Chimeric Reads Detection

M. Di Matteo, G. Di Pascale e M. M. Napolitano

Ottobre 2024

Abstract

L'analisi delle sequenze geniche, e in particolare la rilevazione delle letture chimeriche, rappresenta una sfida significativa in bioinformatica a causa della complessità dei dati e del tempo necessario per le fasi di preprocessing. Generare e trovare dataset specifici che contengano "reads" chimeriche e non chimeriche per specifici geni è fondamentale, ma questo può essere costoso in termini di risorse e tempo. Queste "reads" sono necessarie per la preparazione e la standardizzazione degli strumenti bioinformatici come ad esempio FusionCatcher o per l'addestramento di modelli d'intelligenza artificiale. Il nostro lavoro propone varie pipeline avanzate che automatizzano la creazione di dataset chimerici o meno e la generazione di letture sintetiche, sia single-end sia paired-end, a partire da geni selezionati e adattabili. Utilizzando Python, abbiamo sviluppato una serie di script e automazioni per facilitare il benchmarking di modelli e strumenti bioinformatici. La pipeline integra strumenti modulari, come Fusim per la simulazione di fusioni e ART-Illumina per la generazione delle letture sintetiche, con un sistema avanzato per la costruzione di dataset e dataframe Pandas. A partire dalle sequenze k-merizzate sono stati estratti gli embeddings utilizzando il modello DNA-BERT, la lista di embeddings è stata associata ad una label chimerica/non chimerica e utilizzando questo dataset è stato addestrato un modello RNN per la classificazione delle sequenze. La soluzione proposta facilita quindi tutte le fasi di preprocessing riguardanti sequenze, letture, fusioni chimeriche e la creazione dei relativi dataset. Propone inoltre una pipeline completa, avanzata e modulabile che si conclude con all'addestramento di un modello RNN. Migliorando così l'efficienza nella creazione di dataset, riducendo i costi computazionali e ottimizzando i flussi di lavoro per l'addestramento.

1 Introduzione

Il nostro lavoro affronta la creazione e la gestione di dataset contenenti letture chimeriche e non chimeriche. Questi rappresentano una sfida sia per l'onerosità dei processi di preparazione sia per il dispendio di risorse computazionali. L'analisi delle sequenze geniche e la rilevazione delle letture chimeriche, infatti, rappresenta un tema di grande rilevanza e complessità nell'ambito bioinformatico. La costruzione di questi dataset è indispensabile per addestrare modelli di machine learning avanzati e per valutare strumenti bioinformatici già noti.

In tale documento viene inizialmente fornita una panoramica dei principali concetti di intelligenza artificiale e machine learning applicati alla bioinformatica, con un focus su reti neurali, elaborazione del linguaggio naturale (NLP), tool come `FusionCatcher` e modelli come `DNABERT`, adattati per l'analisi delle sequenze genomiche. Viene esplorata, inoltre, la tecnica di *tokenizzazione* e il suo ruolo cruciale nella rappresentazione delle sequenze genomiche per l'IA.

Viene poi descritta nel dettaglio la pipeline sviluppata per la generazione e la simulazione delle letture chimeriche. In particolare è spiegato il processo di configurazione e utilizzo degli strumenti come `Fusim` e `ART-Illumina`, specificando come i geni selezionati vengono combinati per creare letture chimeriche e sintetiche. È stato inoltre realizzato un container Docker contenente lo strumento `FusionCatcher`, semplificandone la distribuzione e l'utilizzo.

Successivamente si esplora il processo di costruzione dei dataset finali e la generazione degli embeddings. La pipeline produce dataset che includono sequenze etichettate come chimeriche o non chimeriche, utili per l'addestramento di modelli di machine learning. In particolare, è descritto il processo di *k-merizzazione* delle sequenze e l'uso di `BERT` per ottenere rappresentazioni numeriche, *embeddings*, delle letture genomiche.

Si illustra poi l'implementazione di un modello di *reti neurali ricorrenti (RNN)* progettato per la classificazione delle letture chimeriche. Viene discusso il processo di addestramento e testing del modello, insieme alle prestazioni ottenute in termini di accuratezza e capacità di generalizzazione.

