Graboid RFC

Progetto Gestione dell'Informazione
2025-03-09

Menozzi Matteo, Turci Gabriele e Turci Sologni Enrico

Indice

1. Dataset

1.1. Scelta del Dataset

Abbiamo dedicato particolare attenzione alla selezione di un dataset adeguato. La scelta è ricaduta sull'insieme di documenti denominati **Request For Comments** (**RFC**), una serie di pubblicazioni tecniche che definiscono standard, metodologie e sviluppi tecnici di Internet.

La selezione di questo dataset si è basata su una serie di requisiti fondamentali e vantaggi distintivi che gli RFC soddisfano pienamente. Di seguito sono riportati i motivi principali e i benefici derivanti dall'uso degli RFC come dataset per il nostro progetto.

1. **Facilità di Recupero**: Gli RFC sono facilmente accessibili grazie alla struttura del loro sistema di identificazione. Ogni documento è identificato da un numero univoco incrementale, il che rende lo **scraping particolarmente semplice**. Per esempio, è possibile recuperare i documenti incrementando sistematicamente un indice numerico. Inoltre, esistono repository ben organizzati che facilitano il download di massa.

2. **Dimensione del Dataset**: Il corpus degli RFC comprende circa 9600 documenti, un numero perfettamente adeguato per il nostro scopo. Questo range consente di analizzare e testare algoritmi di indicizzazione e ricerca su un dataset realistico senza incorrere in problemi di scalabilità o carenza di dati.

3. **Varietà nella Lunghezza e Complessità dei Documenti**: Gli RFC presentano una significativa varianza nella lunghezza dei documenti:

   - Alcuni documenti contano poche decine di righe.
   - Altri arrivano a diverse centinaia di righe.

   Questa varietà consente di testare il motore di ricerca su casi molto diversi in termini di granularità del contenuto.

4. **Ricchezza di Termini Specifici**: Gli RFC sono documenti tecnici caratterizzati dall'uso di terminologia altamente specifica, inclusi acronimi e parole chiave che rappresentano concetti chiave nel dominio informatico. Questa caratteristica li rende ideali per:

   - Creare indici dettagliati.
   - Formulare query tecniche complesse.
   - Effettuare benchmarking di algoritmi di ricerca avanzati.

5. **Presenza di Metadati Strutturati**: Ogni RFC include metadati ben definiti, quali: *Numero identificativo*, *Titolo*, *Autori*, *Data di pubblicazione*, *Status (es. Standard, Informativo, Obsoleto)*, *Abstract e Parole chiave*, *Sezioni strutturate (Introduzione, Specifiche, Conclusione, ecc.)*.

   La presenza di questi metadati consente di implementare funzionalità avanzate, come la ricerca basata su campi specifici o il filtraggio per criteri.

6. **Compatibilità con Query Complesse**: Gli RFC coprono una vasta gamma di argomenti tecnici e standard, il che li rende perfetti per simulare query complesse e specifiche. Ad esempio, è possibile eseguire ricerche che riguardano concetti tecnici, interazioni tra standard o termini legati a specifiche versioni.

1.2. Scraping del Dataset

Abbiamo deciso di costruire un dataset personalizzato effettuando lo scraping manuale dei documenti e dei relativi metadati. Non abbiamo utilizzato dataset precostruti, poiché volevamo garantire che i dati soddisfacessero specificamente i requisiti del nostro motore di ricerca.

1.2.1. Scraping dei Metadati

La prima fase del processo prevede il recupero dei metadati associati a ciascun documento RFC.

- Il codice HTML degli RFC è carente e i metadati inclusi nei documenti stessi risultano spesso inconsistenti e difficili da estrarre in modo affidabile.
- Per ovviare a questa limitazione, abbiamo individuato una fonte alternativa: i metadati sono disponibili in formato tabellare sul sito ufficiale dell'RFC Editor, raggiungibile al seguente link: Fonte dei Metadati.
- I metadati recuperati includono campi come *numero identificativo*, *titolo*, *autori*, *data di pubblicazione*, *stato*, *estratto*, e *parole chiave*. Questi sono stati scaricati e organizzati in modo strutturato per essere utilizzati successivamente.

