UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

IMPLEMENTAZIONE E ANALISI DI METODI PER LA SCOPERTA DI K-MERS DISCRIMINATIVI NELLA CLASSIFICAZIONE DI SEQUENZE METAGENOMICHE

Relatore: Prof. Matteo Comin

Laureando: Davide Marchiori

Anno accademico 2015/2016

Indice

Elenco delle Figure

Elenco delle Tabelle

Sommario

L’argomento sviluppato nel corso del lavoro di tesi riguarda la classificazione di sequenze metagenomiche tramite allineamento di k-mers con particolare attenzione alla selezione delle informazioni e alla costruzione della base di dati compressa.

L’obiettivo prefissato è quello di studiare ed implementare delle nuove modalità di costruzione e di classificazione che, tramite una selezione mirata delle informazioni, riesca a migliorare le prestazioni rispetto all’algoritmo originale utilizzato dal classificatore ultraveloce Kraken.

Il capitolo 1 prevede un’introduzione generale al concetto di metagenomica, alla descrizione delle varie fasi che compongono un progetto metagenomico e alla descrizione dell’NCBI.

Il capitolo 2 è dedicato alla descrizione dettagliata del funzionamento dell’algoritmo originale utilizzato da Kraken: partendo dalla creazione della base di dati compressa indicizzata, arrivando alla metodologia utilizzata per la classificazione delle sequenze metagenomiche.

Il capitolo 3 descrive il concetto di discriminatività adottato, propone possibili estensioni dell’algoritmo originale di Kraken e descrive dettagliatamente il loro funzionamento. Le nuove varianti intraprenderanno alcune scelte, di creazione del database compresso e di classificazione, basandosi sulla discriminatività dei k-mers delle sequenze di riferimento.

Nel capitolo 4 vengono descritti i test effettuati con le nuove modalità implementate, confrontando i risultati ottenuti con l'algoritmo originale. Vengono analizzati i parametri statistici derivanti dalle classificazioni effettuate su dataset reali e artificiali al fine di valutare un miglioramento prestazionale del classificatore originale tramite le nuove tecniche proposte.

I risultati ottenuti sono incoraggianti e permettono una possibile visione di sviluppo ulteriore.

Capitolo 1 Introduzione

Questo capitolo è dedicato alla definizione generale di metagenomica, alla descrizione delle singole fasi che costituiscono un progetto metagenomico, all'introduzione dell'NCBI e dei materiali che mette a disposizione della comunità scientifica.

1.1 La metagenomica

La metagenomica è lo studio basato sull'utilizzo di tecniche genomiche moderne per l'analisi delle comunità microbiotiche presenti in un campione ambientale. Fornisce la possibilità dell'analisi genomica di specie che altrimenti non sarebbero coltivabili o isolabili singolarmente in laboratorio.

Lo sviluppo di nuove tecniche di sequenziamento dell'acido desossiribonucleico (DNA) da campioni ambientali è un fattore cruciale per la scoperta di un eccezionale grado di diversità tra le specie. Le comunità microbiotiche possono presentare un alto grado di variazioni spaziali/temporali nella loro composizione tassonomica.

Un passo fondamentale nell'analisi metagenomica è la comprensione della struttura microbiologica di una comunità appartenente ad un dato ambiente e la caratterizzazione dei vari elementi che vi risiedono, quantificandone la loro diversità in termini di ricchezza/abbondanza di specie.

Combinati con la capacità dei moderni sequenziatori di ottenere frammenti di DNA molto velocemente, progetti di metagenomica possono generare una quantità enorme di dati che

descrivono questi mondi prima invisibili. Molte sequenze genomiche presentano similarità con quelle già studiate e sono individuabili mediante l'utilizzo di algoritmi di allineamento dotati di elevata sensitività.

Questo approccio da origine ad una serie di studi importanti: si spazia dalla necessità di capire l'esatto funzionamento di specifiche regioni di DNA, confrontando con specie già conosciute e studiate, a comprendere come l'ambiente influisce sulla composizione genomica di intere specie. Il problema di classificazione e determinazione dell'origine di una sequenza di DNA dato un insieme di genomi conosciuti è comune a molti campi della biologia computazionale molecolare.

Algoritmi e tecniche recenti sono stati proposti con l'obiettivo di incrementare i valori di accuratezza ottenuti da classificatori già esistenti.

A seconda del contesto dello studio metagenomico si è in grado di ottenere risultati di varia natura. In un contesto clinico, ad esempio, la classificazione aiuta ad identificare come la presenza e l'abbondanza di diverse specie all'interno di un microbioma siano responsabili dell'insorgere e del progredire di diverse malattie e/o disordini fisiologici.

La metagenomica sta cambiando il modo con cui i ricercatori affrontano molti problemi, ridefinendo il concetto stesso di marcatore genetico ed accelerando il tasso di scoperta di funzionamento dei microbiomi.

1.2 Progetto metagenomico

Un progetto metagenomico si differenzia dai tradizionali progetti di sequenziamento microbiotico in molti aspetti. Il materiale di partenza è un insieme di sequenze di DNA proveniente da una comunità di organismi che può includere batteri, archeobatteri, eucarioti e specie virali a vari livelli di diversità e di quantità. Molti di questi organismi non risultano coltivabili in laboratorio. In alcuni progetti il campionamento del DNA può portare a contaminazioni dovute alla presenza di DNA proveniente da specie non volutamente analizzate ma che possono essere presenti nel campione ambientale. Questo fa comprendere quanto possa essere difficile ottenere sequenze genomiche da un progetto metagenomico. Spesso il reale scopo non è quello di generare genomi completi,

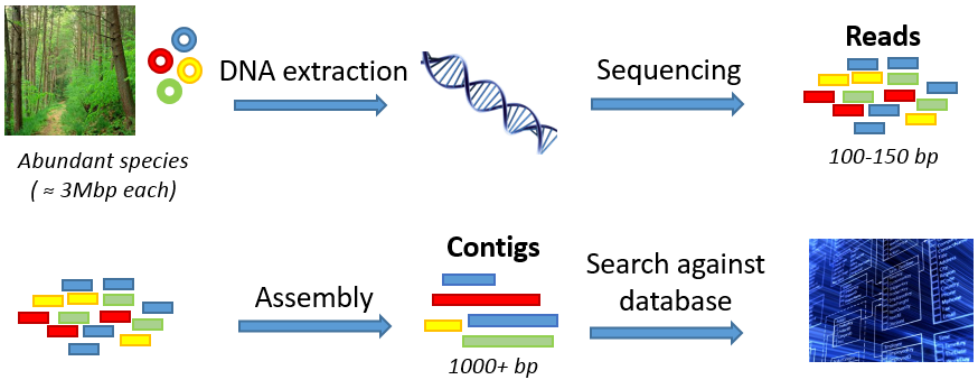

Figura *1.1 Fasi essenziali di un progetto metagenomico.*

ma di capire la composizione della comunità di organismi e le loro interazioni con l'ambiente.

In un progetto metagenomico si possono individuare sei fasi principali:

1. Campionamento
2. Estrazione
3. Sequenziamento
4. Assemblaggio
5. Binning

1.2.1 Campionamento

Il processo di campionamento è il primo ed essenziale passo in ogni progetto metagenomico volto a preservare la qualità del DNA. Il DNA estratto dovrebbe essere rappresentativo di tutte le specie presenti nel campione e fornire una quantità sufficiente di acidi nucleici di elevata qualità che verranno utilizzati poi per il sequenziamento.

In questa fase si raccolgono i cosiddetti "metadati" che possono comprendere informazioni fisiche, chimiche, caratteristiche ambientali, ora e luogo del campionamento, profondità, intensità della luce, pH, patologia e così via. Una conoscenza dettagliata dell'ambiente è essenziale per l'interpretazione biologica che verrà data successivamente ai dati risultanti dal progetto metagenomico.

1.2.2 Estrazione

Ottenuto il campione ambientale si procede all'estrazione degli acidi nucleici contenuti e alla loro purificazione mediante la separazione dagli altri componenti cellulari.

Le metodologie per tale operazione si suddividono in tre fasi:

1. **Lisi cellulare** (spesso combinata con l'inattivazione delle nucleasi cellulari)
2. **Deproteinizzazione** del campione lisato
3. **Precipitazione** dell'acido nucleico

La procedura di lisi utilizzata deve essere abbastanza aggressiva da poter frammentare il materiale di partenza ma, allo stesso tempo, adeguatamente delicata per essere in grado di mantenere l'integrità dell'acido nucleico. Tecniche comuni di lisi si distinguono in base a:

1. Degradazione meccanica (lisi ipotonica)
2. Trattamenti chimici (lisi con detergenti)
3. Digestione enzimatica (lisi con enzimi)

Dalla rottura delle parenti cellulari e della membrana plasmatica si ottiene una miscela complessa costituita da varie componenti cellulari (DNA, RNA, lipidi, carboidrati, proteine).

La rimozione delle proteine dal lisato cellulare è un passaggio particolarmente importante, in quanto la cellula contiene sia enzimi in grado di degradare gli acidi nucleici, sia proteine in grado di legarsi con gli acidi nucleici e che quindi potrebbero interferire con i trattamenti enzimatici successivi. Gli acidi nucleici in soluzione acquosa vengono recuperati mediante precipitazione in appositi composti chimici. In queste condizioni il DNA tende ad aggregarsi in precipitati che possono essere raccolti mediante una bacchetta di vetro o recuperati mediante centrifugazione.

1.2.3 Sequenziamento

Il sequenziamento è il procedimento che consente di decifrare la sequenza delle basi in un determinato segmento di DNA e quindi determinare l'ordine dei diversi nucleotidi che la compongono (le quattro basi azotate: Adenina, Citosina, Guanina, Timina). Per anni la metodica principalmente utilizzata, si basò principalmente sul metodo della

terminazione della catena sviluppato da Frederick Sanger. Questa tecnica sfrutta nucleotidi modificati per interrompere la reazione di sintesi in posizioni specifiche lungo la sequenza. Il sequenziamento tramite il metodo Sanger ha la capacità di sequenziare frammenti fino a 1000 coppie di basi.

Con l'evolversi della tecnologia negli ultimi anni si sono sviluppate nuove metodiche definite come "sequenziamento ad alto parallelismo" in grado di produrre enormi quantità di sequenze di lunghezza minore rispetto al metodo Sanger ma ad un costo nettamente inferiore e soprattutto ad una velocità superiore.

Il pirosequenziamento fu largamente utilizzato perché in grado di generare delle reads (frammenti di sequenze genomiche) molto più lunghe delle piattaforme concorrenti. In altri casi, progetti metagenomici in larga scala, furono preferite le piattaforme Illumia e SOLiD nonostante creassero delle reads più corte. Il motivo di questa scelta fu l'elevato throughput e copertura che furono in grado di offrire allo stesso prezzo. Questo ha reso accessibile il sequenziamento di una enorme quantità di genomi che ha dato vita ad una espansione del settore.

Sequenziatori di nuova generazione (Illumia, SOLiD e Roche 454) sono diventati strumenti essenziali per analisi approfondite nel campo della biologia e della medicina consentendo studi prima inaccessibili. La gestione e l'analisi dell'enorme quantità di dati ricavati da queste elaborazioni hanno determinato la nascita e lo sviluppo della bioinformatica, disciplina che integra competenze informatiche, statistiche e analitiche per risolvere problemi di tipo biologico.

1.2.4 Assemblaggio

L'assemblaggio delle sequenze si riferisce all'allineamento e alla fusione di frammenti di DNA (read) fino a ricostruire la sequenza intera originale.

Ottenute le reads dal sequenziamento (di lunghezza variabile 50 – 1000 bp circa a seconda del metodo di sequenziamento utilizzato) si procede al loro assemblaggio.

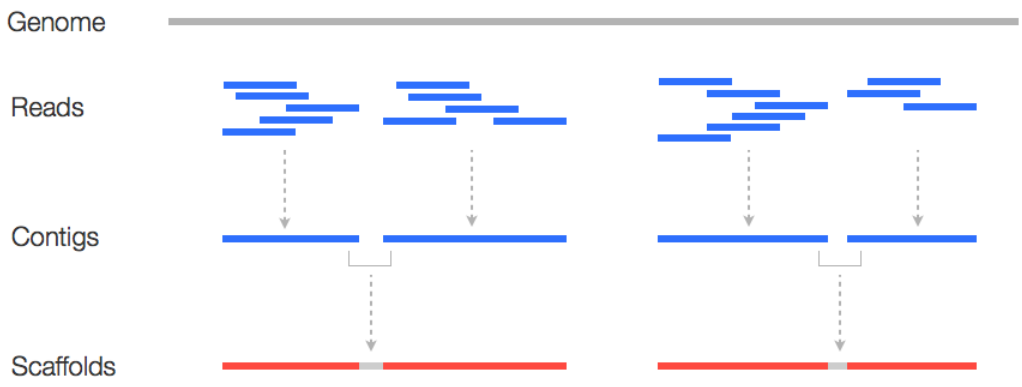

Figura.*1.2 Esempio di Assemblaggio.*

Questa fase permette di riordinare le reads e ottenere le sequenze genomiche complete. Il processo viene eseguito per passi successivi: inizialmente dalle reads si creano sequenze contigue più lunghe chiamate contings (insieme contiguo di reads sovrapponibili), che a loro volta vengono assemblate per formare sequenze ancora più lunghe chiamate scaffolds.

Esistono due strategie per affrontare questo tipo di problema:

- Reference-based assembly
- De novo assembly

Il Reference-based assembly sfrutta sequenze genomiche già conosciute, presenti all'interno di dataset di riferimento, e molto "vicine genomicamente" (ad esempio variazione della stessa specie) alla sequenza da classificare. Esistono comunque differenze tra il genoma campionato e quello di riferimento nel dataset e questo può portare ad intere regioni non coperte da quello conosciuto.

Il De novo assembly provvede a ricreare sequenze più lunghe non avendo nessun genoma di riferimento. Questo approccio richiede capacità di calcolo e di elaborazioni molto più elevate rispetto alla tecnica precedente. Sono stati sviluppati algoritmi appositi alla risoluzione di questo problema basati su grafi di Brujin.

L'assemblaggio resta comunque un problema computazionale molto oneroso in quanto:

- Le sequenze possono contenere errori
- DNA ripetitivo causa grossi problemi
- Alcune regioni possono rimanere scoperte (creare quindi gaps)

La fase dell'annotazione permette di conoscere la localizzazione, la struttura e la funzionalità di tutti gli elementi che compongono l'intero genoma come ad esempio: geni codificanti proteine, geni non codificanti proteine, elementi regolatori, elementi ripetuti, pseudogeni etc.

1.2.5 Binning

Campioni metagenomici possono contenere reads provenienti da un enorme numero di organismi. Ad esempio in un singolo grammo di terreno ci possono essere fino a 18000 organismi differenti, ognuno con il proprio genoma. In molti casi l'assemblaggio di singoli geni è molto difficile data l'incompletezza delle sequenze. Le tecniche di binning rappresentano un modo efficace per identificare read o contig di un certo gruppo di organismi designate come OTUs (Operational Taxonomic Units).

All'interno dei genomi di ciascuna specie sono presenti geni specifici (sequenze di acidi nucleici con un preciso ordine) che la identificano e la caratterizzano univocamente. Sfruttando queste informazioni è possibile individuare organismi che sono presenti all'interno del campione con maggiore precisione.

Tecniche moderne di binning sfruttano informazioni:

- Già disponibili e indipendenti dal campione metagenomico
- Intrinseche ricavate dal campione stesso

Il loro grado di successo varia a seconda della diversità e della complessità del campione. In alcuni casi sono in grado di identificare sequenze fino a raggiungere specie specifiche, in altri le sequenze sono identificate come appartenenti ad un gruppo tassonomico più grande.

Negli anni sono stati sviluppati numerosi algoritmi che risolvono il problema del

binning metagenomico. Alcuni di essi agiscono come classificatori supervisionati, fruttando informazioni già conosciute, oppure come classificatori non-supervisionati, tentando di trovare nuovi gruppi. I classificatori sfruttano sequenze già conosciute effettuando allineamenti con esse e tentando di individuare caratteristiche del DNA delle sequenze campionate, come ad esempio il GC-content (Guanine-Cytosine content, indicatore utilizzato in biologia molecolare).

Un esempio di classificatore statistico è TETRA. Utilizza i pattern dei tetranucleotidi nei frammenti genomici. Essendoci quattro possibili nucleotidi nel DNA (le quattro basi azotate) le combinazioni di differenti possibili tetranucleotidi (quattro nucleotidi in sequenza) sono 256. Il classificatore calcola la frequenza di ogni tetramer (sequenza di lunghezza quattro) di una data sequenza. Ad ognuno di questi tetramer assegna un punteggio e questo viene confrontato con altri valori, generati da genomi già classificati, ed in base alle differenze riscontrate è possibile stabilire se la sequenza appartiene o meno ad un determinato organismo o ad un OTUs. Altri classificatori utilizzano l'allineamento di basi cercando similarità tra la sequenza da analizzare e le sequenze conosciute. Il genoma con maggiore similarità identificherà la specie o l'OTU di appartenenza. Questo approccio porta vantaggi dal punto di vista della precisione dei risultati ma introduce tecniche molto dispendiose dal punto di vista computazionale. Esempi di algoritmi che sfruttano questa tecnica sono: SOrt-ITEMs, DiScRIBinATE e ProViDE. Altri sfruttano metodologie miste, ad esempio SPHINX, ed altri ancora anche le SVN (Support Vector Machine) come Phylopythia.

Rientra nella fase di binning anche la classificazione che permette di etichettare ogni sequenza con un identificatore tassonomico. Tradizionalmente questo problema viene affrontato mediante l'uso di algoritmi basati sull'allineamento locale delle sequenze a livello del nucleotide, confrontando reads ricavate dal sequenziamento con genomi microbiotici conosciuti. Algoritmi di pattern matching (utilizzati per l'allineamento di sequenze piccole dimensioni) non sono utilizzabili in questo contesto perché l'enorme quantità di dati presente genererebbe una complessità computazione elevata non sostenibile. Questa problematica ha richiesto lo sviluppo di algoritmi e tecniche statistiche adeguate.

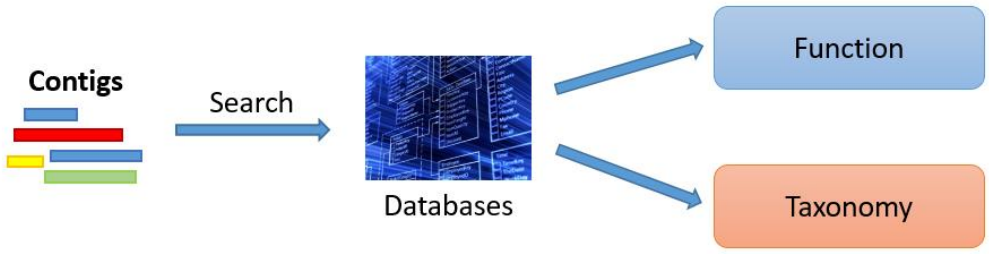

Figura *1.3 Esempio di Binning.*

Classificatori metagenomici moderni sfruttano strutture dati indicizzate tramite funzioni di hash, per velocizzare la ricerca, create a partire da database contenenti genomi completi di riferimento. Per raggiungere elevate velocità di elaborazione non vengono usati i tradizionali metodi di allineamento locale ma ci si basa sull'identificazione di k-mers (sequenze genomiche di lunghezza k). Nella prima fase vengono preprocessati i genomi di riferimento estraendo tutti i k-mers contenuti e memorizzati in strutture dati compresse indicizzate. Nella seconda viene assegnata un'etichetta tassonomica (mediante diverse metodologie) ad ogni k-mer contenuto nella sequenza da classificare arrivando quindi ad ottenere l'etichetta tassonomica finale dell'intera sequenza tramite valutazioni statistiche.

Etichette tassonomiche identificano un singolo individuo oppure gruppi dell'albero tassonomico generale. Un esempio di rank tassonomici standard utilizzati in biologia in figura 1.4.

Nei database genomici reali possono esistere alcuni sottogruppi di ognuno di questi livelli principali. Essi nascono per motivi di organizzazione di dati.

Il classificatore Kraken, presentato nel capitolo successivo, ha un approccio di risoluzione del problema molto simile a quello appena accennato.