2 Background

L'*intelligenza artificiale* (AI) è un campo multidisciplinare che si occupa dello sviluppo di sistemi e algoritmi in grado di emulare o replicare alcune capacità dell'intelligenza umana. L'obiettivo dell'intelligenza artificiale è quello di creare macchine con capacità simili a quelle di un essere umano per ragionare, comprendere, percepire, comunicare e prendere decisioni. Reti neurali, algoritmi genetici, logica simbolica, sistemi esperti e apprendimento automatico sono solo alcuni degli approcci e delle metodologie che compongono l'intelligenza artificiale. L'apprendimento automatico, chiamato in letteratura *Machine Learning*, è il metodo che consente alle macchine di imparare dai dati senza essere programmate in modo esplicito ma si adattano e migliorano attraverso l'esperienza. L'obiettivo del machine learning è quello di sviluppare algoritmi che possano analizzare e interpretare i dati, rilevare modelli e tendenze nascoste e fare previsioni o prendere decisioni basate su queste informazioni.

2.1 Reti Neurali

Le reti neurali sono composte da unità di elaborazione chiamate neuroni o nodi. Tali neuroni sono organizzati in strati: uno strato di input, uno o più strati nascosti e uno strato di output e sono interconnessi tra loro tramite connessioni pesate. Durante la fase di addestramento, vengono forniti alla rete dati di input insieme alle loro etichette di output corrispondenti. La rete regola iterativamente i pesi delle connessioni tra i neuroni per minimizzare l'errore tra le previsioni della rete e le etichette di output desiderate. Tale processo di apprendimento consente alla rete neurale di migliorare le sue prestazioni nel tempo, adattandosi ai dati di addestramento e generalizzando le conoscenze acquisite per fare previsioni su nuovi dati. Le reti neurali possono essere strutturate in diversi modi, tra cui reti neurali feedforward, reti neurali ricorrenti e reti neurali convoluzionali. Una *rete neurale ricorrente* (RNN) è un tipo di rete neurale artificiale che utilizza dati sequenziali o dati di serie temporali. Tali algoritmi di deep learning sono comunemente usati per problemi di tipo ordinale o temporale, come la traduzione linguistica. La caratteristica principale delle RNN è il loro loop ricorrente che consente di condividere i pesi e l'informazione tra gli step temporali. Ogni step temporale di una RNN elabora un input corrente e tiene traccia di uno stato interno che rappresenta l'informazione precedente. Questo stato interno viene quindi

propagato all'input successivo, consentendo alla rete di catturare dipendenze a lungo termine all'interno dei dati sequenziali. Si distinguono, quindi, per la loro *"memoria a breve termine"* che permette di prelevare informazioni dagli input precedenti per influenzare l'input e l'output attuali. Poichè le RNN

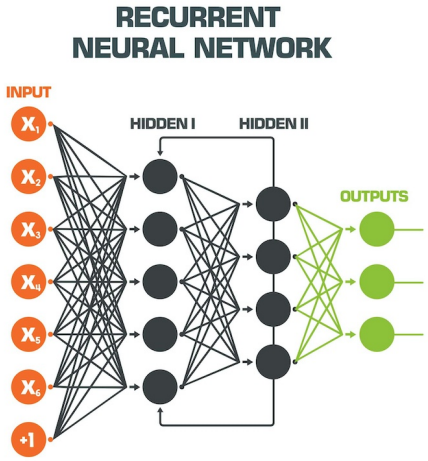

Figura 1: Esempio funzionamento rete neurale ricorrente (RNN)

utilizzano un algoritmo chiamato *backpropagation* per aggiornare i loro pesi, si può incorrere nel problema della scomparsa del *gradiente* il quale rappresenta la direzione e l'intensità del cambiamento necessario per migliorare la performance della rete. Tale problema si verifica quando i gradienti calcolati durante la backpropagation diventano molto piccoli. Quando il gradiente è vicino a zero, l'aggiornamento dei pesi durante il training è quasi inesistente, il che significa che la rete smette di apprendere o impiega molto tempo per farlo. Per affrontare tale problema, sono state sviluppate varianti delle RNN, come le *reti neurali ricorrenti a memoria a lungo termine* (LSTM) e le *reti neurali ricorrenti a porte* (GRU), che introducono meccanismi di memoria aggiuntivi per affrontare le dipendenze a lungo termine.

2.2 Natural Language Processing e Tokenizzazione

L'elaborazione del linguaggio naturale (***NLP***) è un campo multidisciplinare che si occupa dell'interazione tra linguaggio umano e computer, con l'obiettivo primario di consentire ai computer di comprendere, interpretare e generare

il linguaggio umano in modo significativo. La ***NLP*** viene utilizzata in diversi campi, tra cui elaborazione automatica di testi, la traduzione automatica, l'analisi del sentiment e molto altro.