1.2.2. Scraping dei Corpi dei Documenti

Una volta ottenuti i metadati, il passo successivo è stato lo scraping dei corpi dei documenti.

- Gli RFC sono accessibili tramite URL che seguono una struttura numerica incrementale, come: `https://www.rfc-editor.org/rfc/rfcX.txt`, dove `X` è l'identificatore numerico del documento (es. 1, 2, 3, ...).
- Uno script iterativo automatizza il download di ogni documento a partire dal numero 1 fino all'ultimo RFC disponibile. Per velocizzare il processo, viene impiegato un sistema basato su threadpool, che consente il download parallelo di più documenti, migliorando le prestazioni.

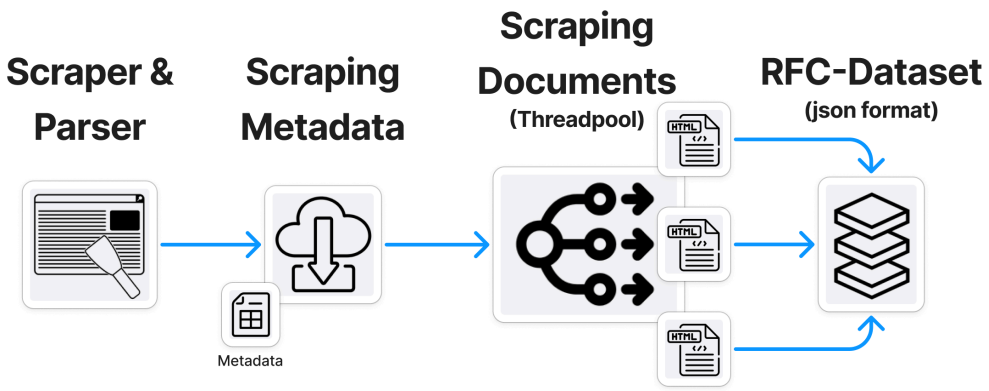

Figura 1: Il diagramma illustra il flusso completo del processo di scraping e costruzione del dataset.

1.2.3. Formattazione del Dataset

Una volta completato il recupero dei metadati e dei corpi dei documenti. Tutte le informazioni sono state consolidate in un unico file in formato **JSON**, dove ogni documento rappresenta un record strutturato contenente sia i metadati sia il testo completo del documento.

Esempio:


```
[
    {
        "Number": "9009",
        "Date": "2021-04",
        "Status": "Proposed Standard",
        "More Info": "Does not obsolete any RFC.",
        "Title": "Efficient Route Invalidation",
        "Authors": [
            "R.A. Jadhav", "P. Thubert", "R.N. Sahoo", "Z. Cao"
        ],
        "Files": [
            "HTML", "TEXT", "PDF", "XML"
        ],
        "Keywords": [
            "NPDAO", "DCO", "no-path", "route", "cleanup"
        ],
        "Abstract": "This document explains ... optimized route invalidation
messaging.",
        "Content": "A directed graph having the property ... timer value of 1
second."
    },
    ...
]
```


1.3. Locazione dei File Interessati

Script di Parsing:

Lo script responsabile per lo scraping dei metadati e dei corpi dei documenti si trova al seguente percorso:

`".../gestione-info/graboidrfc/core/modules/engines/myParser/"`

- Il file di interesse è: **`myParser.py`**

Dataset Generato:

Il dataset prodotto dallo script, contenente i metadati e i corpi dei documenti RFC, è situato nella directory:

`".../gestione-info/graboidrfc/core/data/dataset/"`

- Il file interessati sono: **`dataset.json`**, **`example.json`**

2. Tipologia di Utente e Query Language

2.1. Definizione della Tipologia di Utente

Il nostro motore di ricerca è progettato per soddisfare le esigenze di utenti con poca esperienza o conoscenza del dominio tecnico degli RFC, nello specifico:

- **Studenti universitari**: Studenti che potrebbero necessitare di informazioni specifiche sugli standard e sulle tecnologie di Internet per i loro studi o progetti.
- **Non esperti del settore**: Utenti che non hanno familiarità con gli RFC o con il loro contenuto tecnico ma che hanno necessità di cercare informazioni specifiche.
- **Utenti generici**: Chiunque voglia ottenere informazioni dettagliate presenti negli RFC ma non sappia come localizzarle o in quali documenti siano contenute.