1.3 NCBI

Il "National Center for Biotechnology Information" (NCBI) è parte del "United States National Library of Medicine" (NLM), un ramo dell'istituto nazionale della sanità degli Stati

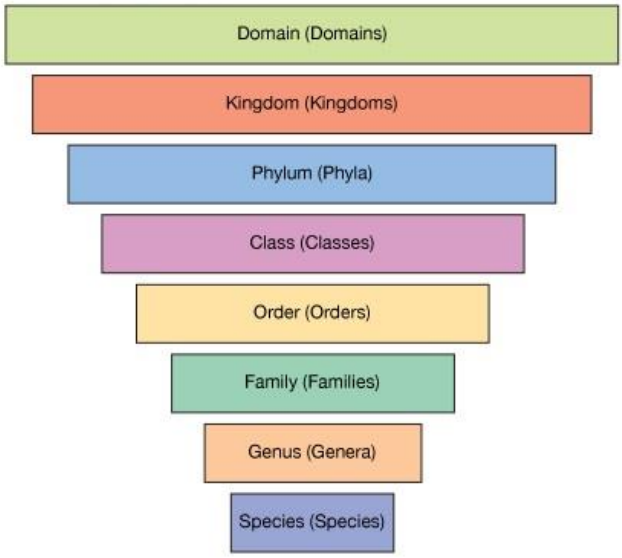

Figura *1.4 Rank tassonomici standard utilizzati in biologia.*

Uniti d'America. Fu fondato nel 1988 e ha sede a Bethesda nel Maryland. L'NCBI fa parte dell'INSDC (International Nucleotide Sequence Database Collaboration), punto di giuntura tra i maggiori database di sequenze di DNA e RNA disseminati nel mondo. Ha permesso la creazione e la comunicazione dei seguenti database computerizzati:

- DNA Data Bank of Japan (Japan)
- GenBank (USA)
- European Nucleotide Archive (United Kingdom)

I database sono quotidianamente sincronizzati tra loro, formando un'unica base di dati che mette a disposizione contributi e aggiornamenti di ogni team di ricerca.

L'NCBI fornisce una serie di database rilevanti per la biotecnologia e la biomedicina ed è un'importante risorsa per strumenti bioinformatici e servizi.

I database forniti di maggior rilievo sono:

1. GenBank
2. PubMed
3. RefSeq

GenBank è un database di libero accesso contenente la collezione annotata di tutte le sequenze di nucleotidi conosciute disponibili pubblicamente. È diventata la più importante ed influente base di dati di riferimento per ricercatori nei vari campi della biologia.

PubMed è un database contente citazioni di articoli scientifici che attualmente ha oltre 24 milioni di riferimenti bibliografici derivati da più di 5300 periodici biomedici. Fu inizialmente progettato per garantire un accesso a riferimenti ad articoli di carattere scientifico e biomedico. Il database MEDLINE è la principale risorsa per PubMed che copre campi come: medicina, odontoiatria, infermieristica, sistema di assistenza sanitaria, veterinaria e scienze precliniche.

RefSeq è anch'esso un database ad accesso libero di sequenze di nucleotidi (DNA, RNA). Diversamente da GenBank fornisce una collezione di record per ogni molecola biologica naturale (DNA, RNA o proteine) per i maggiori organismi spaziando dai virus, ai batteri fino agli eucarioti. RefSeq si limita ad organismi dove sono presenti abbastanza informazioni (genomi completi) mentre GenBank contiene sequenze di ogni specie studiata, anche quelle dove esistono ancora genomi parziali.

Tutti i database sono disponibili online attraverso un motore di ricerca interno, chiamato Entrez, ed è possibile avere un collegamento diretto tra le varie basi di dati per effettuare ricerche incrociate.

Tra i programmi sviluppati dall'NCBI vi è BLAST (Basic Local Alignment Search Tool), un algoritmo usato per comparare le informazioni contenute nelle strutture biologiche primarie, come ad esempio le sequenze proteiche o le sequenze nucleotidiche delle molecole di DNA. BLAST si presenta attualmente come uno dei più utilizzati algoritmi nel campo della bioinformatica e con il diversificarsi delle funzioni ne sono state proposte molte varianti.

Capitolo 2 Kraken

Questo capitolo è dedicato alla descrizione dettagliata del funzionamento dell'algoritmo originale utilizzato da Kraken: partendo dalla creazione della base di dati compressa indicizzata, arrivando alla metodologia utilizzata per la classificazione delle sequenze metagenomiche.

2.1 Introduzione generale

Kraken è un algoritmo di classificazione di sequenze metagenomiche che usa una metodologia di risoluzione basata sull'allineamento esatto. Scritto da Derrick Wood e Steven L. Salzberg attualmente è alla versione 0.10.6 beta. Si differenzia nel panorama dei classificatori già esistenti per via della sua alta velocità di classificazione, precision e sensitivity. La soddisfazione di entrambi questi valori statistici è molto difficile e solitamente, si tende a privilegiare uno rispetto all'altro oppure cercare un compromesso tra i due.

Come tutti i classificatori metagenomici, anche Kraken necessita di risorse di calcolo non indifferenti. Basti pensare che per costruire un database compresso di tutti i genomi dei batteri si necessitano più di 70 GB di memoria primaria e almeno 160 GB di memoria secondaria, quantità che si ottengono solo tramite l'utilizzo di supercalcolatori. Se si vuole sfruttare al massimo la velocità offerta dall'algoritmo i database compressi intermedi devono essere caricati tutti in memoria primaria portando quindi la necessità di risorse ad almeno 160 GB di RAM. La logica del funzionamento dell'algoritmo di Kraken si può

Classifier	Genus Precision	Genus Sensitivity	Speed (reads/min)
Naïve Bayes Classifier	97,64	97,64	7
PhymmBL	96,11	96,11	76
PhymmBL (conf. > 0.65)	99,08	95,45	76
Megablast w/ best hit	96,93	93,67	4511
Kraken	**99,90**	**91,25**	**1307161**
Kraken (quick operation)	**99,92**	**89,54**	**4101162**
MiniKraken (Kraken w/4GB DB)	**99,95**	**65,87**	**1441476**
MiniKraken (quick operation)	**99,98**	**65,31**	**2693119**
MetaPhlAn	n/a	n/a	370770

Tabella 2.1 Confronto prestazioni Kraken con altri classificatori.

suddividere in due fasi distinte:

1. Costruzione della base di dati compressa
2. Classificazione

La prima fase mira a costruire una base di dati compressa che verrà poi utilizzata durante la classificazione per assegnare etichette tassonomiche a sequenze genomiche sconosciute. Tale costruzione è fatta in maniera tale da rendere performante la ricerca di dati all'interno del database e, soprattutto, efficiente il metodo di classificazione statistica. Entrambe le fasi verranno descritte maggiormente in dettaglio nel capitolo.

Kraken presenta una variante dell'algoritmo di classificazione originale chiamato "Quick mode". Permette di velocizzare notevolmente (fino a quasi quattro volte) la classificazione andando a modificare leggermente il metodo di assegnamento delle etichette tassonomiche a sequenze da classificare perdendo valore di sensitivity.

Un obiettivo importante nella metagenomica è la scoperta di nuovi organismi e la classificazione di essi è una sfida enorme per ogni classificatore. Essendo la classificazione basata su informazioni derivanti da specie già conosciute presenti nel database di riferimento un'assegnazione tassonomica corretta a livello specie risulterà impossibile se questa non è già conosciuta. Un ottimo risultato è quello di riuscire a classificare correttamente il livello genere. Gli autori di Kraken hanno condotto test per studiare il comportamento del classificatore alla presenza di specie sconosciute nella classificazione a vari livelli tassonomici.

2.2 Costruzione del Database

La costruzione della base di dati compressa utilizzata da Kraken è la fase più computazionalmente onerosa. Essa si distingue in sei fasi differenti e il flusso dei dati è gestito in maniera tale da permettere la creazione di check-point intermedi tra i vari step. Essendo la quantità di dati molto grande, e il tempo di elaborazione altrettanto, questo approccio si è reso necessario al fine di permettere all'utente, in caso di errore ad uno step intermedio, di riprendere il processo di costruzione senza dover ricominciare dall'inizio tutta l'elaborazione ma dall'ultimo check-point andato a buon fine.

Kraken necessita di database genomici di riferimento da cui partire per creare la propria base di dati di k-mers compressa e delle relative informazioni tassonomiche. L'NCBI (National Center for Biotechnology Information) mette a disposizione tutte le librerie necessarie a questo scopo.

La velocità di creazione della base di dati compressa e di classificazione è fortemente influenzata sia dalla quantità di dati da processare, sia dall'elaboratore su cui si esegue l'algoritmo. Alcune porzioni di codice sfruttano la presenza di più processori o thread nel calcolatore, parallelizzando l'esecuzione di operazioni e velocizzando l'intero processo.

La creazione del database si sviluppa nei seguenti sei step:

1. Creazione della base di dati contenente i k-mers
2. Riduzione delle dimensioni della base di dati (unico step opzionale)
3. Ordinamento dell'insieme di k-mers con creazione del file di indice
4. Creazione della mappa che associa identificativi genomici (GI) e identificativi delle sequenze (seqID)
5. Creazione della mappa che associa identificativi delle sequenze (seqID) e identificativi tassonomici (taxID)
6. Associazione degli identificativi tassonomici (taxID) dei Lower Common Ancestors (LCAs) ai k-mers della base di dati compressa

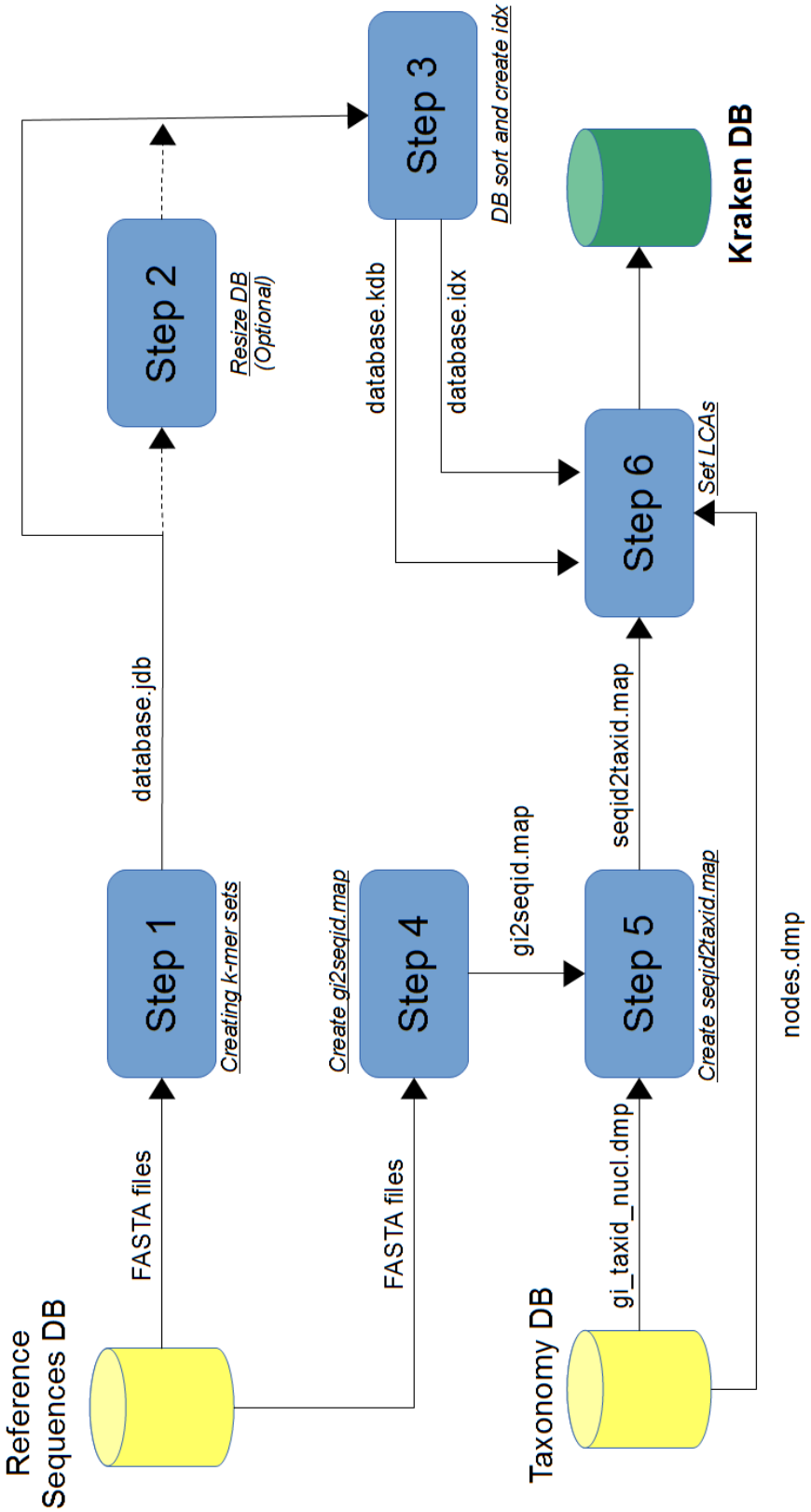

Figura 2.1 Schema delle fasi di costruzione del database di Kraken.

2.2.1 Step 1 - Creazione k-mers set

Il primo passo crea il file contente tutti i possibili k-mers distinti presenti nelle sequenze dei dataset di riferimento. Il parametro 'k', determina la lunghezza dei k-mer ed è liberamente settabile dall'utente (fino ad un valore massimo di 31). Kraken utilizza un programma esterno, chiamato Jellyfish, che esegue questa operazione in maniera efficiente creando una base di dati compressa formata da coppie di valori. Il primo identifica ogni k-mer distinto ed il secondo il numero delle sue occorrenze nelle sequenze ottenute durante l'intero processo.

Stringhe di sequenze genomiche sono costituite da un alfabeto di soli quattro caratteri distinti (le basi azotate del DNA). Sfruttando questa caratteristica si può minimizzare al massimo lo spazio necessario alla memorizzazione di un solo carattere a due bit, riducendo di quattro volte lo spazio standard di un byte utilizzato per una variabile di tipo char.

Il file di output contente la base di dati compressa sarà chiamato *"database.jdb"*.

Le informazioni contenute nell'header del file di output utili per la creazione del database finale di Kraken sono:

- Stringa di controllo, necessaria per verificare che il file da leggere sia nel formato corretto. Nel file di Jellyfish contiene la stringa 'JFLISTDN';
- Key_bits, valore minimo di bit necessari a rappresentare i k-mers, considerando che per ogni lettera sono necessari solo 2 bit;
- Val_len, quantità in byte necessaria a rappresentare il valore associato al k-mer (che a questo step consiste nel numero di occorrenze del k-mer stesso);
- Key_ct, quantità di k-mer distinti nel database.

Da questa struttura si capisce come la base di dati sia gestita con l'obiettivo di ridurre al minimo i byte necessari alla memorizzazione di tutte le informazioni riguardanti i k-mer. La dimensione del file stesso è direttamente proporzionale alla dimensione dei k-mer (ad esempio: se k = 31 il numero di tuple che teoricamente si possono ottenere sono 4^31).

Anche Jellyfish sfrutta la presenza di più processori o thread per aumentare la velocità di

elaborazione delle operazioni.

Le dimensioni del file di output *database.jdb* sono:

$$Dimensione\ header\ (byte) = 80 + 16 * Key\_bits$$

$$Dimensione\ body\ (byte) = (Val\_len + Key\_len) * Key\_ct$$

Dove Key_len è il numero di byte minimo necessario a rappresentare il k-mer.

$$Key\_len = \left\lceil \frac{Key_{bits}}{8} \right\rceil$$

Si ottiene quindi la dimensione del file totale:

$$Dimensione\ file\ .jdb\ (byte) = Header + Body$$

La dimensione finale non può essere stabilita a priori solo tramite i parametri inseriti ma è influenzata dal numero di k-mers differenti presenti nei genomi di riferimento.

2.2.2 Step 2 - Riduzione dimensione k-mers set

Il secondo step è l'unico non obbligatorio durante la creazione della base di dati compressa. Se indicato dall'utente, Kraken è in grado di ridurre, al valore impostato, la dimensione del database finale. Questa operazione è necessaria nel caso la base di dati venga poi utilizzata per effettuare operazioni di classificazione su calcolatori dotati di capacità di memoria primaria o secondaria più limitate. 'MiniKraken' è il database di Kraken completo, con lunghezza dei k-mers uguale a 31 e ridotto di dimensione fino a 4 GB (fornito dagli autori per poter essere utilizzato anche su calcolatori domestici più limitati a livello hardware).

La riduzione si ottiene tramite l'eliminazione casuale di tuple dal database. Vengono rimossi i primi n-1 record da un blocco formato da 'n' record. In base alla quantità fornita dall'utente, l'algoritmo calcola il valore di n come il minimo intero necessario al raggiungimento della dimensione stabilita. L'operazione causerà molto probabilmente una perdita di *Recall* nella classificazione finale.

2.2.3 Step 3 - Indicizzazione del database

Il terzo step è il cuore della velocità di Kraken. L'algoritmo rende particolarmente efficiente l'operazione di classificazione indicizzando l'intera base di dati compressa velocizzando la ricerca dei k-mers contenuti. L'indicizzazione del database permette di ridurre l'insieme dei record da leggere per completare una singola ricerca.

Inizialmente vengono creati diversi blocchi, ognuno dei quali raggruppa tutti i k-mers che condividono una specifica sottostringa in comune chiamata *minimizer*. Nella prima versione di Kraken il minimizer fu implementato come: la sottostringa, di lunghezza prefissata, di un k-mer lessicograficamente più alta. Quando i k-mers raggiungono dimensioni notevoli (k = 31), ed è presente una forte differenza di lunghezza tra stringa e sottostringa, i minimizer restituiti tendono a concentrarsi tra i valori lessicograficamente più alti (ad esempio sottostringhe che iniziano per 'A'). Per avere un'indicizzazione il più efficiente possibile, l'ideale sarebbe riuscire a distribuire in modo equo tutti i k-mers nei vari gruppi. Nella seconda versione, per redistribuire più uniformemente i k-mers nei vari minimizer, sono state introdotte delle maschere apposite, applicate alle stringhe rappresentanti i k-mers prima del calcolo delle sottotringhe tra le quali poi verrà scelto il minimizer.

Ottenuti tutti i gruppi si procede al riordinamento alfabetico dei k-mers contenuti. Questa organizzazione permette, in caso di ricerca di un k-mer nel database, di:

1. Calcolare il minimizer del k-mer da cercare;
2. Individuare, tramite il file di indice, quale gruppo di k-mers condivide lo stesso minimizer. (Se il k-mer è presente nel database allora deve per forza trovarsi in quel gruppo);
3. Effettuare una ricerca dicotomica della stringa cercata sfruttando l'ordinamento lessicografico dei k-mers presenti nel gruppo.