Alcune delle sfide della ***NLP*** sono: l'ambiguità, ovvero la definizione di modelli in grado di comprendere e interpretare correttamente il linguaggio naturale, che è intrinsecamente "ambiguo"; le variazioni linguistiche, ovvero la necessità che i modelli siano in grado di gestire la diversità linguistica e di adattarsi a diversi contesti culturali e linguistici. I recenti progressi nella "Deep Learning" hanno permesso lo sviluppo di modelli che imparano dal linguaggio naturale, fanno previsioni e forniscono testi coerenti.

Esistono due fasi principali nell'elaborazione del linguaggio naturale: *preelaborazione dei dati e sviluppo di algoritmi*, che implica la preparazione e la pulizia dei dati di testo in modo che le macchine possano analizzarli. Per effettuare tale operazione possono essere utilizzati diversi approcci tra cui la ***tokenizzazione***.

La tokenizzazione svolge un ruolo cruciale in quanto stabilisce l'unità di base con cui lavorare per le analisi successive. Tale tecnica consiste nella suddivisione di un testo in unità più piccole chiamate *token*. Un token può rappresentare una singola parola, una sequenza di caratteri o anche un simbolo specifico. Si procede poi alla costruzione del dizionario cioè l'elenco dei token. La frase nell'esempio 2 è stata suddivisa in token, considerando

Figura 2: Esempio tokenizzazione

spazi e punteggiatura. La tokenizzazione può essere effettuata utilizzando

diverse strategie e strumenti. Una volta costruito il dizionario, ogni token viene convertito in un numero che corrisponde all'indice associato all'interno del dizionario. A questo punto le reti neurali sono in grado di elaborare e analizzare il linguaggio (in quanto richiedono input numerici per i calcoli).

2.3 Trasformer

I ***Trasformer***[6] sono un tipo di architettura di rete neurale che ha rivoluzionato il campo NLP, introducendo nuovi paradigmi di modellazione basati sull'attenzione. A differenza delle reti neurali ricorrenti (RNN) e delle reti neurali con memoria a lungo termine (LSTM), ampiamente utilizzate in passato, i trasformer si basano esclusivamente sull'attenzione per catturare le relazioni tra parole o elementi all'interno di una sequenza.

Il concetto fondamentale dei trasformer è la "multi-head attention", che consente alla rete di assegnare pesi diversi a diverse parti dell'input sequenziale in base alla loro idoneità alla situazione corrente. Questo permette al trasformer di affrontare le dipendenze a lungo termine e di catturare relazioni semantiche complesse all'interno della sequenza.

2.4 BERT

BERT[2] rappresenta uno dei più importanti progressi nell'elaborazione del linguaggio naturale. BERT è un modello di linguaggio basato sui transformer che è stato introdotto nel 2018 e ha avuto buoni risultati in una varietà di compiti di traduzione naturale del linguaggio (NLP), come la comprensione del linguaggio naturale, la classificazione del testo e l'analisi del sentimento. La capacità di BERT di apprendere rappresentazioni linguistiche bidirezionali attraverso un processo di addestramento non supervisionato su un corpus di testo diversificato è ciò che lo rende eccezionalmente potente. BERT tiene conto del contesto (sinistro e destro) di ogni singola parola o token, a differenza dei modelli precedenti che si basavano su informazioni unidirezionali.

BERT pratica il cosiddetto "masked language modeling" (*MLM*) durante l'addestramento. Maschera casualmente alcuni token o parole di input e cerca di prevederli in base al contesto circostante. BERT può acquisire rappresentazioni contestualizzate e approfondite delle parole attraverso questo processo di *MLM*.

Due diversi tipi di rappresentazioni di token consentono a BERT di catturare relazioni semantiche complesse: una rappresentazione di *token segment* che indica a quale sezione del testo appartiene una sequenza, come una frase o un contesto, e una rappresentazione di *token position* che indica dove si trova un token all'interno della sequenza.

Attraverso l'addestramento su dati etichettati e l'aggiunta di uno o più strati di output durante la fase di fine-tuning, BERT può essere adattato specificamente a un compito specifico.

2.5 DNABERT

Le *CNN* non sono in grado di cogliere dipendenze a lungo raggio nelle sequenze di DNA, mentre le *RNN*, anche se sono più efficaci in questo senso, hanno problemi con il vanishing gradient e sono poco efficaci. Entrambi i modelli sono difficili da trasferire e hanno difficoltà a generalizzare con pochi dati etichettati.

Per risolvere queste limitazioni, a partire dal modello BERT, è stato sviluppato un nuovo modello riadattandolo al contesto del DNA e delle sequenze genomiche, e sviluppando quindi DNABERT[4]. Gli autori hanno utilizzato la rappresentazione in k-mer per tokenizzare una sequenza di DNA invece di considerare ogni base come un singolo token. Questa rappresentazione è molto utilizzata nell'analisi delle sequenze genomiche. Ad esempio, la sequenza di DNA "ATGCT" può essere convertita in una sequenza di tre k-mer ATG, TGC, GCT e successivamente tokenizzata.