L'obiettivo principale è rendere l'esperienza di ricerca intuitiva e accessibile anche per chi non ha conoscenze approfondite del dominio tecnico.

2.2. Scelta del Query Language

Per soddisfare le esigenze di questa tipologia di utenti, abbiamo optato per un approccio di ricerca basato su:

1. **Keyboard-Based Search**

   Abbiamo scelto di adottare un approccio basato su una ricerca libera (**keyboard-based search**), che consente agli utenti di inserire termini di ricerca nella barra principale in modo diretto e intuitivo. Questo metodo è ideale per chi non ha familiarità con il linguaggio tecnico o con la struttura specifica degli RFC.

2. **Keyboard-Based Search su Campi Specifici**

   Abbiamo inoltre integrato la possibilità di effettuare **ricerche su campi specifici**, come titoli, parole chiave e estratti, per consentire una maggiore precisione nella localizzazione delle informazioni.

2.2.1. Funzionalità di Supporto alla Ricerca

A supporto della ricerca libera, sono state implementate funzionalità avanzate per migliorare ulteriormente l'esperienza utente. Tra queste:

1. **Filtri di Stato**: La possibilità di filtrare i risultati in base allo stato degli RFC, come ad esempio Standard Track, Best Current Practice, Informational, Experimental o Historic.
2. **Filtri Temporali**: Gli utenti possono inoltre limitare i risultati a un determinato periodo temporale, specificando un anno preciso o un intervallo di date, rendendo più facile trovare documenti aggiornati o rilevanti per contesti storici.

La scelta di queste funzionalità e approcci è stata motivata dalla volontà di creare un sistema accessibile, che riduca le barriere tecniche per gli utenti inesperti, senza però sacrificare le capacità avanzate per chi ha esigenze più specifiche.

3. Benchmark

Per valutare le prestazioni del nostro motore di ricerca, abbiamo implementato un sistema di benchmarking basato sui risultati di tre noti motori di ricerca accessibili tramite web: *Google*, *DuckDuckGo* e *Bing*. Questo metodo ci consente di costruire un benchmark costituito da documenti RFC rilevanti per determinate query, associando a ciascun documento un punteggio di rilevanza.

3.1. Obiettivi del Benchmark

L'obiettivo principale è confrontare i risultati dei nostri motori di ricerca con un benchmark che rappresenta uno standard di riferimento. Il benchmark viene costruito raccogliendo i primi risultati restituiti dai tre motori di ricerca per specifiche query e calcolando un punteggio di rilevanza per ciascun documento basato su parametri specifici.

3.2. Costruzione del Banchmark

3.2.1. Fase 1: Recupero dei Risultati

Abbiamo sviluppato uno script, denominato **`autoUrlExtractor.py`**, che utilizza l'API *SerpApi* per effettuare scraping delle pagine di ricerca di *Google*, *DuckDuckGo* e *Bing*. Lo script esegue i seguenti passaggi:

1. Esegue una query predefinita su ciascun motore di ricerca, limitando i risultati al dominio degli RFC (**`site:https://www.rfc-editor.org/rfc/`**).
2. Recupera i primi risultati in formato URL, come ad esempio **`https://www.rfc-editor.org/rfc/rfcNUMBER.html`**, dove **`NUMBER`** rappresenta il numero identificativo dell'RFC.
3. Salva i risultati in un file JSON strutturato, associando ogni query ai relativi URL estratti dai motori di ricerca.

3.2.2. Fase 2: Calcolo della Rilevanza

Il secondo script, denominato **`createBenchMark.py`**, legge il file JSON generato nella fase precedente e calcola la Rilevanza di ciascun documento RFC. Questo processo si basa su due parametri principali: la **Frequenza del Documento** e il **Punteggio basato sulla Posizione** nei risultati.