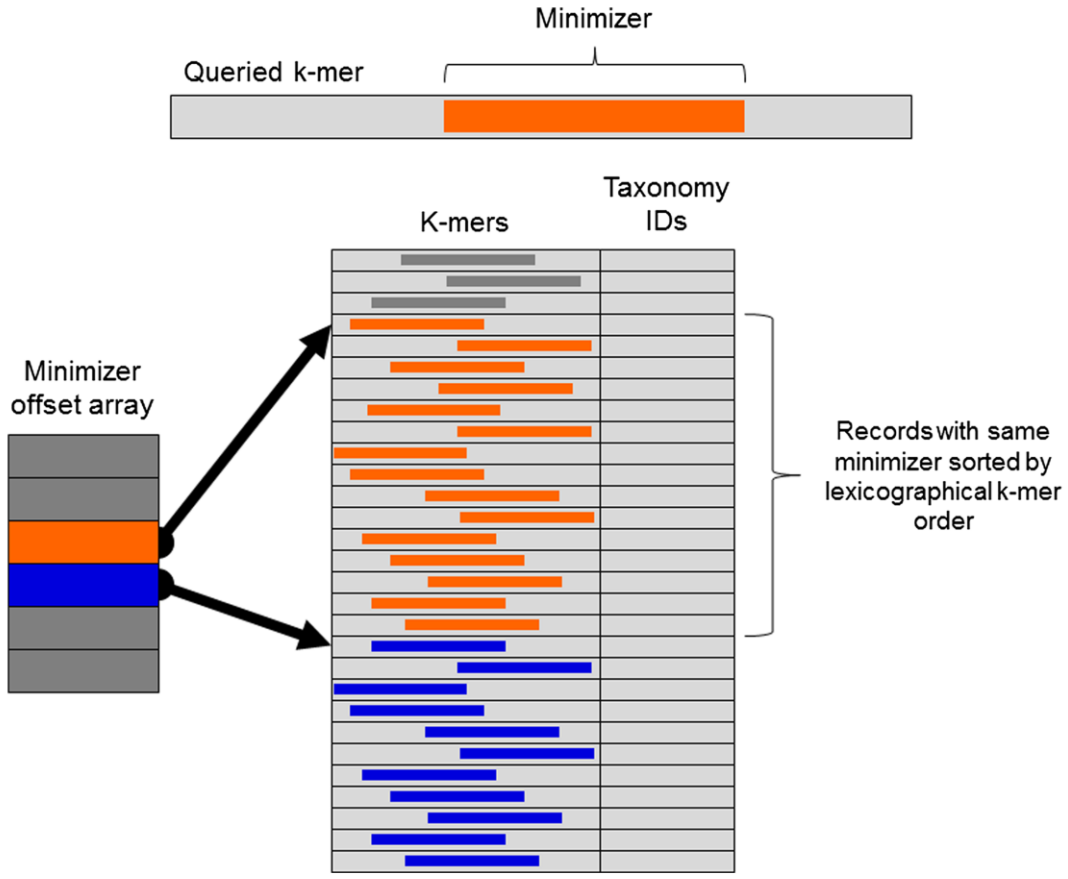

Figura 2.2 Schema di funzionamento del Minimzer.

La lunghezza del minimizer può essere definito dall'utente (dimensione standard = 15) e stabilisce la quantità di possibili gruppi che vengono creati. Con un parametro 'n' si posso creare fino a 4^n minimizer distinti. Questo influisce sia sul numero di gruppi presenti, sia sulla dimensione del file di indice.

Il file di indice viene denominato *database.idx"*.

L'Header del file contiene i seguenti campi:

- Stringa di controllo, necessaria per verificare che il file da leggere sia nel formato corretto. Nel file di indice contiene la stringa 'KRAKIX2' introdotta nella seconda versione del classificatore.
- Nt, Lunghezza dei minimizer

La dimensione del file *database.idx* è determinata dal parametro settato di lunghezza del minimizer.

$$Dimensione\ file\ .idx\ (byte) = 8 + 8 * (4^{Nt} + 1)$$

L'indicizzazione della base di dati è una fase che richiede una quantità notevole di memoria primaria necessaria al mantenimento ed all'elaborazione delle strutture dati necessarie all'intero processo.

La base di dati indicizzata viene memorizzata in un file denominato *"database.kdb"*, la cui dimensione è la stessa del file *"database.jdb"*.

2.2.4 Step 4 - Creazione mappa GI – SeqID

Il quarto passaggio crea una mappa di associazione identificativo genomico (GI) e identificativo della sequenza (SeqID). Il quarto e quinto step servono per associare ad ogni sequenza del database di riferimento un identificativo tassonomico (TaxID). Informazioni di associazione si ottengono mediante l'analisi del file FASTA in cui è memorizzata la sequenza.

In bioinformatica FASTA è un formato di file di testo utilizzato per rappresentare sequenze di nucleotidi o peptidi dove nucleotidi o aminoacidi sono rappresentati usando singoli caratteri ASCII. Il formato permette di nominare sequenze e aggiungere commenti. Originariamente è nato come il formato utilizzato dal software FASTA che poi è diventato uno standard, per la sua semplicità di lettura e scrittura. Essendo un file di testo, lettura e modifica vengono rese possibili tramite un semplice parser o un editor di testo.

Una sequenza nel formato FASTA comincia con una singola riga di descrizione seguita da righe contenenti i dati genomici della sequenza. La prima riga di descrizione, chiamata 'defline' si distingue dalle altre iniziando sempre con un simbolo 'maggiore di' (">"). Le sequenze di aminoacidi o nucleotidi devono essere rappresentate secondo lo standard IUB/IUPAC. È possibile includere più sequenze genomiche appartenenti a specie diverse all'interno dello stesso file e questo formato prende il nome di Multi-FASTA.

La riga di descrizione contiene informazioni riguardanti la sequenza genomica, tra i quali l'identificativo genomico. La mappa quindi associa GI e SeqID:

- GI, numero progressivo assegnato ad ogni record processato e memorizzato nei database dell'NCBI. Il sistema di identificazione ne permette l'accesso tramite i sistemi di ricerca forniti ed un controllo dei cambiamenti che questo subisce nel tempo.
- SeqID, descrittore di una sequenza genomica memorizzata in un file FASTA contente i vari identificatori, tra cui il GI.

La mappa risultante viene memorizzata in un file denominato *gi2seqid.map*.

2.2.5 Step 5 - Creazione mappa SeqID – TaxID

Il quinto passaggio completa la creazione della mappa di associazione SeqID↔TaxID, necessaria per l'assegnamento dell'identificativo tassonomico corretto ad ogni sequenza presente nel database di riferimento.

NCBI fornisce il file denominato *gi_taxid_nucl.dmp* contente tutte le associazioni tra un GI ed il suo corretto TaxID. La dimensione del file è di circa 8,7 GB. Il TaxID è un valore numerico assegnato dall'NCBI per identificare un preciso nodo dell'albero tassonomico generale.

Sfruttando la mappa creata in precedenza ed il file *gi_taxid_nucl.dmp* si è in grado di creare un'ulteriore mappa che associa l'identificativo della sequenza con il corretto TaxID. Questo permette di ottenere una lista ordinata in maniera lessicografica di tutte le *defline,* delle sequenze contenute nel database di riferimento, con i relativi identificatori tassonomici utile per effettuare efficientemente la ricerca durante la sesta fase di assegnamento dei Lower Common Ancestors.

La mappa risultante viene memorizzata in un file denominato *seqid2taxid.map*.

2.2.6 Step 6 - Assegnazione Lower Common Ancestors (LCAs)

Il sesto e ultimo step completa la costruzione della base di dati compressa. Viene applicata la reale logica di risoluzione di Kraken al problema della classificazione metagenomica associando un TaxID ad ogni k-mer presente nel database. Per creare le strutture dati indispensabili all'esecuzione dell'algoritmo è necessario caricare in memoria principale:

- database.kdb, base di dati compressa indicizzata allo step 3
- database.idx, file di indice ottenuto allo step 3
- seqid2taxid.map, mappa SeqID - TaxID ottenuta allo step 5
- nodes.dmp, file fornito dall'NCBI necessario per ricostruire l'intero albero tassonomico con le relazioni padre-figlio di tutti i taxID esistenti

L'algoritmo sfrutta il file già esistente *database.kdb*, di k-mers indicizzati, per memorizzare i valori dei taxID nel campo delle tuple precedentemente destinato al conteggio delle occorrenze dei k-mers calcolato da Jellyfish della dimensione di 4 byte.

Caricate in memoria ed inizializzate tutte le strutture dati necessarie, le sequenze genomiche contenute nelle librerie di riferimento vengono processate in maniera seriale. Per ogni sequenza ne viene ricavato l'identificativo tassonomico grazie alla mappa *seqid2taxid.map* ottenuta al quinto step e ne vengono estratti tutti i k-mer contenuti.

La base di dati indicizzata è inizializzata con un valore di 0 nel campo destinato a contenere il TaxID, permettendo così di controllare se il k-mer è già stato elaborato precedentemente.

Il procedimento per il settaggio del TaxID di ogni k-mer segue la seguente logica:

1. Si cerca il k-mer nel database compresso e indicizzato.
2. Se il valore del TaxID della tupla associato al k-mer è uguale a quello di inizializzazione allora:

- Viene settato con il TaxID della sequenza da cui è estratto il k-mer

Altrimenti:

- viene settato come valore il Lower Common Ancestor nell'albero tassonomico tra il TaxID già presente nel campo analizzato e quello della sequenza da cui è estratto il k-mer

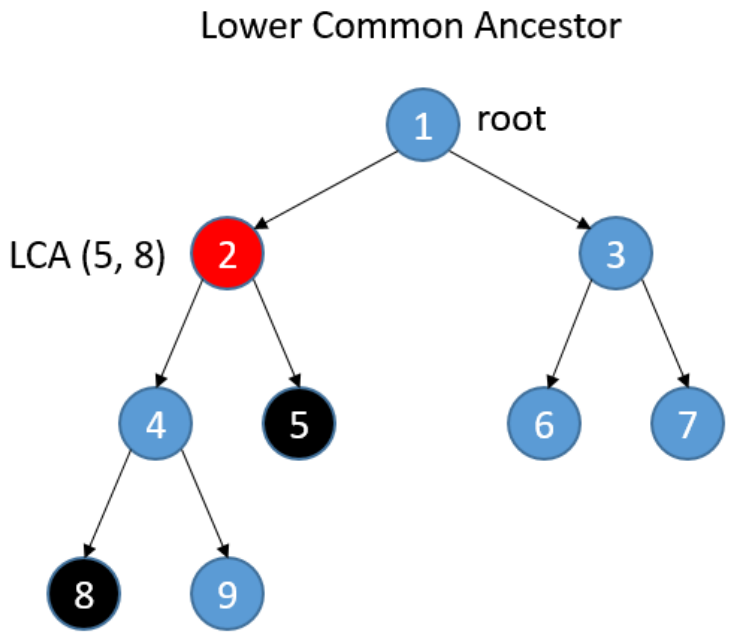

Figura 2.3 Esempio calcolo Lower Common Ancestor.

La lettura dei vari k-mers appartenenti alla stessa sequenza viene effettuata in parallelo sfruttando diversi processori o thread dell'elaboratore, se disponibili, per velocizzare l'intero processo non presentando condizione di concorrenza e quindi evitando di ricorrere a regioni di mutua esclusione che potrebbero causare errori.

Il procedimento descritto si ripete per ogni sequenza presente nella libreria.

Completato il procedimento di assegnamento per l'intera libreria genomica di riferimento, ogni k-mer avrà associato un TaxID che rappresenterà il più basso rank tassonomico raggruppante tutti gli organismi nei cui genomi è stato individuato il k-mer stesso. La costruzione della base di dati compressa creata da Kraken è completata e può essere utilizzata per la classificazione.

2.3 Classificazione

Data una sequenza genomica in ingresso di origine sconosciuta, il processo di classificazione consiste nell'assegnare un'etichetta tassonomica ad essa, utilizzando le informazioni contenute nella base di dati compressa indicizzata creata in precedenza. I file richiesti per inizializzare le strutture dati necessarie al processo di classificazione sono:

- database.kdb, base di dati compressa ed indicizzata ottenuta dal processo di creazione del database di Kraken;
- database.idx, file contente gli indici necessari per effettuare ricerche nel database;
- nodes.dmp, file contente le informazioni riguardanti i nodi dell'albero tassonomico;
- file FASTA contenente le sequenze sconosciute da classificare.

Data la sequenza da classificare l'algoritmo estrae ogni k-mer contenuto in essa della stessa lunghezza di quelli memorizzati nel database compresso. Procede a cercarlo tramite l'ausilio del file di indice all'interno della base di dati e si possono verificare due situazioni:

1. Il k-mer è presente e quindi possiede un'etichetta tassonomica associata
2. Il k-mer non è presente e quindi sconosciuto

Viene mantenuta una mappa che associa ogni nodo dell'albero tassonomico completo con un peso. Il peso di un'etichetta tassonomica viene incrementato di una unità ogni volta che un k-mer con quel TaxID associato viene trovato nel database.

Completata l'analisi di tutti i k-mer ogni nodo dell'albero tassonomico avrà un peso pari al numero dei k-mers della sequenza ad esso associati. L'insieme di tutti i nodi con peso diverso da 0 (valore standard di inizializzazione) formerà un sottoalbero, chiamato *classification tree*, la cui radice è il nodo di livello più basso (nel caso di due nodi allo stesso livello si considera come radice il LCA di essi).

Riassumendo il procedimento: ogni k-mer viene associato al LCA degli organismi, presenti nel database, con genomi che lo contengono. Considerando il *classification tree*, e supponendo che abbia 'n' diverse foglie, sarà possibile creare 'n' diversi cammini dalla radice a ciascuna foglia, chiamati cammini 'Root-to-Leaf (RTF). Ad ogni cammino sarà possibile associare un punteggio dato dalla somma dei pesi dei nodi incontrati durante la

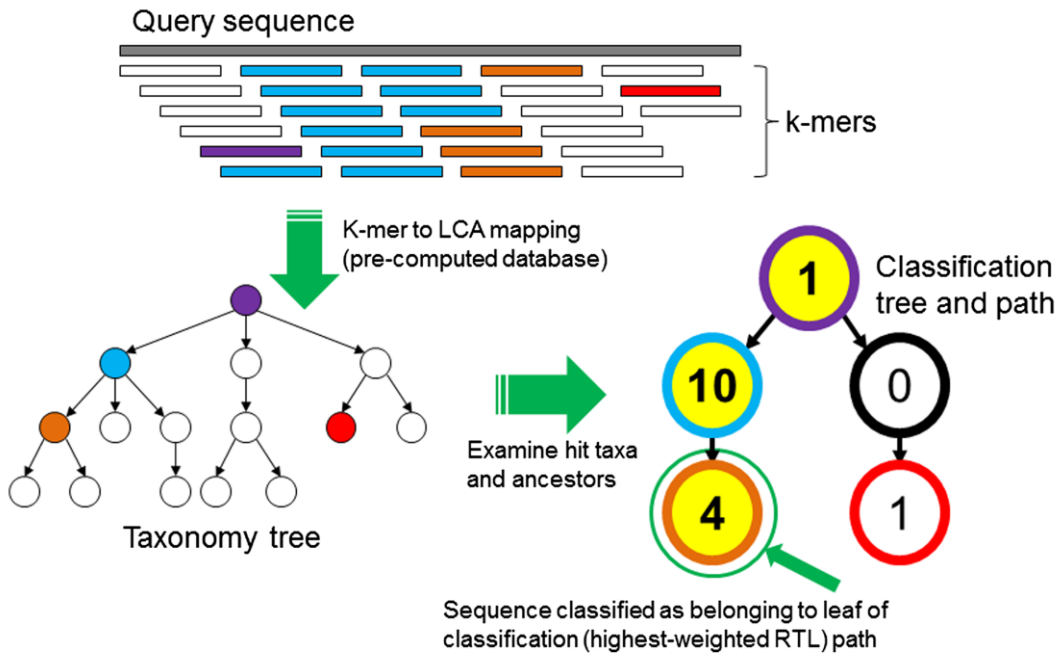

Figura 2.4 Schema di classificazione di Kraken.

discesa fino alla foglia. Alla sequenza verrà infine assegnata l'etichetta tassonomica appartenente alla foglia del cammino RTL con punteggio maggiore (nel caso di punteggi pari si assegna il LCA tra le due foglie). Se l'algoritmo non trova nessuna corrispondenza dei k-mer estratti con quelli contenuti nel database la sequenza viene lascia 'non classificata'.

La capacità con cui l'algoritmo è in grado di classificare una sequenza è fortemente determinata dalla quantità e dalla qualità dei k-mers contenuti all'interno della base di dati.

Capitolo 3 Metodi per la scoperta di k-mers discriminativi

Questo capitolo è dedicato alla descrizione del concetto di discriminatività adottato, alla proposta di possibili estensioni dell'algoritmo originale di Kraken e alla descrizione dettagliata del loro funzionamento. Le nuove varianti intraprenderanno alcune scelte, di creazione del database compresso e di classificazione, basandosi sulla discriminatività dei k-mers delle sequenze di riferimento.

3.1 Discriminatività

La discriminatività di un k-mer definisce quanto è rappresentativo per un'etichetta tassonomica il k-mer ad essa associato. L'algoritmo utilizzato da Kraken crea il database dei LCAs senza considerare questo parametro. Durante il sesto step (assegnazione dei LCAs) un k-mer viene spostato a livelli sempre meno profondi dell'albero tassonomico man mano che viene trovato su sequenze appartenenti a specie differenti (foglie dell'albero) tassonomicamente distanti. Più un k-mer è associato a nodi meno profondi dell'albero, e quindi rappresentativo di gruppi sempre più ampi di organismi, meno sarà determinante nella classificazione finale a rank tassonomici bassi.

Vengono di seguito proposte delle varianti dell'algoritmo originale che considerano il parametro di discriminatività dei k-mers durante la creazione della base di dati compressa oppure durante la classificazione. Ogni variante proposta è stata progettata come

un'estensione di Kraken, dando la possibilità all'utente di utilizzare il classificatore originale, o nelle varie versioni, definendo alcuni parametri all'avvio della creazione della base di dati personalizzata.

Le modalità proposte differiscono in base alla fase in cui vanno ad operare:

Fase di costruzione del database:

- Absolutely discriminative
- Score discriminative

Fase di classificazione:

- Score weighted classification

3.2 Absolutely discriminative

La prima soluzione proposta è basata sulla "discriminazione assoluta". Un k-mer viene definito assolutamente discriminativo se appare unicamente in un solo nodo del rank tassonomico stabilito. Questa variante prevede una modifica del funzionamento del sesto step, agendo sulla creazione della base di dati, eliminando i k-mers che non soddisfano la discriminatività assoluta.

Lo scopo è quello di:

- Ridurre le dimensioni del database originale
- Studiare le variazioni dei parametri statistici del classificatore con questo approccio al problema

3.2.1 Nuove funzioni e strutture dati

Per il corretto funzionamento dell'algoritmo modificato sono state progettate e implementate funzioni e strutture dati aggiuntive.

È stata introdotta la seguente struttura dati nel file *set_lcas.cpp*:

- map <uint32_t, uint8_t> Rank_map

  Implementa una mappa contenente le coppie (etichetta tassonomica, intero a 32

bit senza segno rappresentante il rank tassonomico associato).

Permette, tramite una sua interrogazione, di ottenere il rank di un nodo cercato dell'albero tassonomico.

Sono state implementate le seguenti funzioni nel file *krakenutil.cpp*:

- uint8_t **str_to_rank** (string str_rank)
- map <uint32_t, uint8_t> **build_rank_map** (string filename)

La funzione **str_to_rank** restituisce il valore intero associato ad una stringa rappresentante un rank tassonomico contenuto nel file nodes.dmp fornito dall'NCBI (ad esempio: 'species' -> 7).

La funzione **Build_rank_map** crea la Rank_map utilizzando il file nodes.dmp contente le informazioni per la ricostruzione dell'albero tassonomico. Effettuando un parsing del file, estrae informazioni riguardanti ogni nodo dell'albero tassonomico: taxID e rank (in formato stringa), traduce il rank nel formato numerico (tramite la funzione str_to_rank) e crea la Rank_map inserendo tutte le coppie, una per ogni nodo dell'albero tassonomico.

Sono state implementate le seguenti nuove funzioni nel file *set_lcas.cpp*:

- uint32_t **at_rank** (uint32_t taxid, uint8_t rank)
- void **set_discriminative** (uint32_t taxid, string &seq, size_t start, size_t finish)

Dati in ingresso: TaxID e rank, la funzione **at_rank**, restituisce il TaxID del nodo antenato al rank stabilito del nodo dato in input. Se il nodo è già al rank stabilito il TaxID non subisce variazioni. Questa funzione è necessaria in quanto l'NCBI utilizza, oltre ai classici rank tassonomici, dei sotto-rank per motivi di organizzazione interna dei dati. Se l'algoritmo venisse applicato considerando solo il TaxID della sequenza attualmente analizzata si potrebbero incorrere in errori. L'etichetta tassonomica di un nodo rappresentante una specie è diversa da una rappresentante una sua sottospecie. K-mers condivisi tra due sequenze genomiche (specie e sottospecie) verrebbero eliminati dal database, in quanto considerati non 'assolutamente discriminativi', quando invece dovrebbero essere considerati come appartenenti alla stessa specie.

La funzione **at_rank** serve per uniformare le etichette tassonomiche allo stesso livello al momento del confronto e quindi eliminare solo i k-mers realmente condivisi allo stesso rank (stabilito dall'utente) da nodi diversi.

La funzione **set_discriminative** sostituisce la funzione **set_lcas** di Kraken. Effettua i confronti descritti in precedenza e, nel caso vengano trovati k-mer non 'assolutamente discriminativi' viene importato il loro TaxID ad un valore standard di 1.

