherlin 2129207
Nicola Castelletti 2109302

2023/2024

Indice della presentazione

- Obiettivi
- Metodi utilizzati
- Esperimenti
- Conclusioni

Background

Problema: in grandi raccolte di documenti (in particolare testuali) è possibile che vi siano dei documenti duplicati, quasi uguali, oppure che un documento sia già contenuto in un altro; è inoltre possibile che tali documenti abbiano identificativi diversi o comunque non direttamente riconducibili tra loro.

Degli esempi possono essere:
- medesimi documenti scannerizzati più volte da entità diverse.
- un'organizzazione che mantiene versioni diverse, ovvero con lievi modifiche, di uno stesso documento.

Obbiettivi

Sviluppare un metodo basato su MinHash e LSH per identificare i documenti quasi-duplicati.

Valutare l’efficacia e analizzare l’efficienza
-> Specifica misure di efficacia

Metodi utilizzati

- Creazione della collezione sperimentale
- Shingling
- MinHash
- LSH

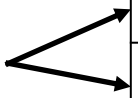
Creazione collezione sperimentale (1)

1) Si assume di avere una collezione iniziale senza duplicati e quasi – duplicati. Ogni documento presenta un codice identificativo (id).
2) Si seleziona (ad esempio tramite un generatore di numeri casuali) un sottoinsieme di documenti da cui generare i quasi duplicati.
3) Scelta del tipo di modifica dei documenti per la creazione dei semi duplicati: sostituzione di alcuni caratteri con altri (random), per simulare un risultato di un programma di riconoscimento ottico di testo. Parametro relativo alla frequenza di sostituzione (bassa o elevata)

Creazione collezione sperimentale (2)

4) A ogni quasi–duplicato si assegna un id univoco (id1) e l'id del documento da cui è stato generato (id2). Viene creata una nuova collezione con i quasi-duplicati e i documenti originali (per questi ultimi il campo l'id2 è posto uguale a None)

Id	text
1	hello world
2	Bob, Alice and Ulm
3	nosce te ipsum



id1	id2	text
1	None	hello world
2	None	Bob, Alice and Ulm
3	None	nosce te ipsum
4	3	n@sCe t3 ipsuN

Creazione collezione sperimentale (3)

4)Durante la creazione della nuova collezione si crea una struttura dati (ad esempio dizionario) o file in cui per ogni id di un documento originale sono associati gli id dei documenti artificiali (o viceversa), in questo modo la procedura di valutazione è più efficiente.

Shingling (1)

- Dimensione delle shingles: w = 9 [5]
- Trattamento caratteri speciali (spazio, newline etc)
- Salvataggio delle shingles, proposte di [1]:
  - Salvare anche la frequenza delle shingles nel documento (o usare una shingle più volte) -> possibilmente confrontare
  - Salvare solo le shingles uniche (maggiore efficienza)
- Hash per le shingles: compromesso tra memoria (definizione del numero di bit dell'hash) impiegata e numero di collisioni per shingles diverse. [1] consiglia di impiegare la Rabin fingerprint.

Shingling (2)

- Struttura dati per salvare le shingles (i loro hash): l'insieme delle shingles è temporaneo, serve solo per calcolare la signature, poi è eliminato
  - Lista ordinata:
    - Pro: occupa poca memoria O(n)
    - Contro: per ordinarla la complessità è O(n * log(n))
    - Neutrale: ricerca di un elemento in O(log(n)) (binary search)
  - Hash list
    - Pro: ricerca in O(1)
    - Contro: (se in media vogliamo la ricerca in O(1)) deve occupare molta memoria (celle vuote in circa il 70 % delle posizioni)

MinHash (1)

- Valutare se le signatures di tutti i documenti possono risiedere in memoria centrale, altrimenti
  - Scrittura su disco
  - Modificare parametri delle shingles e delle permutazioni
- Scelta del numero k di permutazioni e relative funzioni hash
- Nella costruzione del MinHash: valutare se tenere in memoria un dizionario con valori di vettori di k vettori (tanti quante sono le funzioni hash di permutazione) oppure mantenere in memoria le shingles e fare un dizionario alla volta

MinHash (2)

- Scelta della struttura dati per le signatures: ciascun documento deve essere identificabile tramite id (per i confronti tra bande di signatures dopo LSH) e le signatures devono essere delle liste ordinate.

LSH

- Numero di bande
- Funzioni hash per la bande
- Struttura dati per i bucket: hash-table dove ad ogni chiave è associata una lista con gli id dei documenti
- Diverse implementazioni:
  - in memoria centrale
    - Hash-table
    - Hash-tree o b-tree
  - In memoria di massa

Esperimenti

- Prima esecuzione della procedura sul dataset originale (senza semi-duplicati artificiali) per tarare i parametri
- Esecuzioni successive con diversi livelli di modifica dei documenti semi-duplicati artificiali

Conclusioni

Bibliografia

1. A.Z. Broder, “On the resemblance and containment of documents,” Proc. Compression and Complexity of Sequences, pp. 21–29, Positano Italy, 1997
2. A.Z. Broder, M. Charikar, A.M. Frieze, and M. Mitzenmacher, “Min-wise independent permutations,” ACM Symposium on Theory of Computing, pp. 327–336, 1998.
3. P. Indyk and R. Motwani. “Approximate nearest neighbor: towards re moving the curse of dimensionality,” ACM Symposium on Theory of Com puting, pp. 604–613, 1998.
4. A. Gionis, P. Indyk, and R. Motwani, “Similarity search in high dimensions via hashing,” Proc. Intl. Conf. on Very Large Databases, pp. 518 529, 1999
5. J. Leskovec, A. Rajaraman, and J. Ullman. “Mining of Massive Dataset“ Cambridge University Press, Second edition, (2014)