Hanno creato modelli distinti per ogni valore di k, considerando k=3,4,5,6, quindi il vocabolario contiene tutte le permutazioni di k-mer oltre a cinque token specifici: *[CLS]* classification token, *[PAD]* padding token, *[MASK]* masked token, *[SEP]* separation token e *[UNK]* unknown token.

2.6 Sequenziamento e allineamento

Il **sequenziamento** del DNA o dell'RNA è il processo di determinazione dell'ordine preciso delle basi azotate (A, C, G e T) all'interno di una molecola di DNA o di una molecola di RNA (A, C, G e U) e forniscono informazioni dettagliate sulla composizione genetica di un organismo. Esistono diverse tecniche di sequenziamento, tra cui una delle più utilizzate è la Tecnologia di sequenziamento ad alto throughput, chiamata **Next Generation Sequencing (NGS)** che, differenza del sequenziamento Sanger, consente un'analisi

massiva parallela. Questo approccio, infatti, è utile per applicazioni come il sequenziamento di interi genomi.

In particolare, genera enormi quantità di dati sequenziali noti come **reads**, porzioni sequenziate di DNA o RNA. Una read rappresenta un segmento della sequenza genomica originale, che è stato "letto" dalla macchina di sequenziamento. Le reads si differenziano principalmente in base alla loro lunghezza e al modo in cui vengono ottenute. I tre tipi principali sono **short reads, long reads e paired-end reads**.

Le *short reads* sono sequenze di DNA o RNA di lunghezza relativamente breve, tipicamente comprese tra 50 e 300 nucleotidi (bp). Sono principalmente generate da piattaforme di sequenziamento come Illumina che producono grandi quantità di reads di breve lunghezza e sono utilizzate soprattutto per il sequenziamento di genomi di piccole dimensioni.

Le *long reads* sono sequenze di DNA o RNA di lunghezza maggiore, variando da 1.000 a oltre 100.000 bp. Sono utilizzate per assemblare genomi complessi come il genoma umano.

Le *paired-end reads* sono un tipo di short read in cui entrambi i lati di un frammento di DNA sono sequenziati, fornendo due reads per ciascun frammento, registrate come coppia. Sono molto vantaggiose in quanto le informazioni provenienti da entrambe le estremità permettono di ricostruire meglio la sequenza del DNA intermedio, permettono una maggiore precisione nell'assemblaggio e di allineare reads su regioni ripetitive.

Durante l'analisi bioinformatica, le reads vengono assemblate e successivamente è svolto l'**allineamento**, il processo di confronto delle reads sequenziate con un genoma o un trascrittoma di riferimento. L'obiettivo dell'allineamento è di identificare la posizione corretta di ciascuna read all'interno del riferimento genetico, in modo da poter ricostruire l'intera sequenza ed ottenere informazioni sulla struttura e sulle funzioni dei geni.

2.7 Fusim - FUsion SIMulator

Le anomalie cromosomiche causano fusioni geniche che producono RNA chimerici (trascritti di fusione). Sono emerse nuove tecniche computazionali per la scoperta delle fusioni come risultato dei recenti sviluppi nel sequenziamento del trascrittoma. Per testare e perfezionare questi strumenti è necessario poter simulare i trascritti di fusione. Per tale motivo, è stato creato **FUSIM** *(FUsion SIMulator)* [1], un programma software per la simulazione di trascrizioni di fusione. Il programma supporta la simulazione di processi come

la traslocazione intercromosomica, il trans-splicing, i riarrangiamenti cromosomici complessi e gli eventi di read-through trascrizionale che sono noti per produrre geni di fusione e le proteine chimeriche che ne derivano. FUSIM, dunque, consente di compilare un dataset di trascritti di fusione.

Tale tool richiede i seguenti **input** per simulare efficacemente i trascritti di fusione:

- *File del modello genico:* l'input principale è un file del modello genico, che può essere in formato tabella UCSCGenePred o in General Feature Format (GFF) e Gene Transfer Format (GTF). Se si usa GFF o GTF, FUSIM ha un convertitore incorporato per il formato GenePred.

- *Genoma di riferimento:* Per generare sequenze di fusione grezze è necessario un genoma di riferimento indicizzato faidx in formato FASTA. È possibile creare un formato indicizzato faidx da un file FASTA utilizzando strumenti come SAMtools.