Finita l'analisi di tutte le sequenze genomiche ogni k-mer nella base di dati compressa potrà trovarsi in due situazioni differenti:

- K-mer assolutamente discriminativo, con associato il TaxID del nodo appartenente
- K-mer non assolutamente discriminativo e con associato un TaxID standard di 1

Se è presente almeno una tupla da eliminare viene creato un file denominato "*del.cnt*" contente il loro conteggio. Si procede quindi all'eliminazione dei k-mers non assolutamente discriminativi dalla base di dati tramite un settimo step implementato appositamente che è descritto alla fine del capitolo.

3.2.2 Utilizzo

I parametri da inserire, durante la creazione di un database personalizzato di Kraken, per la creazione della base di dati assolutamente discriminativa sono i seguenti:

- --abs-discriminative
- --cl-rank NUM

Il parametro cl-rank permette di definire, tramite NUM, il rank tassonomico a cui uniformare tutte le sequenze prima di effettuare la discriminazione assoluta dei k-mers.

Non è obbligatorio e, se non specificato, viene posto ad un valore standard di 8.

Le associazioni rank-NUM utilizzate dall'algoritmo si possono trovare alla tabella 2.1.

Super Regno	0
Regno	1
Phylum	2
Classe	3
Ordine	4
Famiglia	5
Genere	**6**
Specie	**7**
Altri	8

Tabella 3.1 Tabella associazioni rank-NUM utilizzate.

3.3 Score discriminative

La seconda soluzione proposta valuta la discriminatività di un k-mer basandosi su uno score. Lo score assegnato ad ogni k-mer, riguardante la sua discriminatività, non esiste in letteratura. È stato appositamente pensato ed implementato per descrivere, tramite un valore statistico, il concetto di 'grado di rappresentanza' di un k-mer per un organismo, o un gruppo di essi. Viene modificato il sesto step. La nuova variante dell'algoritmo sfrutta cicli già esistenti per ottenere informazioni aggiuntive, necessarie al calcolo finale degli score.

Lo schema generale di funzionamento è il seguente:

1. L'assegnamento dei LCAs avviene come progettato nell'algoritmo originale di Kraken
2. Viene calcolato lo score di ogni k-mer presente nella base di dati compressa
3. Ad ogni k-mer, con uno score inferiore a quello stabilito dall'utente, viene posto il valore a 1 del TaxID associato. Saranno i k-mers destinati ad essere eliminati perché non abbastanza discriminativi secondo i parametri stabiliti

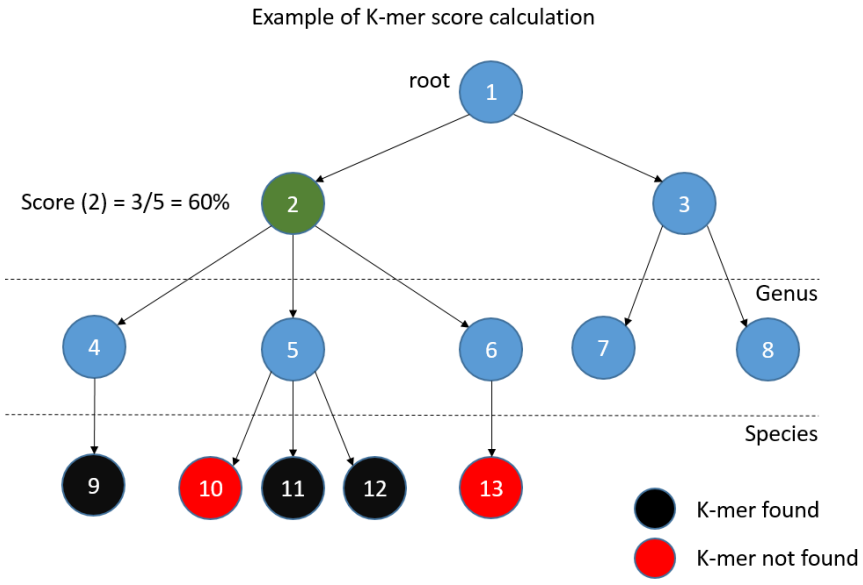

Figura 3.1 Esempio di calcolo dello Score di un k-mer.

4. Se presenti k-mer da eliminare, viene generato il file del.cnt, contenente la loro quantità in termini di numero, e viene richiamato il settimo step che provvede a creare la nuova base di dati ridotta tramite la loro eliminazione

Lo score di un k-mer è definito come:

$$\text{Kmer\_score} = \frac{A}{B}$$

Dove:

- A: numero di nodi foglia (specie) del sottoalbero tassonomico, con radice il TaxID associato, che presentano il k-mer in almeno una delle loro sequenze nel database riferimento
- B: numero di nodi foglia totali (specie) del sottoalbero tassonomico con radice il TaxID associato al k-mer

Nel calcolo di entrambi i valori viene considerato l'albero tassonomico potato, i cui nodi foglia sono rappresentati da solo gli organismi presenti nei database di riferimento.

Lo score definisce la percentuale di quante specie presentano il k-mer nelle loro sequenze genomiche rispetto a tutte quelle che presentano come antenato il TaxID associato.

Ad esempio:

- Dato un k-mer (M) con associato un taxID (T) e uno score di 100 significa che il 100% degli organismi presenti nel database di riferimento che sono discendenti di T nell'albero tassonomico presentano M in almeno una delle loro sequenze genomiche. Il k-mer viene quindi considerato al 100% rappresentativo del nodo tassonomico associato.

Lo scopo è quello di:

- Ridurre la dimensione della base di dati compressa eliminando le informazioni poco importanti
- Analizzare come varia la qualità della classificazione con l'introduzione di questo parametro di discriminazione dei k-mers

3.3.1 Nuove funzioni e strutture dati

Per il corretto funzionamento dell'algoritmo modificato sono state progettate e implementate funzioni e strutture dati aggiuntive.

Sono state introdotte le seguenti strutture dati nel file *set_lcas.cpp*:

- set <uint64_t> **kmer_check_set**;
- map <uint32_t, uint32_t> **Pruned_parent_map**;
- multimap <uint32_t, uint32_t> **Children_multimap**;
- uint32_t ***kmer_weight**;
- uint8_t ***kmer_perc**;

Il **kmer_check_set** è un insieme utilizzato per il controllo ed il conteggio dei k-mers presenti in una specie. Gestisce i valori contenuti nell'array **kmer_weight**, controllando che avvengano aumenti nei conteggi se e solo se si stanno considerando specie diverse.

La **Pruned_parent_map** è la mappa che rappresenta l'albero tassonomico potato costruito considerando solamente i nodi foglia appartenenti al database di riferimento. Durante l'intera esecuzione dell'algoritmo la struttura dati creata viene riutilizzata (essendo poi non ulteriormente sfruttata) per contenere i valori che andranno a comporre

il denominatore della frazione (**B**) che costituisce il calcolo del k-mer score.

La **Children_multimap** è la multimappa (che al contrario della struttura dati "mappa" consente la presenza di chiavi con lo stesso valore) utilizzata per trovare i figli di un nodo interno dell'albero tassonomico.

Kmer_weight è l'array di interi non segnati a 32 bit contente i valori che costituiranno il numeratore (**A**) nel calcolo del k-mer score. L'ordinamento degli elementi appartenenti all'array segue quello della base di dati ordinata ed indicizzata. L'elemento i-esimo dell'array corrisponde al k-mer nella posizione i-esima del database. Questo evita di dover mantenere informazioni doppie e quindi la necessità di ulteriore memoria primaria durante l'esecuzione.

Kmer_perc, array contente gli score finali.

Sono state implementate le seguenti funzioni nel file *krakenutil.cpp*:

- std::multimap<uint32_t,uint32_t> **build_children_multimap** (std::map<uint32_t, uint32_t> pruned_parent_map)

La funzione **build_children_multimap** crea la struttura dati multimappa **Children_multimap**. Le informazioni fornite dall'NCBI riguardante l'albero tassonomico permettono una sua navigazione solamente con un approccio di tipo bottom-up. Per ragioni di spazio di memorizzazione, conoscendo un nodo si può solamente sapere chi è il padre risalendo tutto l'albero tassonomico fino alla radice. La progettazione della modalità alternativa (score-discriminative) prevede invece la ricerca dei figli di un nodo interno. La multimappa permette la presenza di chiavi con lo stesso valore. La funzione memorizza le coppie nella multimappa come (nodo padre, noto figlio) consentendo quindi una sua successiva interrogazione per taxID del padre restituendo un iteratore contente tutti i figli.

Tutte le strutture dati descritte (set, map, multimap) appartengono alla libreria standard di C++ e quindi ottimizzate per tutte le operazioni di inserimento, cancellazione e ricerca.

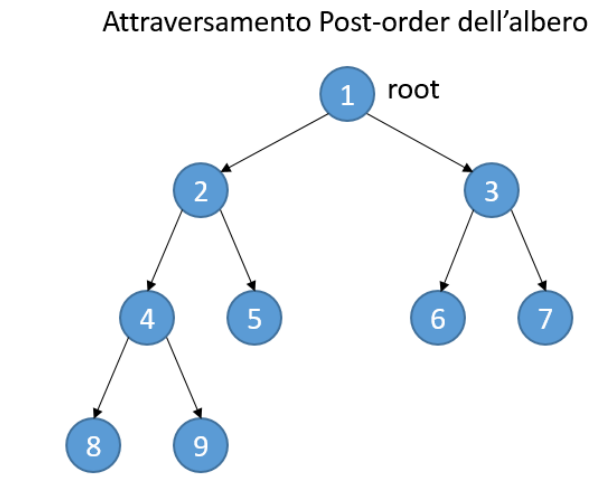

Figura 3.2 Esempio attraversamento Post Order.

Sono state implementate le seguenti funzioni nel file *set_lcas.cpp*:

- uint32_t **count_children** (uint32_t taxid)
- void **calculate_percentage**();

La funzione **count_children,** dato in ingresso un indice tassonomico rappresentate un nodo dell'albero potato, fornisce la quantità di nodi foglia presenti ad esso associato. L'algoritmo implementato sfrutta l'attraversamento in **Post-order** dell'albero potato. L'attraversamento in Post-order di un albero permette di processare prima tutti i figli e poi il padre riuscendo così a sfruttare i conteggi già eseguiti. Il numero di nodi foglia associato ad un nodo interno è uguale alla somma dei nodi foglia associati a figli. Se un figlio è un nodo foglia allora viene considerato con un conteggio pari a 1. L'algoritmo è in grado di fornire il risultato per ogni nodo dell'albero. Dando in input alla funzione il taxID della radice dell'albero tassonomico (1) si riesce ad ottenere in conteggio di ogni nodo durante l'attraversamento in Post-order. I valori ottenuti per ogni nodo andranno a popolare la mappa **Pruned_parent_map** (coppie [nodo – conteggio] al posto di [nodo figlio – nodo padre]) che verrà ora utilizzata per rappresentare il denominatore della frazione (**B**) durante il calcolo del k-mer score. Viene riutilizzata la **Pruned_parent_map** perché già

presente in memoria principale con una coppia per ogni nodo interno e non più sfruttata dall'algoritmo.

La funzione calculate_percentage calcola gli score finali e infine con essi popola l'array kmer_perc.

Per il calcolo dei k-mer score si utilizzano le strutture dati già create e popolate:

- database.kdb
- kmer_weight
- Pruned_parent_map

Durante l'esecuzione dell'algoritmo il valore immesso dall'utente come soglia e la percentuale calcolata per ogni k-mer vengono normalizzati con valori appartenenti all'intervallo di interi [0,255]. Questo permette di sfruttare tutti gli 8 bit utilizzati per la memorizzazione e consente una maggiore precisione rispetto all'intervallo di interi [0,100].

Se è presente almeno una tupla (k-mer, taxID) da eliminare viene creato un file denominato del.cnt contente il loro conteggio. Si procede quindi all'eliminazione dei k-mers non discriminativi, in base allo score fornito dall'utente, dalla base di dati tramite un settimo step implementato appositamente e descritto in seguito nel capitolo.

3.3.2 Utilizzo

Il parametro da inserire, durante la creazione di un database personalizzato di Kraken, per la creazione della base di dati discriminativa in base agli score è il seguente:

- --scr-discriminative NUM

Il valore NUM inserito dall'utente rappresenta la soglia minima che lo score di un k-mer deve raggiungere per non essere eliminato dal database. Sono accettati numeri interi o con la virgola appartenenti all'intervallo [0,100].

Analizzando i casi limite:

- NUM = 0 non eliminerà alcun k-mer e quindi il database finale sarà esattamente uguale a quello creato dall'algoritmo originale di Kraken.
- NUM = 100 non si comporta come una discriminazione assoluta. Potrebbe presentare qualche k-mer che ha ottenuto uno score uguale a 100 e che sarebbe stato considerato come assolutamente discriminativo.

3.4 Score weighted classification

La terza variante presentata valuta la discriminazione di un k-mers basandosi su uno score. Al contrario delle due modalità precedenti non modifica la fase di creazione in quanto il database dei LCAs risulta lo stesso di quello creato dall'algoritmo originale di Kraken. Questa variante modifica la fase di classificazione pesando diversamente i nodi del classification tree su cui viene calcolato il cammino Root-to-Leaf (RTL) con l'obiettivo di scegliere il taxID vincitore. Ogniqualvolta viene trovato il k-mer estratto dalla sequenza da classificare nel database compresso, l'algoritmo originale aumenta di un'unita il peso del taxID corrispondente. Questa modalità aumenta il peso relativo ad un nodo in base alla discriminatività del k-mer di cui è stata trovata la corrispondenza nella base di dati. Lo scopo è quello di rendere significativa la presenza di k-mers più discriminativi rispetto ad altri con un score minore. Lo score assegnato ad ogni k-mers è lo stesso utilizzato dalla modalità "score discriminative".

Il funzionamento generale è il seguente:

1. Durante la sesta fase vengono inizializzate tutte le strutture dati descritte nella modalità "score discriminative" necessarie al calcolo degli score dei k-mers memorizzati nella base di dati compressa
2. Le informazioni relative agli score vengono memorizzati in un file esterno denominato *database.scr*
3. Durante la fase di classificazione viene assegnato come unità di peso lo score corrispondente al taxID di ogni k-mer che viene trovato nel database al posto del valore standard 1

Lo scopo è quello di:

- Studiare come varia la qualità della classificazione aggiungendo informazioni sulla discriminatività di ciascun k-mer influenzando il peso del cammino RTL durante la classificazione

3.4.1 Nuove funzioni e strutture dati

La modalità prevede la condivisione di tutte le strutture dati descritte nella "score discriminative". Contrariamente alla precedente non effettua alcuna eliminazione dei k-mers ma si limita a memorizzare i valori ottenuti nel calcolo degli score.

È stata implementata la seguente funzione al file database.cpp:

- void KrakenDB::**make_score** (string score_filename, uint64_t ct, uint8_t *kmer_score)

Dati in ingresso il nome del file di score di output, il conteggio totale dei k-mers presenti e l'array contente i valori degli score calcolati (nel nostro caso l'array **kmer_perc**), la funzione crea il file di output contente gli score dei k-mers.

Le informazioni contenute nell'header sono:

- **Stringa di controllo**, necessaria per verificare che il file da leggere sia nel formato corretto. Nel file di score contiene la stringa 'KRAKSCR';
- **Key_ct**, quantità di k-mer distinti nel database.

La dimensione del file score è dipendente dal numero di k-mer distinti presenti nel file database.kdb

$$Dimensione\ file\ .scr\ (byte) = 15 + Key\_ct$$

3.4.2 Utilizzo

Il parametro da inserire, durante la creazione di un database personalizzato di Kraken, per la creazione del file di output contente gli score discriminativi per ogni k-mer contenuto nella base di dati compressa è il seguente:

- --scr-out-file

Per avviare la classificazione pesata tramite il file di score si inserisce il seguente parametro:

- --s Score_filename

Dove Score_filename è il file creato precedentemente.

I parametri inseriti per le tre modalità sono mutuamente esclusivi quindi è possibile creare il database di Kraken con solo una variante per volta.

3.5 Step 7 - Pulizia database compresso

Il settimo passaggio è utilizzato dalle versioni alternative di Kraken proposte che prevedono delle possibili rimozioni di tuple dal database. L'algoritmo provvede a copiare in una nuova base di dati compressa tutte le tuple (derivanti dal sesto step) aventi un TaxID associato diverso da 1 (valore standard per i k-mers considerati non sufficientemente discriminativi dalla modalità utilizzata). La dimensione della nuova base di dati ridotta è influenzata dalla quantità di tuple eliminate.

L'eliminazione di tuple dal database necessita della ricreazione del file di indice. Non è necessario un riordinamento dei k-mers nella base di dati (come durante il terzo step) in quanto la sola rimozione va a spostare verso l'alto la posizione delle tuple senza cambiarne l'ordine complessivo.

Il settimo step viene eseguito solamente se è presente il file *del.cnt* (generato dal sesto step in caso di k-mers da rimuovere nel database).

Lo flusso generale di esecuzione è il seguente:

1. Viene letto il conteggio dei k-mers da eliminare dal file del.cnt
2. Crea un nuovo file che conterrà il database ridotto
3. Copia l'intero header del database originale nel ridotto modificando il campo key_cnt con il conteggio di k-mers rimanenti dopo la pulizia
4. Serialmente vengono lette le tuple nel database originale e copiate in quello ridotto solo nel caso ai cui k-mers sono associati taxID diversi da 1
5. Ricrea l'indice relativo al database compresso

3.5.1 Utilizzo

La pulizia del database compresso si avvia autonomamente, durante la costruzione di un database di kraken personalizzato, utilizzando le modalità che la prevedono e nel caso ci siano tuple da eliminare. Tuttavia è possibile avviarla manualmente tramite l'esecuzione del file *clean_reduced_db.cpp*.

I parametri accettati in ingresso sono:

- -d filename, database indicizzato compresso da ridurre, con i k-mers da eliminare con un taxID impostato a 1
- -i filename, file di indice relativo al database da ridurre
- -o filename, database ridotto ottenuto in output
- -p filename, file di indice ricalcolato relativo al database ridotto
- -l filename, file contente il conteggio dei k-mers da eliminare

Tutti i parametri sono obbligatori.

Capitolo 4 Test

Questo capitolo descrive i test effettuati con le nuove modalità implementate, confrontando i risultati ottenuti con l'algoritmo originale. Vengono analizzati i parametri statistici derivanti dalle classificazioni effettuate su dataset reali e artificiali al fine di valutare un miglioramento prestazionale del classificatore originale tramite le nuove tecniche proposte.

4.1 Dataset

Sono stati utilizzati diversi dataset metagenomici per valutare l'accuratezza delle nuove varianti di Kraken su comunità microbiotiche simulate. Alcuni di questi dataset sono gli stessi utilizzati dai creatori di Kraken nell'articolo di presentazione dell'algoritmo, creati partendo da dati ottenuti da sequenziamenti reali (HiSeq, MiSeq e simBA5). La differenza tra dataset reali e artificiali è la sicurezza riguardante le etichette tassonomiche assegnate come rappresentative della realtà ad ogni read, utilizzate poi per determinare se la classificazione ha avuto esito positivo per ognuna di esse. Questo aspetto è essenziale in quanto influenza fortemente i parametri di valutazione statistica calcolati per ogni variante del classificatore. Durante la fase di test viene utilizzato un dataset reale Paired-end chiamato SRR1804065. La determinazione delle etichette tassonomiche considerate come 'vere' sono state ottenute mediante la classificazione con BLAST.

Tipo	Dataset	# Read	# Read Sconosciute	% Read Sconosciute
Single-end	HiSeq_accuracy	10.000	1000	10%
	MiSeq_accuracy	10.000	1000	10%
	simBA5_accuracy	10.000	394	3,94%
Paired-end	MixKraken_1	1.000.000	52.623	5,26%
	MixKraken_2	1.000.000	64.910	6,49%
	SRR1804065	1.053.741	71	0,01%

Tabella 4.1 Statistiche dei dataset utilizzati in base al numero di read contenute.