3.2.2.1. Parametri Indiziali

Per ogni documento RFC, calcoliamo i seguenti parametri:

1. **Frequenza del Documento**: La frequenza indica in quanti motori di ricerca il documento compare tra i risultati. È definita come:

$$\text{Frequenza}_{\text{doc}} = \#\ \text{Numero di motori di ricerca in cui doc compare}$$

2. **Punteggio in base alla Posizione**: Per ciascun motore di ricerca, calcoliamo il punteggio di un documento in base alla sua posizione nei risultati. Il punteggio diminuisce con il crescere della posizione, secondo la seguente formula di decremento logaritmico:

$$\text{PunteggioSingolaPosizione}_{\text{doc,engine}} = \frac{1}{\log_2\left(\text{Posizione}_{\text{engine}}(\text{doc}) + 1\right)}$$

   Qui, la posizione $\mathbf{Posizione_{engine}(doc)}$ rappresenta l'indice del documento nei risultati di un motore di ricerca. Il punteggio totale del documento basato sulle posizioni viene calcolato sommando i singoli punteggi di tutti i motori di ricerca in cui il documento compare:

$$\text{PunteggioTotale}_{\text{doc}} = \sum_{i=1}^{E} \text{PunteggioSingolaPosizione}_{\text{doc},i}$$

3.2.2.2. Punteggio Finale

Successivamente calcoliamo per ciascun documento la Rilevanza combinando il Punteggio Totale e la Frequenza ottenute in precedenza, utilizzando la formula:

$$\text{Rilevanza}_{\text{doc}} = \text{PunteggioTotale}_{\text{rfc}} * (1 + \alpha * \text{Frequenza}_{\text{rfc}})$$

Dove α è un fattore di peso configurabile tra **`0`** e **`1`**, che determina l'importanza della frequenza.

3.2.2.3. Normalizzazione

Le Rilevanze vengono normalizzate tra 1 e 3 per rendere i risultati comparabili e più interpretabili:

$$\text{RilevanzaNormalizzata}_{\text{doc}} = \left\lfloor 2 \cdot \frac{\text{Rilevanza}_{\text{doc}} - \text{Min}}{\text{Max} - \text{Min}} \right\rfloor + 1$$

Dove **Min** e **Max** sono rispettivamente la minima e la massima rilevanza calcolata per i documenti.

3.2.3. Risultato del Benchmark

Alla fine del processo, otteniamo un benchmark strutturato che associa a ogni query una lista di documenti RFC con i relativi punteggi di rilevanza. Questo benchmark fornisce un riferimento oggettivo per confrontare le prestazioni dei nostri motori di ricerca (*Whoosh*, *PyLucene*, *PostgreSQL*) rispetto ai motori di ricerca di riferimento (*Google*, *Duckduckgo*, *Bing*).

- Di seguito un esempio di **output strutturato** dello script:

```
Testo della Query: 'Prompt query 2 site:rfc-editor.org'
Rfc: 9308, Rilevanza: 4.89279, Normalizzata: 3.00000, Arrotondata: 3
Rfc: 9001, Rilevanza: 4.00000, Normalizzata: 2.58616, Arrotondata: 3
⋮
Rfc: 9287, Rilevanza: 2.79203, Normalizzata: 2.02622, Arrotondata: 2
⋮
Rfc: 9297, Rilevanza: 0.60206, Normalizzata: 1.01109, Arrotondata: 1
Rfc: 9443, Rilevanza: 0.57813, Normalizzata: 1.00000, Arrotondata: 1
```

3.2.4. Revisione Manuale dei Risultati

Come fase finale del processo di benchmarking, i risultati generati automaticamente dallo script verranno sottoposti a un controllo manuale. Questa revisione ha l'obiettivo di garantire che i punteggi di rilevanza assegnati ai documenti siano coerenti e rispecchino accuratamente la loro importanza rispetto alle query formulate.

3.3. Locazione dei File Interessati

Gli script responsabili per lo scraping dei risultati ed il calcolo della rilevanza si trovano al seguente percorso:

`".../gestione-info/graboidrfc/core/moduels/engines/myBanchmark/"`

- Gli script: **`benchmark.py`**, **`extractor_online.py`**

`".../gestione-info/graboidrfc/core/data/benchmark/"`

- I file: **`benchmark.json`**, **`extracted_online.json`**, **`extracted_online_revisited.json`**