Tipo	Dataset	# Specie	# Specie Sconosciute	Lunghezza media Reads (bp)
Single-end	HiSeq_accuracy	10	1	92
	MiSeq_accuracy	10	1	156
	simBA5_accuracy	1.216	91	100
Paired-end	MixKraken_1	10	1	100
	MixKraken_2	10	1	100
	SRR1804065	775	8	100

Tabella 4.2 Statistiche dei dataset utilizzati in base al numero di specie contenute.

HiSeq e MiSeq sono dataset creati con 10 shotgun reads appartenenti a interi genomi di batteri. Il 10% delle reads di ognuno appartengono ad una specie di batterio differente.

Sono stati volutamente introdotti degli ostacoli al classificatore per studiarne il comportamento in situazioni difficili. Nel dataset HiSeq sono presenti 1000 reads da classificare appartenenti alla specie *Pelosinus fermentans,* non presente nelle librerie genomiche dell'NCBI utilizzate e quindi impossibile da classificare correttamente. Stesso discorso vale per il dataset MiSeq con la specie *Proteus vulgaris*.

SimBA5 possiede reads ricavate da insiemi completi di genomi appartenenti a specie di batteri e archeobatteri presenti in RefSeq. È creato con un alto tasso di errore al

fine di valutare l'accuratezza delle classificazioni con sequenze contenenti molti errori o aventi forti differenze dalla libreria genomica di riferimento utilizzata.

Per maggiori informazioni riguardante gli dataset utilizzati da Kraken si può fare riferimento all'articolo originale del classificatore.

MixKraken1 e MixKraken2 sono due dataset Paired-end creati con il dataset di short-reads illumia MiSeq, con due abundance profiles.

SRR1804065 è un dataset Paired-end reale contenente short-reads di ricavate dal sequenziamento di un tampone di feci effettuato tramite il sequenziatore Illumia.

4.2 Parametri statistici

L'accuratezza del classificatore si ottiene valutando i parametri statistici ricavati dai risultati dei test effettuati sui dataset.

Le possibili classificazioni delle reads ottenute sono:

1. Reads possibili da classificare e classificate correttamente (*possClassAndClassCorrect*)
2. Reads possibili da classificare e classificate sbagliate (*possClassAndClassUncorr*)
3. Reads possibili da classificare e non classificate (*possClassAndNotClass*)
4. Reads impossibili da classificare e classificate (*notPossAndClass*)
5. Reads impossibili da classificare e non classificate (*notPossAndNotClass*)

Vengono quindi generate le seguenti popolazioni:

- PopolazionePrecision = 1 + 2 + 4
- PopolazioneTotale = 1 + 2 + 3 + 4 + 5

Indice	Sigla	Composizione
True Positive	TP	1
False Positive	FP	2 + 4
False Negative	FN	3
True Negative	TN	5

Tabella 4.3 Legenda di composizione degli indici statistici utilizzati.

Si definiscono i parametri statistici utilizzati per valutare i test:

- La *precision* rappresenta la frazione di reads classificate correttamente tra quelle classificate. È la probabilità che data una sequenza classificata questa sia corretta.

$$Precision = \frac{TP}{TP + FP} = \frac{possClassAndClassCorrect}{PopolazionePrecision}$$

- La *recall* (o sensitivity) rappresenta la frazione di reads correttamente classificate considerando anche quelle lasciate non classificate. È la percentuale di read classificate correttamente tra tutte quelle date in input. È importante notare che la Recall utilizzata nel contesto della classificazione metagenomica non è il parametro standard utilizzato in letteratura con ad esempio la classificazione binaria che invece al denominatore considera solo la somma di *True Positive* e *False Negative*. La recall considerata avrà un valore più basso rispetto a quella standard.

$$Recall = \frac{TP}{SUM\ OF\ ALL} = \frac{possClassAndClassCorrect}{PopolazioneTotale}$$

- *F-Measure* è la combinazione, tramite media armonica, della *precision* e della *recall*.

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

- *Il coefficiente di correlazione di Pearson* misura l'indipendenza lineare tra due variabili aleatorie X e Y, restituendo un valore appartenente all'intervallo [-1, +1] dove 1 indica una correlazione positiva totale, 0 nessuna correlazione e -1 correlazione negativa totale. Viene utilizzata nel contesto metagenomico per valutare l'*abbondanza*. Il variare di questo parametro individua un incremento o decremento della correlazione tra i valori presenti nel testset e quelli restituiti dalla classificazione.

$$Correlazione\ di\ Pearson = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

- $cov(X,Y)$: *Covarianza* delle due variabili statistiche *X* e *Y*
- σ_X, σ_Y*:* Rispettivamente *Deviazione standard* di *X* e *Y*

  Riarrangiando la formula per il calcolo dell'indice di correlazione di Pearson si ottiene la seguente forma, comoda per essere eseguita da un algoritmo in un unico ciclo di lettura dei dati:

$$Correlazione\ di\ Pearson = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

- n: Numero di valori in ogni vettore
- x_i, y_i: Valori nei vettori *X* e *Y* alla *i-esima* posizione

4.3 Risultati

Vengono di seguito riportati i risultati derivanti dalle classificazioni effettuate con le nuove varianti proposte su dataset descritti in precedenza. Vengono considerati due livelli differenti di classificazione: *Specie* e *Genere*. Le classificazioni metagenomiche sono difficili da ottenere con un'ottima precisione a livello Specie e normalmente in letteratura vengono fatti i confronti tra classificatori usando i risultati al livello tassonomico *Genere*. Per completezza vengono riportati i risultati ottenuti in entrambi i livelli.

4.3.1 Tabelle

Sono stati utilizzati i seguenti parametri per la costruzione delle basi di dati compresse nella modalità standard dell'algoritmo e nelle varianti proposte:

- 15 = Lunghezza minimizer (*minimizer-len*)
- 31 = Lunghezza k-mer (*kmer-len*)

Il valore di lunghezza dei k-mer con k=31 è stato considerato il miglior compromesso in termini di prestazioni (Precision / Recall) ottenuto in via sperimentale dagli autori di Kraken.

Con i valori sopracitati si ottengono le seguenti dimensioni di file per il database completo standard:

File	Dim (Byte)	Dim (GB)
database.kdb	70.381.181.032	65,54
database.idx	8.589.934.608	8

Tabella 4.4 Dimensione dei file output.

È di seguito presente una legenda esplicativa delle voci delle tabelle che si trovano lungo il capitolo:

Legenda	
Standard	Algoritmo standard di Kraken
Abs_Sp	Absolutely Discriminative a livello Species
Abs_Ge	Absolutely Discriminative a livello Genere
Scr_N	Score Discriminative con parametro di valore N
Weighted	Score Weighted Classification
Dim (bytes)	Dimensione del file *database.kdb* in Byte
Dim (GB)	Dimensione del file *database.kdb* in Giga Byte
# K-mer	Numero di K-mers presenti nella base di dati compressa
# K-mer Eliminati	Numero di K-mers eliminati dalla base di dati compressa rispetto a quella creata dall'algoritmo originale di Kraken
% K-mer Eliminati	Percentuale di K-mers eliminati rispetto alla base di dati creata dall'algoritmo originale di Kraken

Tabella 4.5 Legenda valori rappresentati nei risultati dei test.

HiSeq SPECIE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,766667	0,850816	0,77021	0,766861	0,776736	0,778335	0,791912	0,810149	0,834391	0,76692
Recall	0,6049	0,6051	0,605	0,6049	0,605	0,605	0,6051	0,6051	0,6051	0,6051
F-measure	0,676244	0,707223	0,677681	0,676319	0,680196	0,680808	0,686016	0,69277	0,701484	0,676467
Pearson	0,818301	0,83464	0,813169	0,818306	0,817264	0,81698	0,823542	0,829758	0,834762	0,818227

MiSeq SPECIE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,783087	0,883507	0,799157	0,785347	0,792505	0,794013	0,80087	0,817588	0,852084	0,783212
Recall	0,626	0,6257	0,6259	0,626	0,626	0,626	0,6258	0,6257	0,6256	0,6261
F-measure	0,695787	0,732584	0,701996	0,696678	0,69948	0,700067	0,702593	0,708888	0,721485	0,695899
Pearson	0,865057	0,890823	0,866892	0,865116	0,866371	0,866435	0,867477	0,876101	0,888542	0,865677

simBA5 SPECIES

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,824274	0,963404	0,964908	0,82445	0,854982	0,861015	0,873636	0,894878	0,902972	0,824274
Recall	0,7688	0,7687	0,7699	0,7688	0,7688	0,7688	0,7688	0,7687	0,7687	0,7688
F-measure	0,795571	0,855109	0,856444	0,795653	0,809604	0,812299	0,817872	0,827004	0,830443	0,795571
Pearson	0,611984	0,868702	0,717452	0,612034	0,725108	0,728609	0,744991	0,766308	0,771684	0,612664

Tabella 4.6 Risultati statistici a livello SPECIE dei dataset: HiSeq, MiSeq, simBA5.

MixKraken_1 SPECIE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,627223	0,768068	0,628373	0,627321	0,633219	0,634801	0,658012	0,686333	0,750486	0,627241
Recall	0,535809	0,535793	0,535816	0,535807	0,535816	0,535821	0,535822	0,535798	0,535798	0,535825
F-measure	0,577923	0,631241	0,578415	0,577964	0,58046	0,581126	0,590664	0,601795	0,625225	0,57794
Pearson	0,506286	0,585484	0,506936	0,50685	0,507309	0,507358	0,531911	0,555874	0,584787	0,507745

MixKraken_2 SPECIE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,690789	0,750739	0,693145	0,691096	0,69339	0,694324	0,704035	0,717613	0,725806	0,691011
Recall	0,529717	0,529686	0,529802	0,529694	0,52981	0,529849	0,529855	0,529733	0,529734	0,529887
F-measure	0,599624	0,621131	0,600565	0,599725	0,600662	0,601038	0,604651	0,609524	0,61246	0,599817
Pearson	0,777532	0,784979	0,778083	0,777988	0,778102	0,778134	0,780349	0,782497	0,783047	0,778197

SRR1804065 SPECIE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,907237	0,96168	0,91937	0,908245	0,912009	0,917739	0,953925	0,95483	0,955794	0,907257
Recall	0,85616	0,856434	0,85623	0,856161	0,856165	0,856095	0,856125	0,856269	0,856254	0,856179
F-measure	0,880959	0,906011	0,886677	0,881434	0,883205	0,885846	0,902383	0,902868	0,90329	0,880978
Pearson	0,990729	0,99297	0,990864	0,99073	0,990794	0,990859	0,992903	0,992903	0,992906	0,990816

Tabella 4.7 Risultati statistici a livello SPECIE dei dataset: MixKraken_1, MixKraken_2, SRR1804065.

HiSeq GENERE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,988593	0,991282	0,992998	0,988844	0,990499	0,99048	0,990054	0,990226	0,99021	0,988466
Recall	0,78	0,705	0,78	0,78	0,7715	0,7699	0,7565	0,7396	0,7181	0,7799
F-measure	0,871996	0,823983	0,873705	0,872093	0,86739	0,86637	0,857661	0,846757	0,832483	0,871884
Pearson	0,990854	0,961638	0,989928	0,990856	0,991197	0,991133	0,988658	0,983925	0,968409	0,990873

MiSeq GENERE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,907931	0,921774	0,926583	0,910551	0,91885	0,920091	0,921679	0,923951	0,922909	0,908181
Recall	0,7258	0,6528	0,7257	0,7258	0,7258	0,7254	0,7202	0,7071	0,6776	0,726
F-measure	0,806713	0,764313	0,81393	0,807746	0,810995	0,811228	0,808578	0,80111	0,781455	0,806936
Pearson	0,929938	0,916286	0,931807	0,93002	0,931562	0,93158	0,932687	0,932558	0,921815	0,930712

simBA5 GENERE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,932776	0,964908	0,961641	0,932976	0,956294	0,958786	0,959773	0,960186	0,961001	0,932776
Recall	0,87	0,7699	0,8699	0,87	0,8599	0,8561	0,8446	0,8248	0,8181	0,87
F-measure	0,900295	0,856444	0,913473	0,900388	0,905539	0,904538	0,898511	0,887359	0,883811	0,900295
Pearson	0,925555	0,935839	0,965886	0,925571	0,96663	0,967524	0,967102	0,966714	0,965816	0,925863

Tabella 4.8 Risultati statistici a livello GENERE dei dataset: HiSeq, MiSeq, simBA5.

MixKraken_1 GENERE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,992937	0,985278	0,994746	0,993083	0,993577	0,993653	0,991256	0,988931	0,985326	0,992004
Recall	0,848221	0,687315	0,848223	0,848211	0,840743	0,83872	0,807184	0,772026	0,703457	0,847425
F-measure	0,914892	0,809757	0,91566	0,914948	0,910793	0,909637	0,8898	0,86712	0,820869	0,914033
Pearson	0,994078	0,897988	0,994055	0,994053	0,994547	0,994626	0,987942	0,97463	0,911982	0,993901

MixKraken_2 GENERE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,986134	0,986927	0,989515	0,986576	0,988319	0,988635	0,988487	0,987726	0,987662	0,986282
Recall	0,756196	0,69633	0,756332	0,756165	0,75516	0,754441	0,743931	0,729128	0,720851	0,756309
F-measure	0,855992	0,816544	0,857351	0,856139	0,856149	0,855805	0,848948	0,838951	0,833423	0,856121
Pearson	0,991846	0,986036	0,991892	0,991851	0,991926	0,992069	0,99208	0,990692	0,990095	0,991914

SRR1804065 GENERE

	Standard	Abs_Sp	Abs_Ge	Scr_1	Scr_15	Scr_25	Scr_50	Scr_75	Scr_100	Weighted
Precision	0,971429	0,981054	0,984377	0,972509	0,976494	0,982601	0,979971	0,980354	0,980771	0,970405
Recall	0,916738	0,873688	0,916773	0,91674	0,916702	0,9166	0,8795	0,879159	0,87863	0,915772
F-measure	0,943292	0,924264	0,949373	0,943801	0,945654	0,948453	0,927021	0,927003	0,926895	0,942297
Pearson	0,998346	0,998418	0,998439	0,998347	0,998388	0,998436	0,998397	0,998398	0,998397	0,998344

Tabella 4.9 Risultati statistici a livello GENERE dei dataset: MixKraken_1, MixKraken_2, SRR1804065.

	Dim (bytes)	Dim (GB)	# K-mer	# K-mer Eliminati	% K-mer Eliminati
Standard	70.381.181.032	65,54758272	5.865.098.330	0	0,000%
Abs_SPECIES	66.786.464.104	62,1997417	5.565.538.586	299.559.744	5,325%
Abs_GENUS	69.895.336.432	65,09510468	5.824.611.280	40.487.050	0,720%
Scr_disc_1	67.505.732.728	62,86961281	5.625.477.638	239.620.692	4,260%
Scr_disc_15	66.598.545.820	62,02472916	5.549.878.729	315.219.601	5,603%
Scr_disc_25	66.338.547.700	61,78258704	5.528.212.219	336.886.111	5,989%
Scr_disc_50	65.603.439.808	61,09796447	5.466.953.228	398.145.102	7,078%
Scr_disc_75	64.785.718.108	60,33640179	5.398.809.753	466.288.577	8,289%
Scr_disc_100	64.692.329.584	60,24942695	5.391.027.376	474.070.954	8,427%

Tabella 4.10 Dimensione della base di dati e numero di k-mers filtrati al variare del parametro di Score.

4.3.2 Grafici

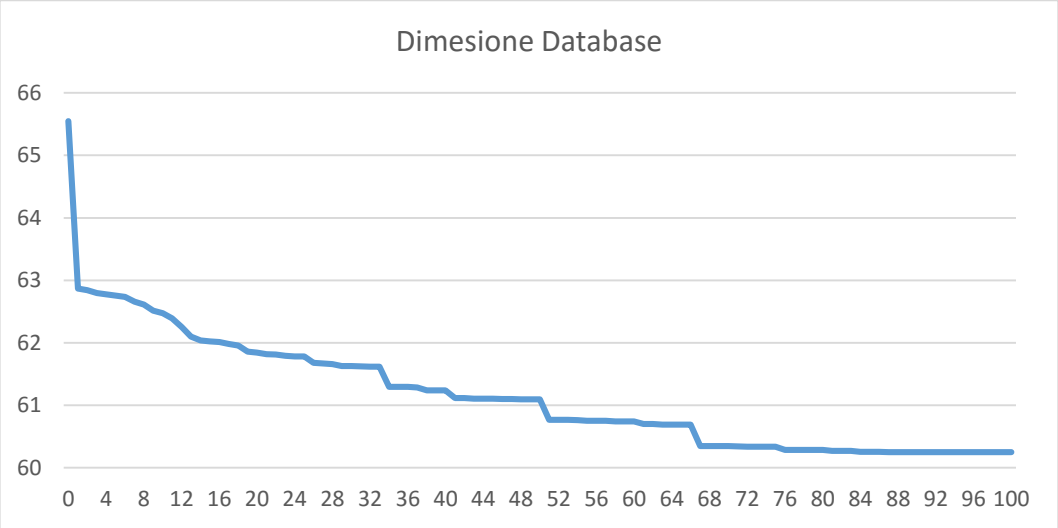

Figura 4.1 Dimensione del database in GB al variare del parametro di Score.

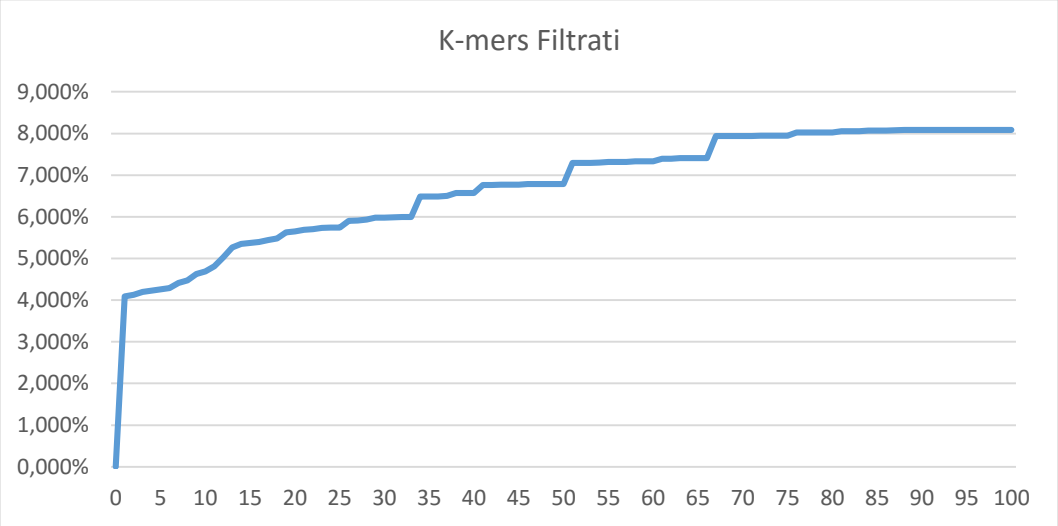

Figura 4.2 Percentuale di K-mers filtrati al variare del parametro di Score.

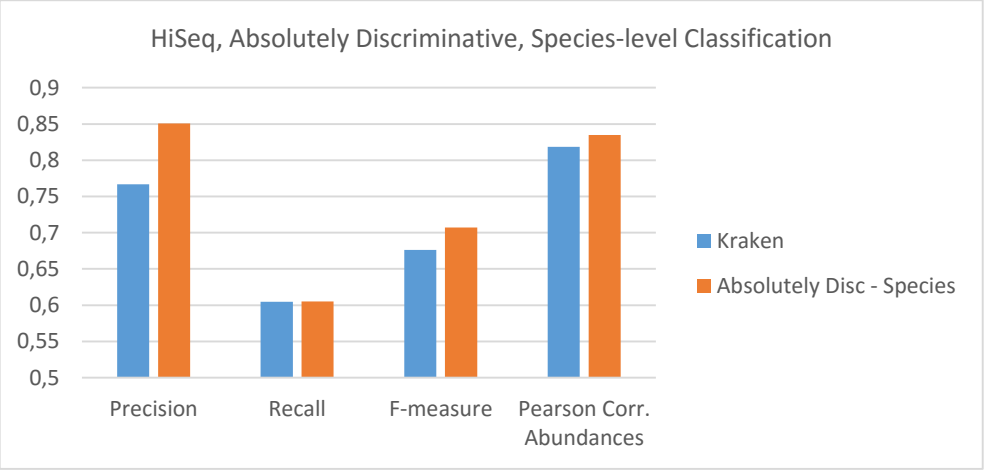

Figura 4.3 Risultati Absolutely Discriminative rank Specie sul dataset HiSeq.

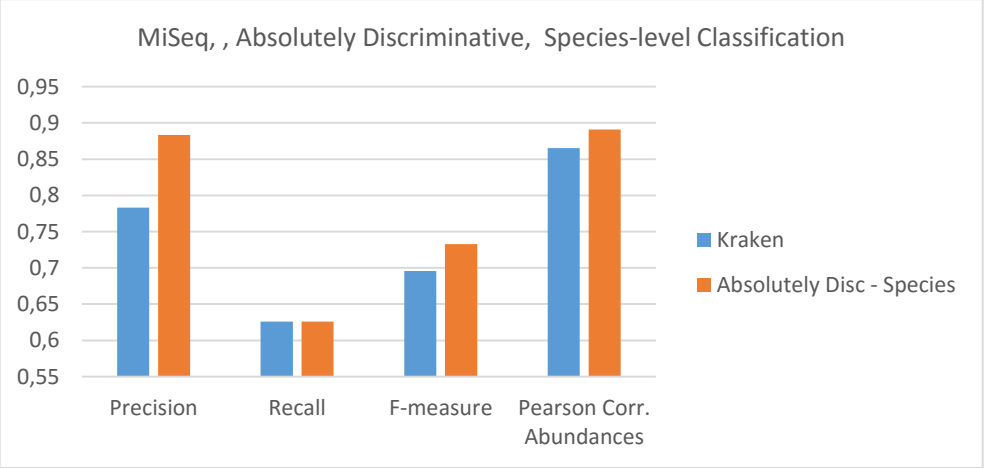

Figura 4.4 Risultati Absolutely Discriminative rank Specie sul dataset MiSeq.

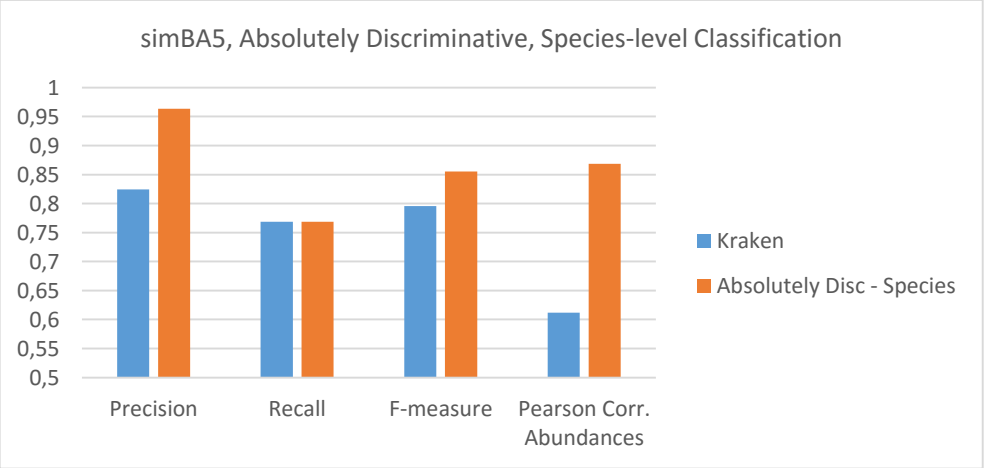

Figura 4.5 Risultati Absolutely Discriminative rank Specie sul dataset simBA5.

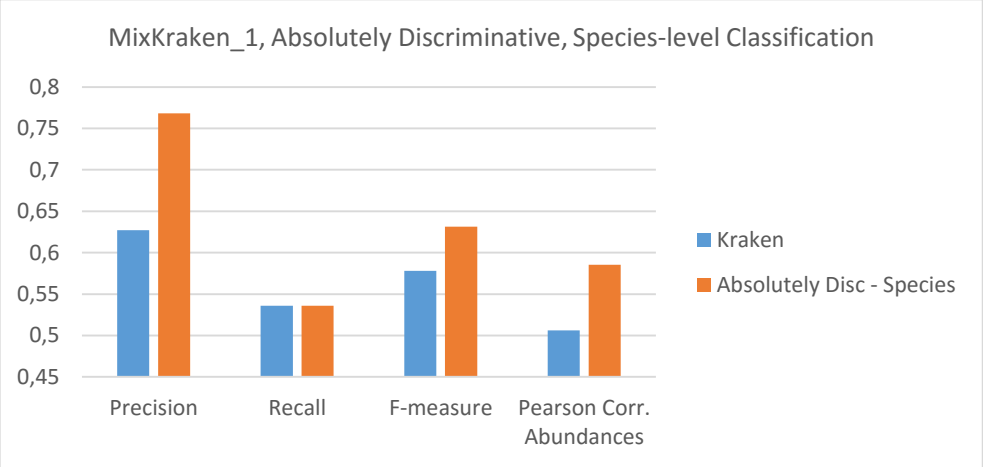

Figura 4.6 Risultati Absolutely Discriminative rank Specie sul dataset MixKraken_1.

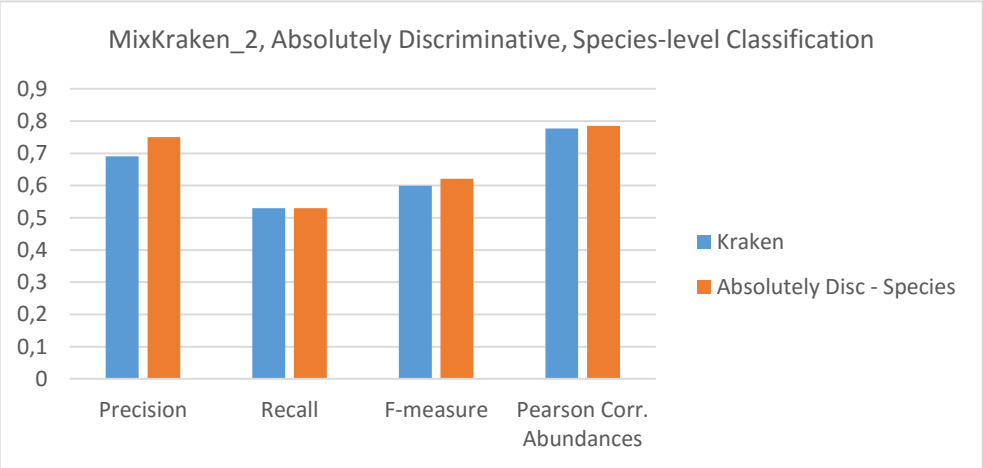

Figura 4.7 Risultati Absolutely Discriminative rank Specie sul dataset MixKraken_2.

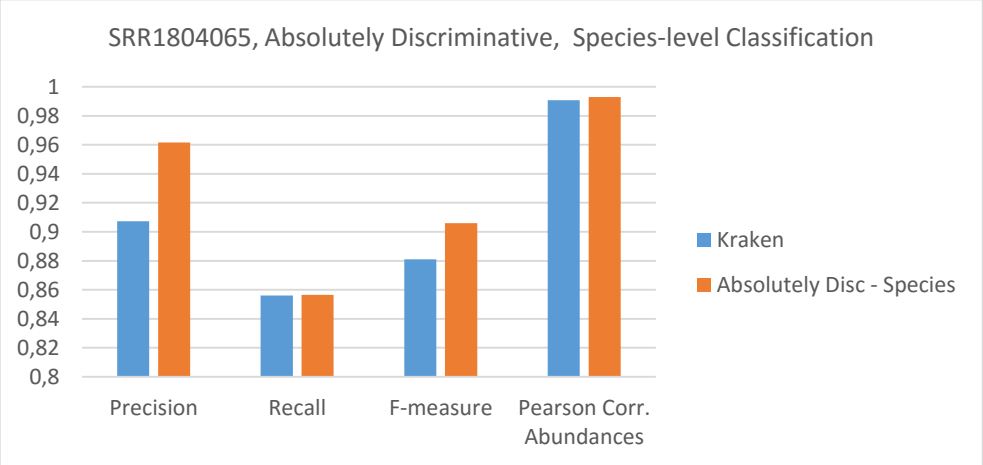

Figura 4.8 Risultati Absolutely Discriminative rank Specie sul dataset SRR1804065.

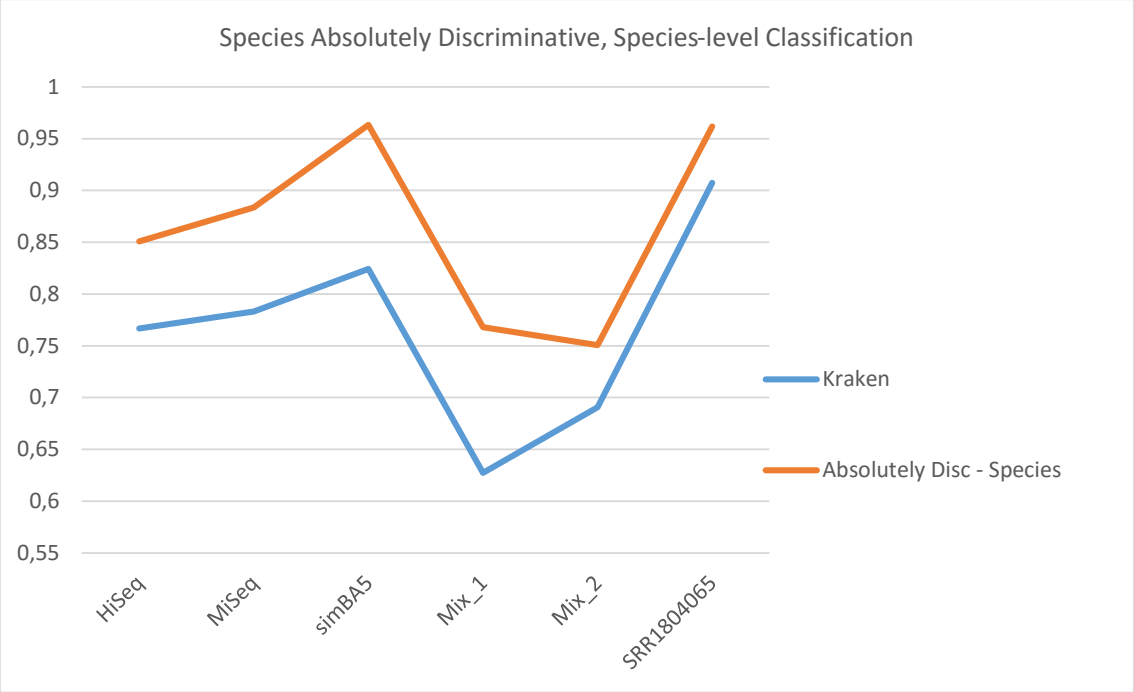

Figura 4.9 Precision della classificazione a livello Specie di Kraken e Absolutely Discriminative Species per tutti i dataset.

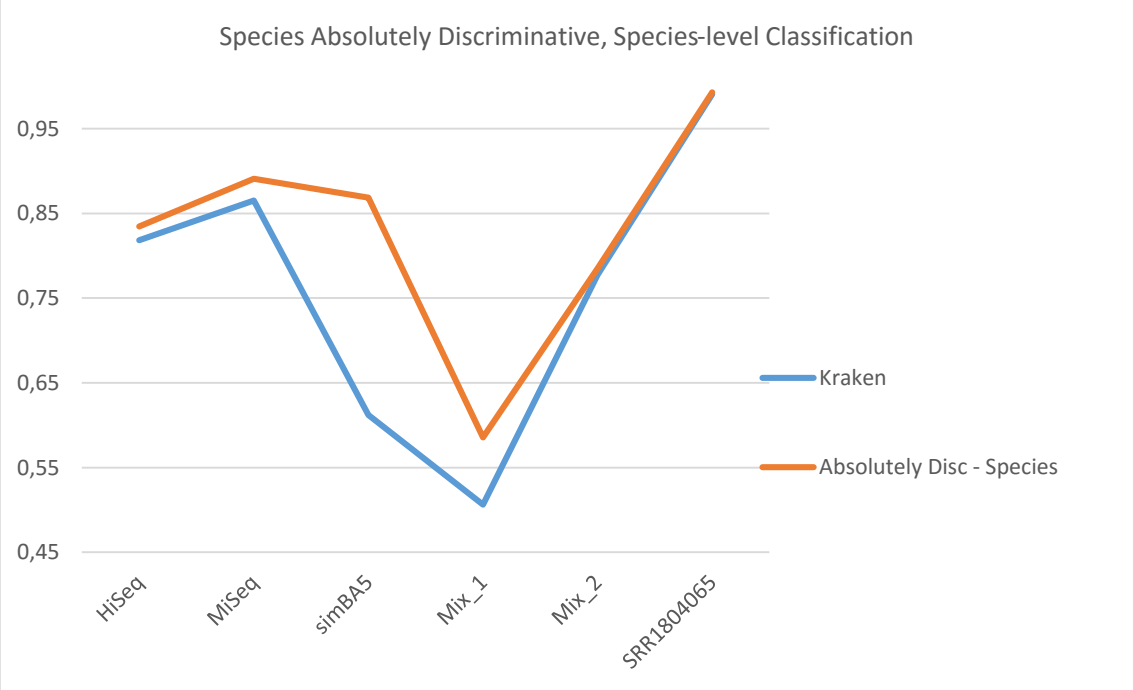

Figura 4.10 Indice di correlazione di Pearson della stima delle abbondanze per la classificazione a livello Specie di Kraken e Absolutely Discriminative Species per tutti i dataset.

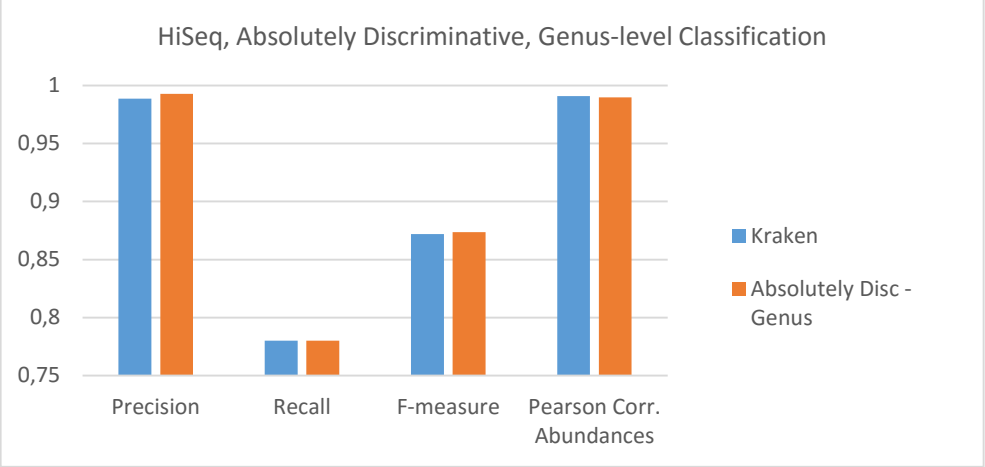

Figura 4.11 Risultati Absolutely Discriminative rank Genere sul dataset HiSeq.

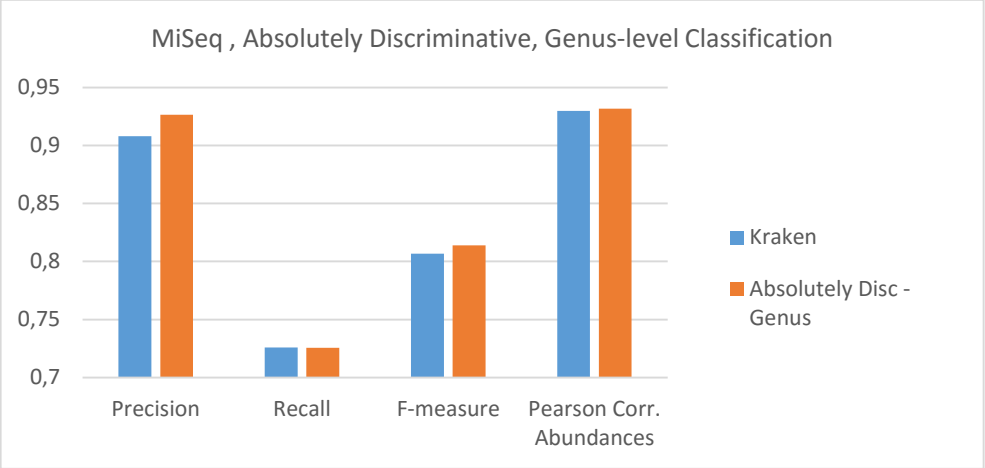

Figura 4.12 Risultati Absolutely Discriminative rank Genere sul dataset MiSeq.

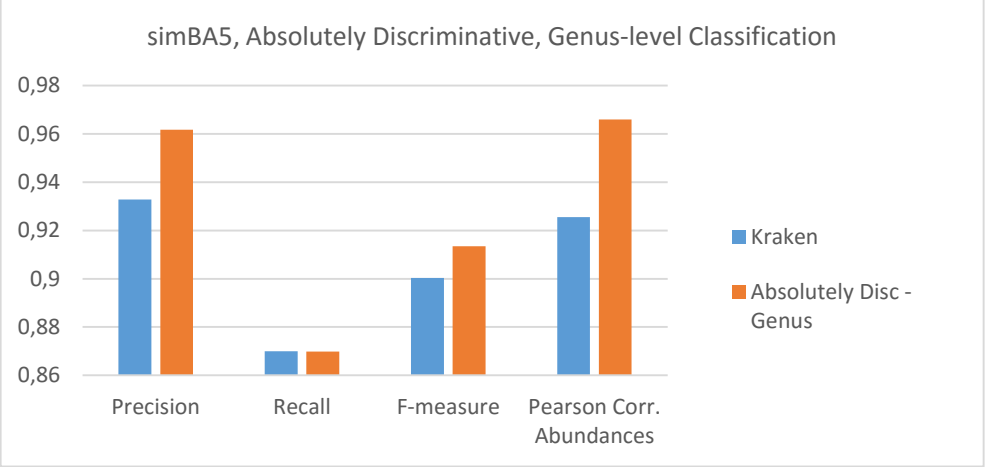

Figura 4.13 Risultati Absolutely Discriminative rank Genere sul dataset simBA5.

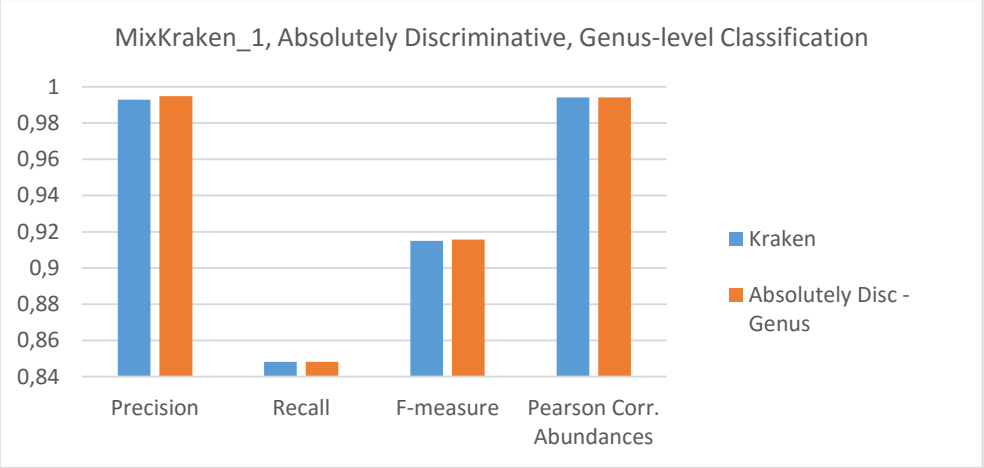

Figura 4.14 Risultati Absolutely Discriminative rank Genere sul dataset MixKraken_1.