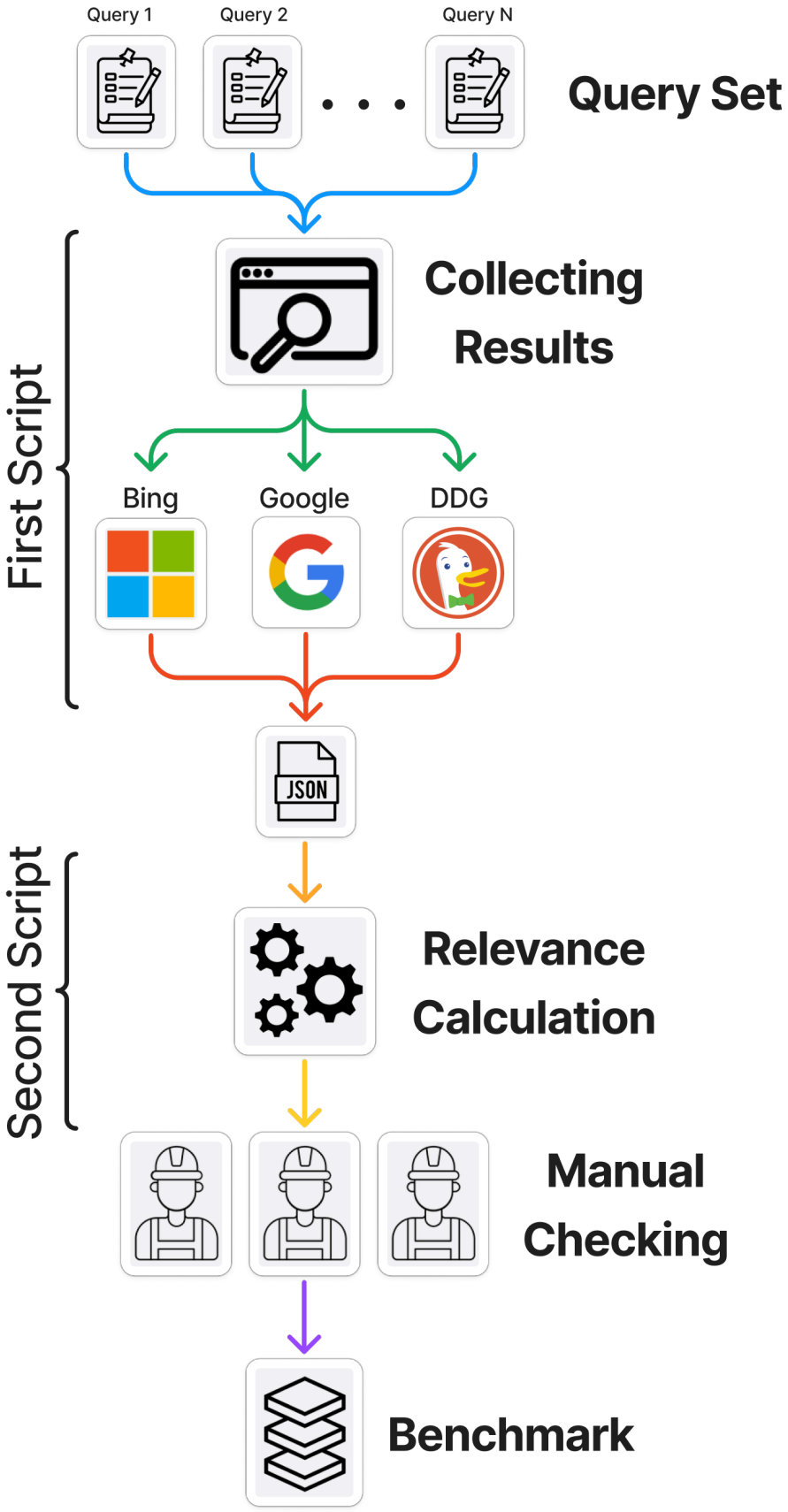

Figura 2: Il diagramma illustra il flusso completo del processo di creazione del benchmark.

4. Queries

Le seguenti query sono state selezionate per mettere alla prova i sistemi da noi sviluppati, permettendo un confronto finale.

4.1. Elenco

1. Query Prima
   - **UIN**: L'utente vuole trovare informazioni dettagliate su TCP Fast Open, un'estensione di TCP che riduce la latenza della connessione iniziale.
   - **MOTIVO**: Questa query testa la capacità del motore di ricerca di recuperare documenti che trattano un'estensione specifica di un protocollo ben noto. La difficoltà sta nel fatto che TCP è molto generico e potrebbe portare a molti risultati irrilevanti.
   - **QUERY**: `"TCP Fast Open"`

2. Query Seconda
   - **UIN**: L'utente vuole trovare informazioni sulle implementazioni dei protocolli di autenticazione EAP e RADIUS in reti Wi-Fi sicure.
   - **MOTIVO**: Questa query stressa il motore perché richiede di trovare documenti che trattano EAP e RADIUS specificamente nel contesto delle reti Wi-Fi, filtrando documenti generici sulla sicurezza di rete.
   - **QUERY**: `"EAP RADIUS Wi-Fi security"`

3. Query Terza
   - **UIN**: L'utente cerca informazioni dettagliate sulla sicurezza di TLS 1.3 e sulle principali differenze rispetto a TLS 1.2.
   - **MOTIVO**: Query specifica che richiede distinzione tra più versioni di un protocollo, valutando la gestione di documenti con termini simili.
   - **QUERY**: `"TLS 1.3 security differences from TLS 1.2"`

4. Query Quarta
   - **UIN**: L'utente vuole un elenco di tutti gli RFC che trattano autenticazione e crittografia nei protocolli di posta elettronica come SMTP, POP3 e IMAP.
   - **MOTIVO**: Questa query stressa il motore di ricerca perché deve collegare concetti di sicurezza (autenticazione e crittografia) con protocolli email specifici. Inoltre, diversi RFC trattano ciascun protocollo separatamente.
   - **QUERY**: `"SMTP POP3 IMAP authentication encryption"`

5. Query Quinta
   - **UIN**: L'utente vuole esplorare la relazione tra sicurezza nei protocolli di routing e il protocollo BGP, cercando RFC che trattino entrambi i temi.
   - **MOTIVO**: Questa query combina due concetti correlati ma trattati in RFC separati. Stressa il motore di ricerca nel riconoscere documenti che parlano di sicurezza nel routing senza escludere BGP.
   - **QUERY**: `"BGP security routing"`

6. Query Sesta
    - **UIN**: L'utente vuole confrontare IPv4 e IPv6 in termini di gestione della frammentazione dei pacchetti.
    - **MOTIVO**: La query sfida il motore nel trovare documenti che menzionano sia IPv4 che IPv6, ma in un contesto specifico come la frammentazione, evitando documenti che parlano di IPv6 in generale.
    - **QUERY**: `"IPv4 IPv6 fragmentation"`

7. Query Settima
    - **UIN**: L'utente vuole capire come i protocolli di controllo della congestione hanno influenzato lo sviluppo di QUIC.
    - **MOTIVO**: La query è difficile perché lega concetti di performance e ottimizzazione della rete con QUIC, richiedendo di trovare documenti che parlano di controllo della congestione e del loro impatto su QUIC.
    - **QUERY**: `"QUIC congestion control impact"`

8. Query Ottava
    - **UIN**: L'utente vuole approfondire il funzionamento del protocollo TCP con particolare attenzione ai meccanismi di controllo della congestione utilizzati per ottimizzare le prestazioni di rete.
    - **MOTIVO**: Query multi-campo che testa la capacità del motore di identificare documenti specifici sul controllo della congestione in TCP. Il motore deve bilanciare la rilevanza tra contenuto, titolo e abstract.
    - **QUERY**:`"TCP protocol and congestion control title:TCP abstract:Congestion Control"`

9. Query Nona
    - **UIN**: L'utente è interessato alla gestione degli indirizzi IPv6 e alla sua evoluzione rispetto alle versioni precedenti del protocollo IP, con particolare riferimento agli RFC.
    - **MOTIVO**: Query multi-campo che valuta la capacità del motore di recuperare documenti focalizzati sia sulla gestione degli indirizzi IPv6 che sulla sua evoluzione nel tempo. Il campo abstract fornisce un contesto più generale sulla rete.
    - **QUERY**:`"IPv6 address management and evolution title:IPv6 abstract:network"`

10. Query Decima
    - **UIN**: L'utente vuole analizzare le diverse versioni del protocollo TLS, le differenze tra loro e il livello di sicurezza offerto da ciascuna.
    - **MOTIVO**: Query multi-campo che verifica la capacità del motore di classificare documenti sulle versioni di TLS e i loro miglioramenti in termini di sicurezza, sfruttando il campo delle keyword per migliorare la pertinenza dei risultati.
    - **QUERY**: `"TLS protocol versions and security keywords:security"`

5. Motori di Ricerca

L'obiettivo è progettare, sviluppare e confrontare diverse varianti di tre motori di ricerca, analizzandone il comportamento in relazione agli UIN utente precedentemente enunciati.