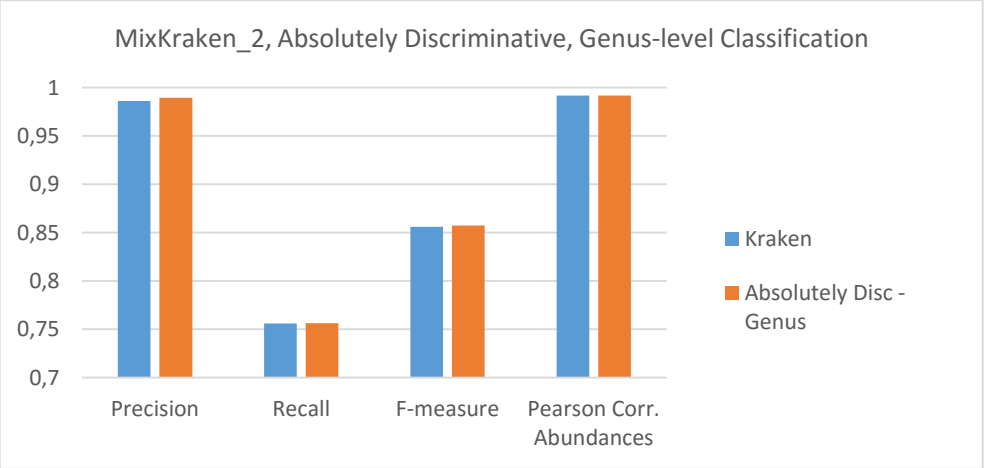

Figura 4.15 Risultati Absolutely Discriminative rank Genere sul dataset MixKraken_2.

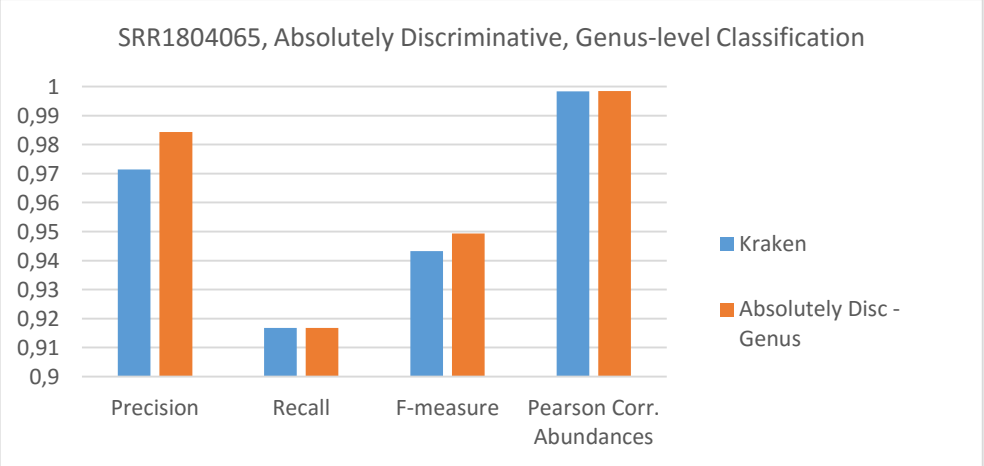

Figura 4.16 Risultati Absolutely Discriminative rank Genere sul dataset SRR1804065.

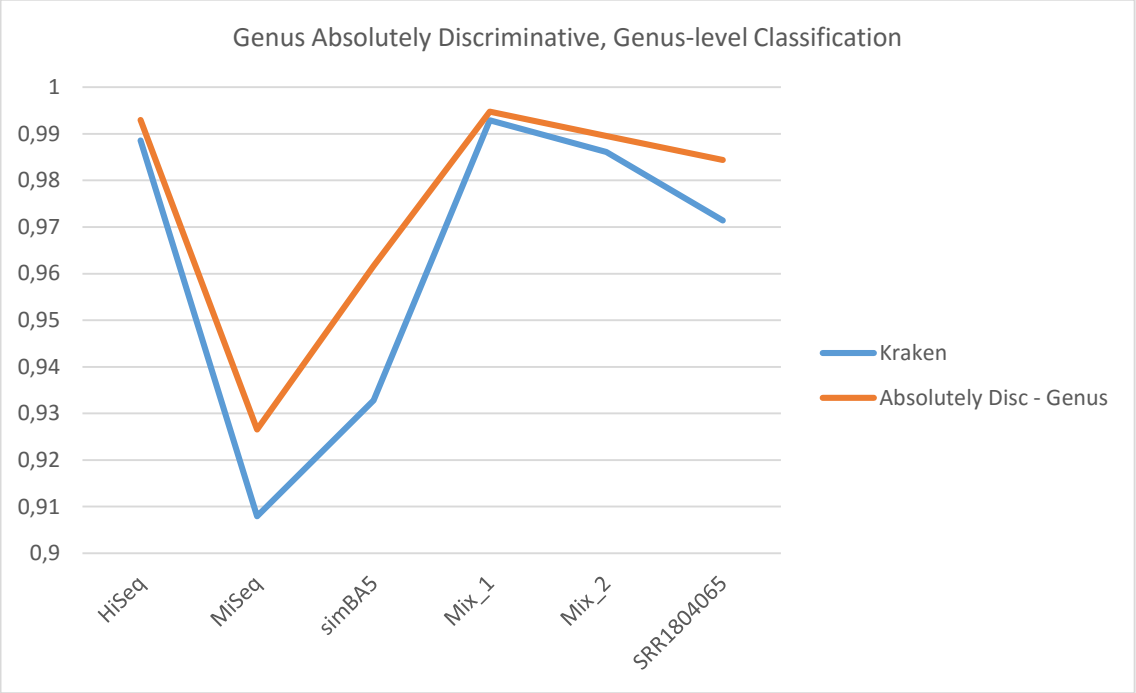

Figura 4.17 Precision della classificazione a livello Genere di Kraken e Absolutely Discriminative Genus per tutti i dataset.

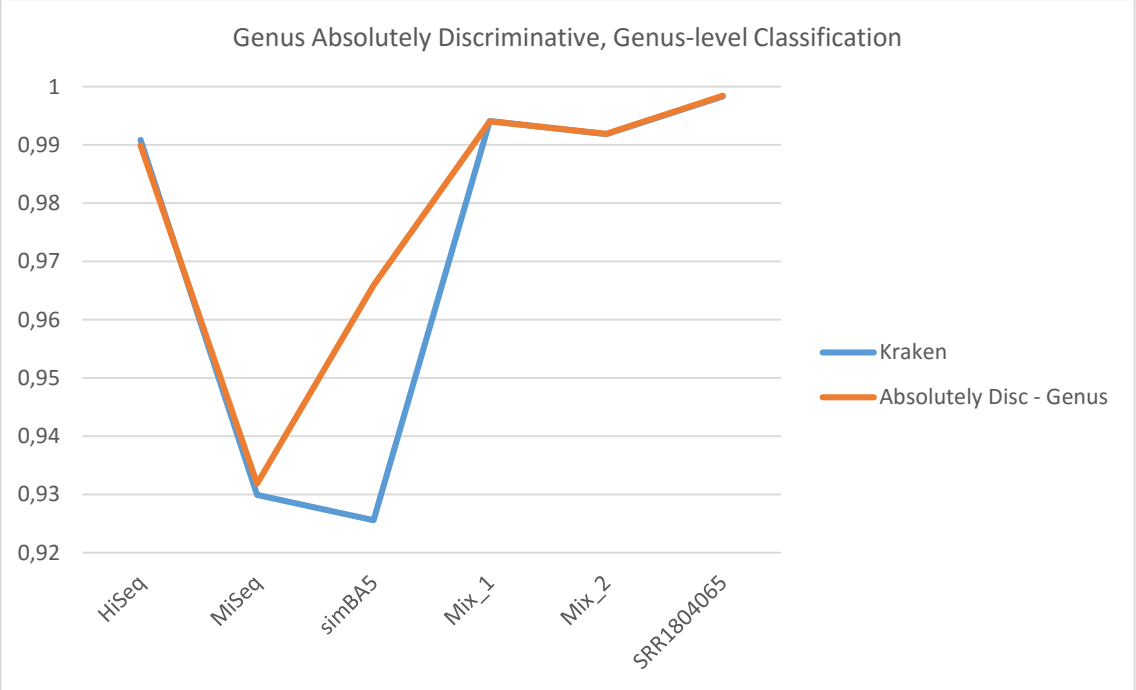

Figura 4.18 Indice di correlazione di Pearson della stima delle abbondanze per la classificazione a livello Genere di Kraken e Absolutely Discriminative Genus per tutti i dataset.

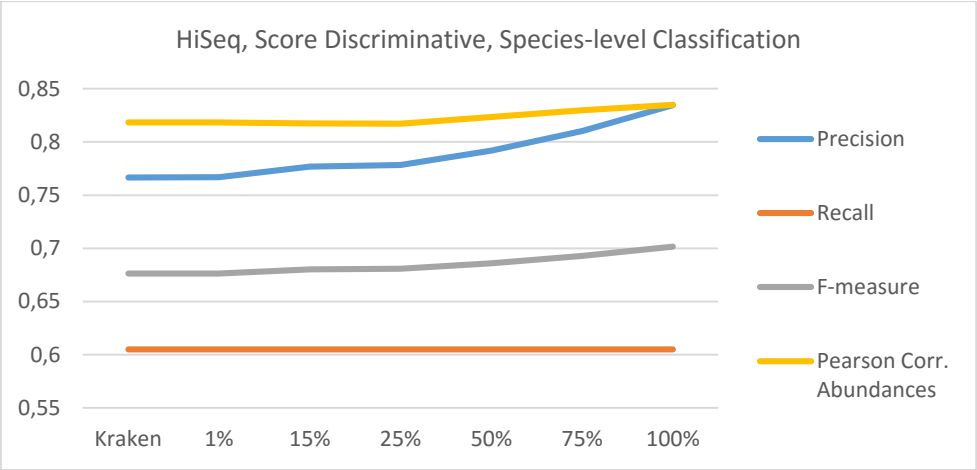

Figura 4.19 Risultati classificazione livello Specie su dataset HiSeq con la modalità Score Discriminative al variare del parametro di discriminazione.

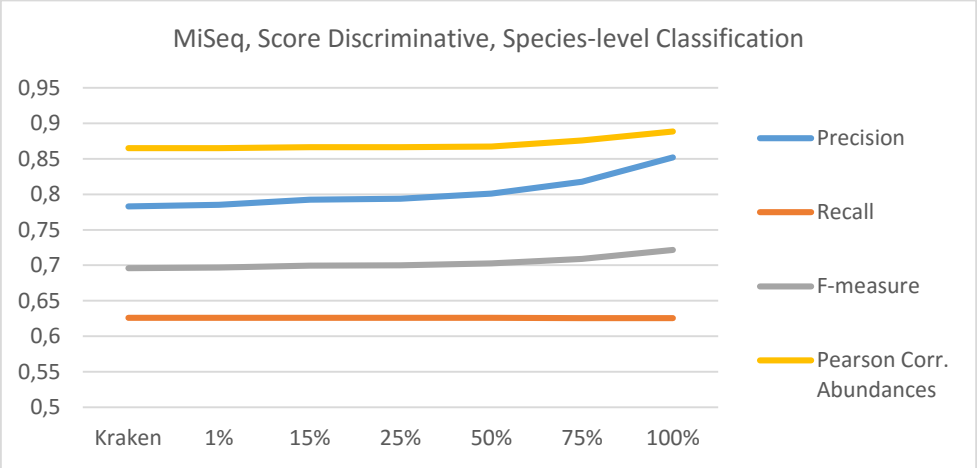

Figura 4.20 Risultati classificazione livello Specie su dataset MiSeq con la modalità Score Discriminative al variare del parametro di discriminazione.

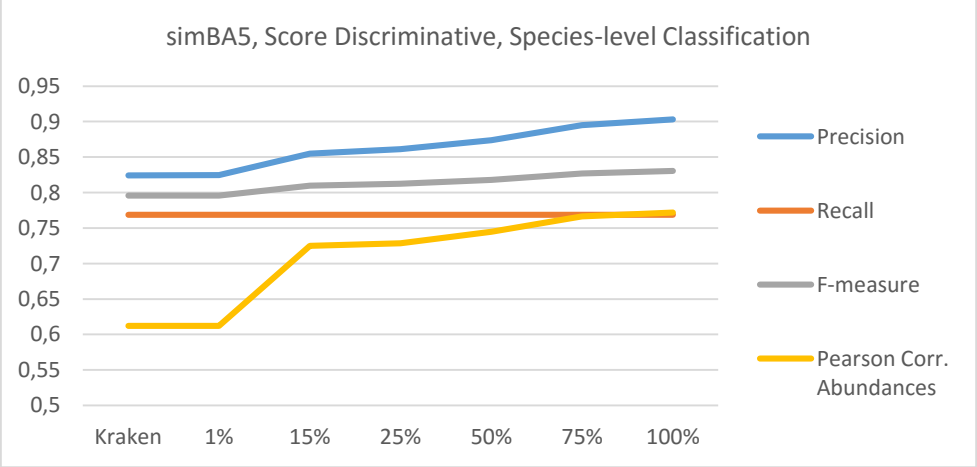

Figura 4.21 Risultati classificazione livello Species su dataset simBA5 con la modalità Score Discriminative al variare del parametro di discriminazione.

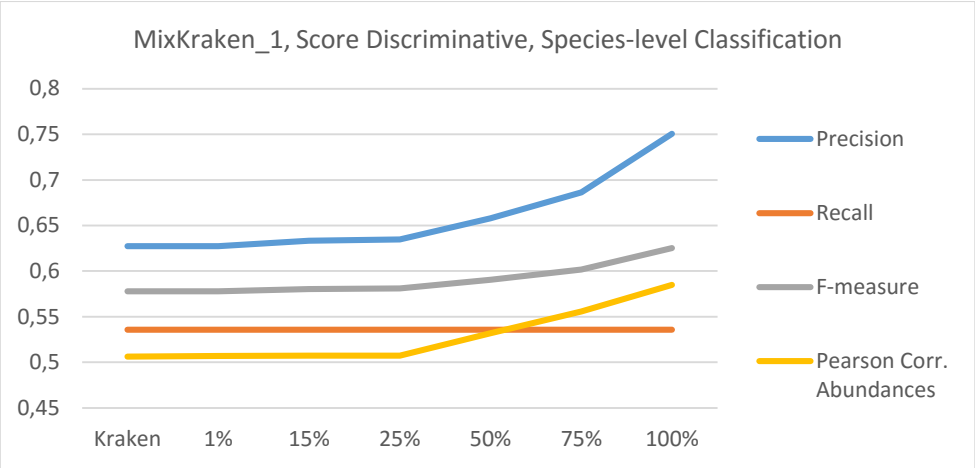

Figura 4.22 Risultati classificazione livello Specie su dataset MixKraken_1 con la modalità Score Discriminative al variare del parametro di discriminazione.

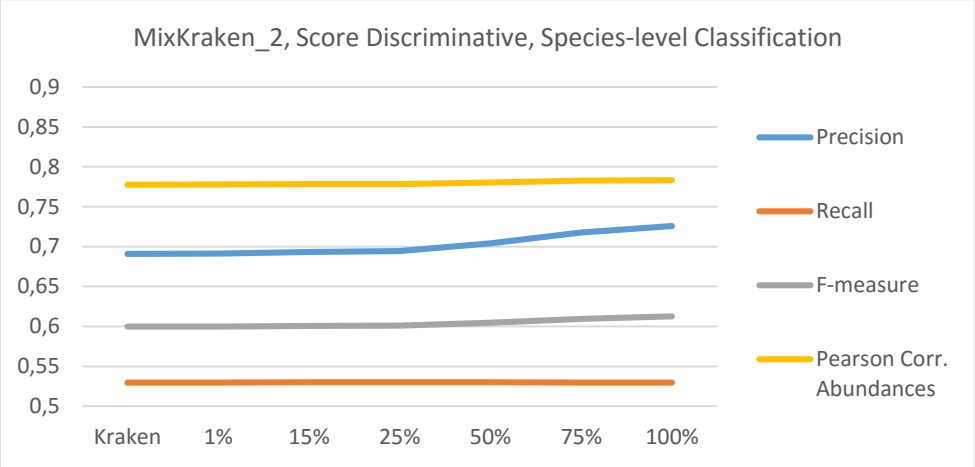

Figura 4.23 Risultati classificazione livello Specie su dataset MixKraken_2 con la modalità Score Discriminative al variare del parametro di discriminazione.

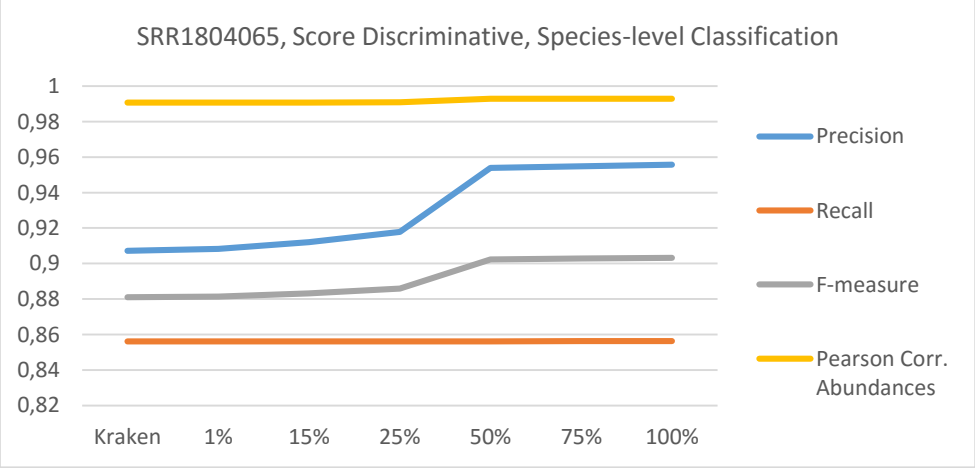

Figura 4.24 Risultati classificazione livello Specie su dataset SRR1804065 con la modalità Score Discriminative al variare del parametro di discriminazione.

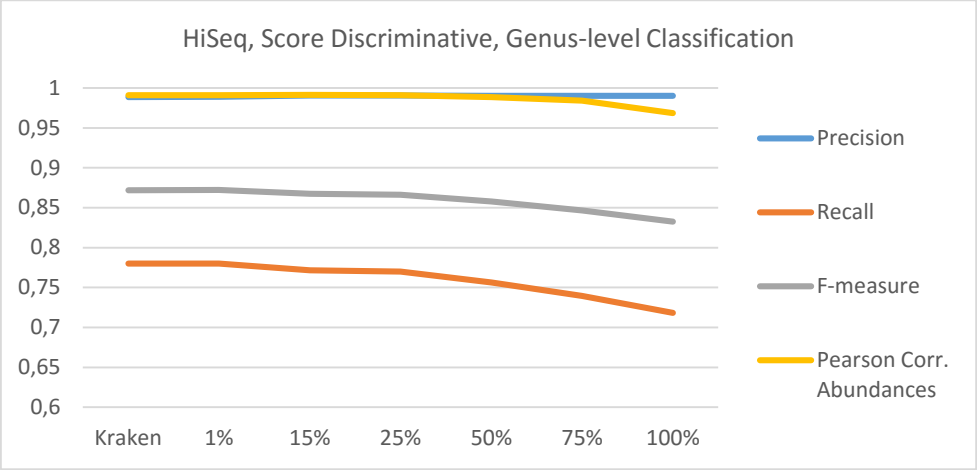

Figura 4.25 Risultati classificazione livello Genere su dataset HiSeq con la modalità Score Discriminative al variare del parametro di discriminazione.

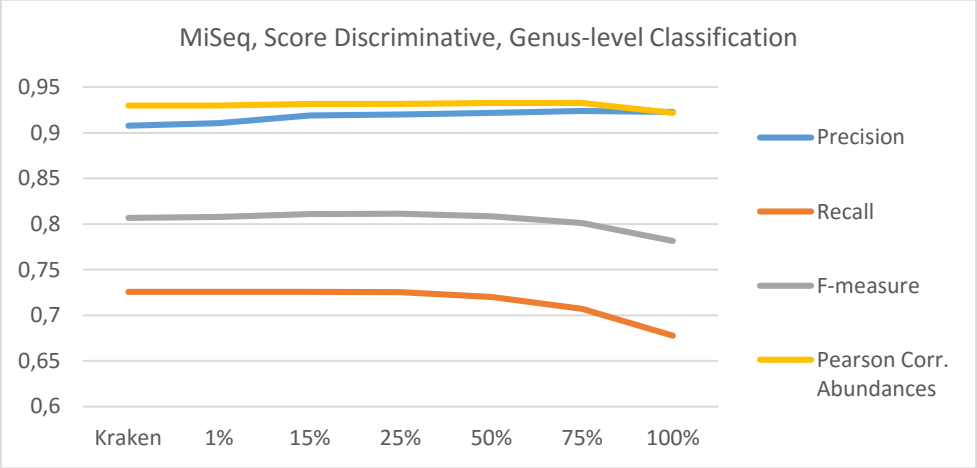

Figura 4.26 Risultati classificazione livello Genere su dataset MiSeq con la modalità Score Discriminative al variare del parametro di discriminazione.