I tre motori di ricerca presi in esame si basano sui seguenti sistemi: *Whoosh*, *PyLucene* e *PostgreSQL*. Questi sistemi consentono l'indicizzazione e la ricerca efficiente dei documenti.

Struttura e differenze dei sistemi

- **Interfaccia comune**: I tre sistemi condividono la stessa interfaccia web, garantendo un'esperienza uniforme e il supporto per tutti i filtri elencati nella sezione Scelta del Query Language.
- **Preprocessing indipendente**: Ogni sistema indicizza i documenti del dataset applicando il proprio meccanismo di preprocessing predefinito, senza una fase di normalizzazione comune tra i tre motori.

Varianti sviluppate

Per ciascuno dei tre sistemi, abbiamo implementato due varianti basate su altrettanti modelli di ranking. Per ciascuno dei modelli di ranking abbiamo progettato due varianti:

- Una **prima variante** basata sull'algoritmo di ranking standard.
- Una **seconda variante** dello stesso modello standard, modificata attraverso personalizzazioni della formula o dei parametri di ranking.

  Ad esempio, nel caso di *Whoosh*, abbiamo scelto il modello di ranking *TF-IDF*, che calcola la rilevanza di un documento sommando i contributi dei pesi dei singoli termini, determinati tramite la formula *TF-IDF*.

  Successivamente, abbiamo sviluppato una versione modificata moltiplicando la formula per un parametro aggiuntivo, il Freshness Factor, che favorisce i documenti più recenti rispetto a quelli meno recenti. Abbiamo denominato questa variante *TF-IDF-FF*.

Successivamente, analizzeremo e confronteremo il comportamento delle varianti rispetto agli UIN utente, impiegando metriche appropriate e rappresentandone i risultati attraverso grafici intuitivi.

Nei prossimi paragrafi presenteremo e discuteremo le caratteristiche delle varianti sviluppate.

5.1. Motore Whoosh

Per il motore di ricerca *Whoosh*, abbiamo selezionato due modelli di ranking implementati di default nel sistema: *BM25* e *TF-IDF*.

Locazione dello Script

Lo script che gestisce l'integrazione con *Whoosh* è situato al seguente percorso:

`".../gestione-info/graboidrfc/core/moduels/engines/myWhoosh/myWhoosh.py"`

5.1.1. Modello BM25

5.1.2. Modello BM25 Custom

5.1.3. Modello TF-IDF

5.1.4. Modello TF-IDF Custom (TF-IDF-FF)

5.2. Motore PyLucene

Per il motore di ricerca *PyLucene*, abbiamo selezionato due modelli di ranking implementati di default nel sistema: *BM25* e *VSM*.

Locazione dello Script

Lo script che gestisce l'integrazione con *PyLucene* è situato al seguente percorso:

`".../gestione-info/graboidrfc/core/moduels/engines/myPylucene/myPylucene.py"`

5.2.1. Modello BM25

5.2.2. Modello BM25 Custom

5.2.3. Modello VSM

5.2.4. Modello VSM Custom (TFLN-PIDF)

5.3. Motore PostgreSQL

Per il motore di ricerca *PostgreSQL*, abbiamo selezionato due modelli di ranking implementati di default nel sistema: *BM25* e *TF-IDF*.

Locazione dello Script

Lo script che gestisce l'integrazione con *PostgreSQL* è situato al seguente percorso:

`".../gestione-info/graboidrfc/core/moduels/engines/myPostgres/myPostgres.py"`

5.3.1. Modello BM25

5.3.2. Modello BM25 Custom (DLN)

5.3.3. Modello TF-IDF

5.3.4. Modello TF-IDF Custom (DLN)

6. Analisi Performance