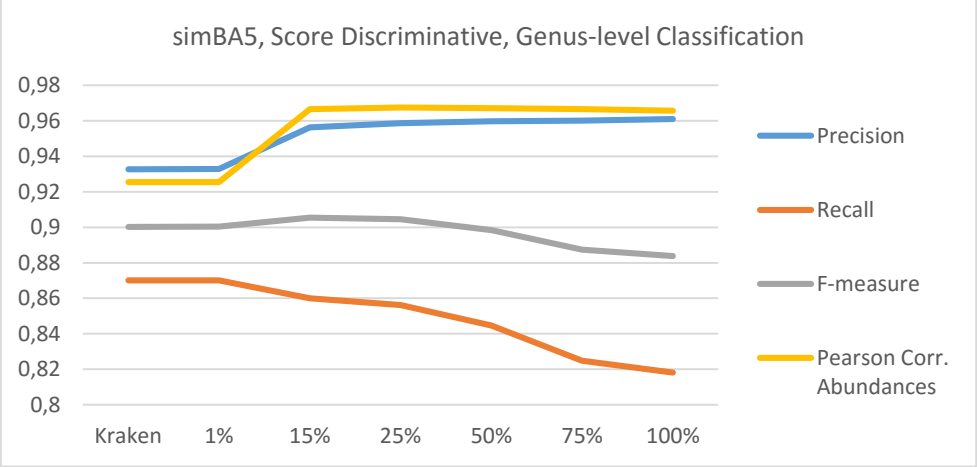

Figura 4.27 Risultati classificazione livello Genere su dataset simBA5 con la modalità Score Discriminative al variare del parametro di discriminazione.

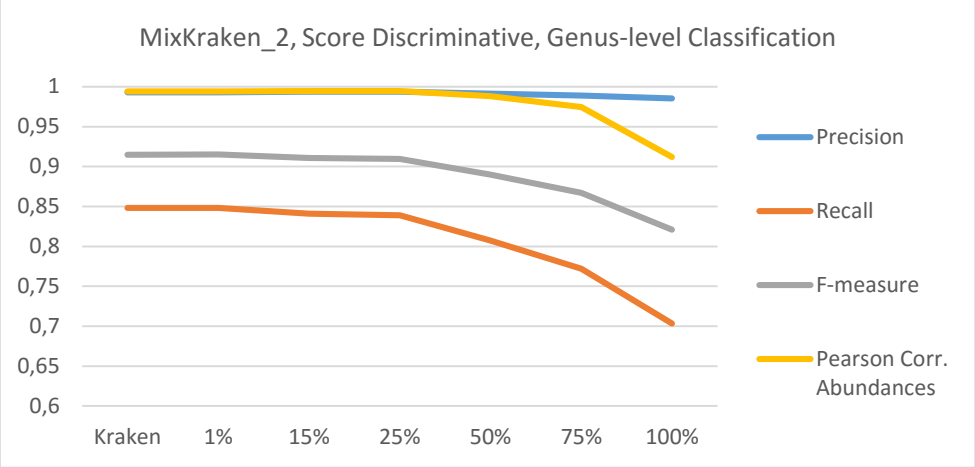

Figura 4.28 Risultati classificazione livello Genere su dataset MixKraken_1 con la modalità Score Discriminative al variare del parametro di discriminazione.

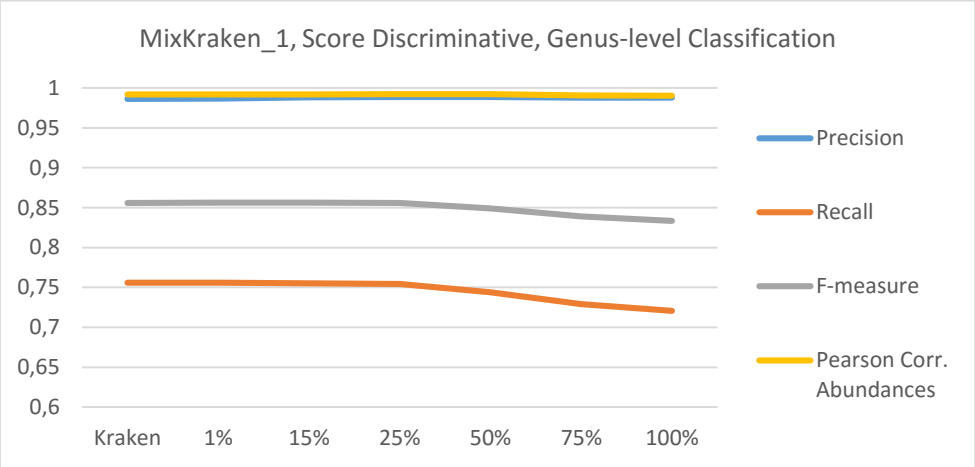

Figura 4.29 Risultati classificazione livello Genere su dataset MixKraken_2 con la modalità Score Discriminative al variare del parametro di discriminazione.

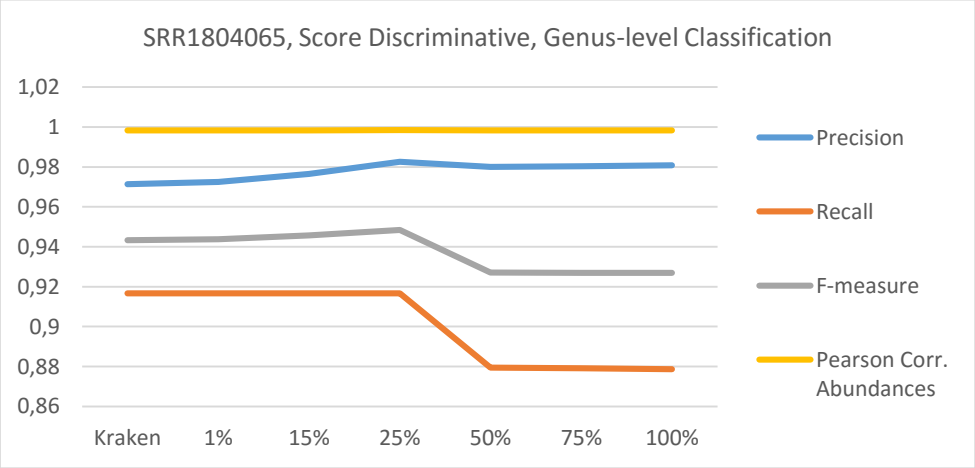

Figura 4.30 Risultati classificazione livello Genere su dataset SRR1804065 con la modalità Score Discriminative al variare del parametro di discriminazione.

4.3.3 Considerazioni

La costruzione del database e i relativi test sono stati effettuati con il cluster di calcolo 'Blade' messo a disposizione dal Dipartimento di Ingegneria dell'Informazione (DEI) dell'Università degli Studi di Padova.

Il calcolatore utilizzato ha se seguenti caratteristiche tecniche:

- 64 Processori quad core Intel Xeon E5450 (12MB Cache, 3.00 GHz)
- 256 GB di RAM

Si possono fare considerazioni approfondite sulle tabelle e grafici ottenuti. Lo scopo è quello di analizzare la qualità della classificazione delle nuove modalità proposte, e implementate, rispetto all'algoritmo originale di Kraken.

Absolutely Discriminative con normalizzazione dei nodi al rank Specie presenta un miglioramento effettivo dei parametri statistici per tutti i dataset. In particolare per la validazione a livello Specie c'è un aumento notevole della Precision, anche fino a dieci punti percentuali, mantenendo invariata la Recall con un conseguente aumento del valore della F-measure e dell'indice di correlazione di Pearson [Figura 4.3 – 4.10]. Ricordando che l'indice di Recall utilizzato è differente dalla definizione standard presente in letteratura, una sua stabilità relazionata ad un aumento della Precision, rispetto all'algoritmo standard, indica che durante la validazione è rimasto stabile il valore dei True Positive ma è diminuita la quantità di False Positive restituita. Questo significa che: tramite un filtraggio assolutamente discriminativo al livello in cui si vuole validare, vengono eliminati i k-mers che introducono il rumore responsabile di una errata classificazione di alcune read da parte dell'algoritmo standard oppure della classificazione di read che non sono possibili da classificare perché sconosciute sulla base delle sequenze di riferimento. Rimane invariato il numero di read classificate correttamente restituite ma aumenta la robustezza in termini di Precision. Aumenta la probabilità che: restituita una read come 'classificata' questa sia corretta. Un aumento dell'indice di correlazione di Pearson, e quindi dell'abbondanza, indica un miglioramento della correlazione tra le specie (o generi a seconda del livello di validazione), in termini di proporzioni, contenute nel dataset fornito in ingresso e quelle restituite in output dalle read classificate. Un valore prossimo a 1 indica una correlazione assoluta e quindi una maggiore affidabilità da parte del classificatore nell'individuare la

diversità e le proporzioni delle specie (o generi) presenti. Discorso analogo viene fatto per la modalità Absolutely Discriminative con normalizzazione dei nodi al rank Genere per la validazione a livello Genere [Figura 4.11 - 4.18]. La modalità Absolute Discriminative permette, oltre al miglioramento delle prestazioni di classificazione, una diminuzione della dimensione del database dovuto al filtraggio dei k-mers non considerati assolutamente discriminativi [Tabella 4.10].

Score Discriminative con validazione a livello Specie presenta per tutti i dataset considerati un aumento della Precision con un mantenimento stabile della Recall, un conseguente aumento della F-measure e dell'indice di correlazione di Pearson [Figura 4.19 – 4.24]. In particolare si nota come gli aumenti diventino sempre più significativi con l'incrementare del parametro di discriminazione. Il vantaggio della modalità Score-discriminative rispetto alla Absolutely-discriminative è un maggior filtraggio dei k-mers con conseguente diminuzione della dimensione del database finale [Tabella 4.10]. Si vede che con un parametro pari a 100 (il massimo settabile) si ottiene il miglioramento delle prestazioni di classificazioni con un'eliminazione dei k-mers pari a circa l'8% del totale presente [Figura 4.1, 4.2]. La validazione a livello Genere evidenzia delle perdite prestazionali delle classificazioni all'aumentare del parametro di discriminazione [Figura 4.25 – 4.30]. Una possibile soluzione a questo problema potrebbe essere di uniformare le etichette tassonomiche delle sequenze analizzate a livello Genere e calcolare gli score sull'albero tassonomico risultante. Questo approccio potrebbe portare a notevoli miglioramenti degli indici statistici analogamente a quello che è successo con la modalità Absolutely Discriminative.

Non sono stati riportati i grafici riguardanti la modalità Score Weighted Classification in quanto, come si vede dai risultati [Tabella 4.6 - 4.9] le prestazioni rimangono invariate e quindi non c'è un reale vantaggio nell'utilizzo degli score come peso durante la classificazione. Questo è dovuto al fatto che nel calcolo degli score si ottiene che circa il 92% dei k-mers ottiene un punteggio pari al 100%. Questo non influisce in maniera significativa nella scelta del cammino RTL (root to leaf) durante la fase di classificazione. Inoltre questa tecnica non porta nessun vantaggio dal punto di vista del risparmio di risorse rispetto all'algoritmo originale.

È stata testata anche una ulteriore modalità che prevedeva una variazione della Absolutely Discriminative con pesatura dei k-mers in base allo score (ottenuto con la stessa modalità dello Score Discriminative). Non sono stati riportati i risultati poiché di poco interesse. I valori statistici rimangono pressoché invariati e questo è probabilmente dovuto al fatto che se un k-mers è stato considerato assolutamente discriminativo allora avrà molto probabilmente ottenuto anche uno score pari a 100 o prossimo ad esso, con conseguente normalizzazione di tutti i pesi dei possibili k-mers considerati allo stesso valore.

Per problemi di natura tecnica non è stato possibile ricavare informazioni utili riguardante la variazione dei tempi di classificazione delle varie modalità. Il problema è causato dal cluster di calcolo utilizzato per effettuare i test. Tutti i processi eseguiti hanno una priorità in base al tipo di utente e alle priorità degli processi nella stessa coda in quel momento. Per questo motivo capita che per uno stesso test ci si impieghi un secondo oppure addirittura un migliaio. La soluzione per ottenere dei valori veritieri sarebbe quella di effettuare i test su un elaboratore dotato delle risorse computazioni adeguate con tutti i permessi disponibili in modo da sfruttare tutte le risorse senza interruzioni di alcun genere. Si possono comunque fare delle assunzioni riguardo i tempi di classificazione. Il tempo di classificazione è fortemente influenzato dall'operazione di ricerca dei k-mers all'interno del database. Tramite l'indicizzazione Kraken riesce ad effettuare queste operazioni molto velocemente. Una diminuzione dei k-mers memorizzati tramite il filtraggio (modalità Absolutely Discriminative e Score Discriminative) prevede quindi una riduzione della quantità dei k-mers nella base di dati compressa in cui cercare la sottosequenza della read che si sta classificando. Questo porta a supporre una riduzione dei tempi necessari alla classificazione in proporzione al numero di k-mers eliminati dal database originale.

Capitolo 5 Conclusioni

In questo lavoro di tesi è stato analizzato il problema della classificazione delle sequenze metagenomiche e studiato l'approccio di risoluzione, tramite allineamento dei k-mers, adottato dal classificatore ultraveloce Kraken. Sono state progettate, implementate e testate nuove varianti dell'algoritmo originale al fine di valutare un possibile miglioramento in termini prestazionali delle classificazioni.

Sono stati ottenuti ottimi risultati, in particolare con la validazione dei test a livello Specie che è considerato il livello più difficile da classificare. Le varianti Absolutely Discriminative e Score Discriminative permettono un miglioramento delle prestazioni statistiche delle classificazioni tramite un filtraggio dei k-mers contenuti nel database. Permettono inoltre una compressione della base di dati necessaria alla classificazione.

I risultati ottenuti sono soddisfacenti ma ci sono dei possibili sviluppi che potrebbero portare ad un ulteriore miglioramento delle prestazioni:

- Implementazione di un algoritmo di assegnamento dello Score considerando la normalizzazione dei nodi dell'albero di classificazione ad uno specifico livello tassonomico. Questo approccio potrebbe portare ad un miglioramento significativo delle prestazioni della modalità Score Discriminative con validazione a livello Genere.
- Classificazione pesata dei k-mers considerati assolutamente discriminativi con la nuova modalità di assegnamento dello Score descritta al punto precedente.

- Analisi della relazione tra tempo necessario alla classificazione di una read e percentuale di k-mers filtrati (oppure dimensione finale base di dati compressa).
- Implementazione delle nuove modalità proposte sfruttando cicli già esistenti effettuati dall'algoritmo originale in modo tale da velocizzare la creazione della base di dati.

Si può notare come l'importante non è solamente avere a disposizione una grande quantità di dati ma anche che questi siano di qualità poiché se anche solo pochi di questi non sono correttamente utilizzati possono indurre in errore un algoritmo di classificazione di tipo statistico come è quello utilizzato da Kraken. Una procedura di filtraggio tramite una metodologia di valutazione della qualità dei singoli k-mers (in questo caso la discriminatività) permette di memorizzare solo informazioni realmente significative al fine della classificazione. Se correttamente eseguita questa procedura porta ad una riduzione delle risorse computazioni necessarie e un conseguente incremento nella velocità di classificazione delle sequenze.

Bibliografia

[Alt90] Altschul S, Gish W, Miller W, Myers E, Lipman D: Basic local alignment search tool. J Mol Biol. 1990, 215: 403-410.

[Ame13] Ames, S. K., Hysom, D. A., Gardner, S. N., Lloyd, G. S., Gokhale, M. B., and Allen, J. E. (2013). Scalable metagenomic taxonomy classification using a reference genome database. Bioinformatics, 29.

[Bra09] Brady A, Salzberg SL: Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models. Nat Methods. 2009, 6: 673-676.

[Bra11] Brady A, Salzberg S: PhymmBL expanded: confidence scores, custom databases, parallelization and more. Nat Methods. 2011, 8: 367.

[Bro15] Brown, C., Hug, L., Thomas, B., Sharon, I., Castelle, C., and Singh, A. e. a. (2015). Unusual biology across a group comprising more than 15% of domain bacteria. Nature, 523(7559):208–11.

[Cap10] Caporaso, J. G., Kuczynski, J., Stombaugh, J., Bittinger, K., Bushman, F. D., Costello, E. K., Fierer, N., Pea, A. G., Goodrich, J. K., Gordon, J. I., Huttley, G. A., Kelley, S. T., Knights, D., Koenig, J. E., Ley, R. E., Lozupone, C. A., McDonald, D., Muegge, B. D., Pirrung, M., Reeder, J., Sevinsky, J. R., Turnbaugh, P. J., Walters, W. A., Widmann, J., Yatsunenko, T., Zaneveld, J., and Knight, R. (2010). Qiime allows analysis of high-throughput community sequencing data. Nature methods, 7(5):335336.

[Ger11] Gerlach W, Stoye J: Taxonomic of metagenomic shotgun sequences with CARMA3. Nucleic Acids Res. 2011.

[Hus07] Huson, D. H., Auch, A. F., Qi, J., and Schuster, S. C. (2007). Megan analysis of metagenomic data. Genome Res., 17.

[Ken12] Kent WJ: BLAT - the BLAST-like alignment tool. Genome Res 2002;12(4):656–64.

[Lan09] Langmead B, Trapnell C, Pop M, Salzberg SL: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology. 2009, 10:R25.

[Lin15] Lindgreen, S., Adair, K. L., and Gardner, P. (2015). An evaluation of the accuracy and speed of metagenome analysis tools. bioRxiv.

[Liu10] Liu B, Gibbons T, Ghodsi M, et al. MetaPhyler: taxonomic profiling for metagenomic sequences. Proc IEEE Bioinform. Biomed. 2010, 95–100.

[Liu11] Liu, B., Gibbons, T., Ghodsi, M., Treangen, T., and Pop, M. (2011). Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences. BMC Genomics, 12.

[Man12] Mande, S. S., Mohammed, M. H., and Ghosh, T. S. (2012). Classification of metagenomic sequences: methods and challenges. Briefings in Bioinformatics, 13(6):669–681.

[Oun15] Ounit, R., Wanamaker, S., Close, T. J., and Lonardi, S. (2015). Clark: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. BMC Genomics, 16(1):1–13.

[Qin10] Qin, J., Li, R., Raes, J., and et al. (2010). A human gut microbial gene catalogue established by metagenomic sequencing. Nature, (464):5965.

[Seg12] Segata, N., Waldron, L., Ballarini, A., Narasimhan, V., Jousson, O., and Huttenhower, C. (2012). Metagenomic microbial community profiling using unique clade-specific marker genes. Nat Methods, 9.

[Rac15] Rachid Ounit, Steve Wanamaker, Timothy J Close, Stefano Lonardi: CLARK fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. BMC Genomics. 2015, 16:236.

[Ros08] Rosen G, Garbarine E, Caseiro D, Polikar R, Sokhansanj B: Metagenome fragment classification using N-mer frequency profiles. Adv Bioinformatics. 2008, 1-12.

[Woo14] Wood, D. and Salzberg, S. (2014). Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biol., 15.

[Zel14] Zeller, G., Tap, J., Voigt, A. Y., Sunagawa, S., Kultima, J. R., Costea, P. I., Amiot, A., B¨ohm, J., Brunetti, F., Habermann, N., Hercog, R., Koch, M., Luciani, A., Mende, D. R., Schneider, M. A., Schrotz-King, P., Tournigand, C., Tran Van Nhieu, J., Yamada, T., Zimmermann, J., Benes, V., Kloor, M., Ulrich, C. M., von Knebel Doeberitz, M., Sobhani, I., and Bork, P. (2014). Potential of fecal microbiota for early-stage detection of colorectal cancer. Molecular Systems Biology, 10(11).

[Zhe04] Zheng Zhang, Scott Schwartz, L.W. and Miller.,W. (2004). A greedy algorithm for aligning dna sequences. Journal of Computational Biology, 7(1-2):203–214