- *Allineamenti delle letture RNA-Seq:* Se gli utenti desiderano che la simulazione tenga conto dei livelli di espressione genica, possono fornire un file di allineamento delle letture RNA-Seq in formato BAM. Ciò consente a FUSIM di selezionare i geni per la fusione in base ai profili di espressione, seguendo metodi come la selezione uniforme o la distribuzione empirica.

Fusim produce in **output** trascritti di fusione in formato testo e FASTA. Inoltre, per simulare le letture di sequenziamento, offre wrapper che integrano strumenti esterni per produrre file in formato FASTQ, che possono essere inseriti in dataset RNA-Seq esistenti.

2.8 ART_Illumina

ART è un insieme di strumenti di simulazione che generano letture sintetiche di sequenziamento [3]. ART_Illumina è un programma di simulazione per generare dati sintetici di letture single-end, paired-end, mate-pair e simulare il sequenziamento di ampliconi della piattaforma di sequenziamento Illumina. In input richiede:

- *fasta*: un file contenente le sequenze di riferimento da cui generare le letture simulate tipicamente in formato FASTA e contenente le sequenze di DNA (o RNA convertito in cDNA) dalle quali verranno simulate le letture;

- *sequencing_system*: il nome del sistema di sequenziamento Illumina del profilo incorporato utilizzato per la simulazione;
- *fold_coverage*: la fold di copertura delle letture da simulare o il numero di letture/coppie di letture generate per ogni amplicone;
- *read_length*: la lunghezza delle letture da simulare.

ART, di default, seleziona un built-in quality score integrato in base alla lunghezza della lettura specificata per l'esecuzione. Supporta due modalità di lettura: single-end e paired-end. Nella la simulazione **single-end** genera letture singole, come in esperimenti in cui si sequenzia solo un'estremità del frammento di DNA. Tale modalità richiede il file di sequenza di input, il prefisso del file di output, la lunghezza delle letture e il conteggio delle letture. Per la simulazione **paired-end** genera letture di entrambi gli estremi di un frammento di DNA, una modalità comunemente utilizzata nei sequenziamenti Illumina, che migliora l'accuratezza nell'allineamento e nell'assemblaggio delle sequenze. Tale modalità richiede anche i valori dei parametri della media e della deviazione standard delle lunghezze dei frammenti di DNA/RNA (tranne che per il sequenziamento di ampliconi).

Genera in output file di lettura FASTQ, di allineamento ALN e/o SAM. In particolare, nel file FASTQ (*.fq.gz) sono riportate le read simulate mentre il file ALN (*.aln) è un file di allineamento delle letture simulate.

2.9 Fusion Catcher

FusionCatcher [5] è un tool open-source progettato per l'analisi dei dati di sequenziamento allo scopo di identificare fusioni geniche. In particolare effettua la ricerca di fusioni somatiche tra geni, a partire da dati di sequenziamento RNA, in questo caso reads paired-end o single-end, provenienti dall'uomo o da altri vertebrati. La maggior parte dei dati che analizza FusionCatcher sono trascritti RNA. FusionCatcher effettua delle operazioni di preprocessing, in particolare:

- rimuove le letture che si allineano su RNA ribosomiale/transfer, DNA mitocondriale, geni HLA o genomi noti di virus/fagi/batteri;
- rimuove reads che contengono adattatori e code in poli-A/C/G/T;
- ritaglia le reads in funzione dei punteggi di qualità.

- rimuove le reads che sono valutate negativamente dal sequenziatore Illumina.

Successivamente, allinea le sequenze di lettura sul genoma di riferimento utilizzando un software di allineamento. Questo processo consente di mappare le sequenze di lettura sulla loro posizione precisa nel genoma di riferimento. Completato l'allineamento, FusionCatcher analizza gli allineamenti delle sequenze di lettura per individuare potenziali fusioni geniche.

Per cercare fusioni di geni in un organismo umano, FusionCatcher ha bisogno dei seguenti parametri di input:

- *-d /some/human/data/directory/:* contiene i dati e i file generati da fusioncatcher-build;
- *-i /some/input/directory/containing/fastq/files/*: contiene il file di input FASTQ;
- *-o /some/output/directory/*: la directory di output.

In particolare, i file FASTQ devono:

- provenire da un esperimento di sequenziamento dell'RNA econtenere letture paired-end;
- i file FASTQ accoppiati devono essere sincronizzati (cioè le letture di una coppia devono trovarsi sullo stesso numero di riga in entrambi i file FASTQ);
- letture paired-end che seguono il protocollo di preparazione del campione suggerito da Illumina, cioè i due read-mates sono: (i) provenienti da filamenti opposti e (ii) in direzioni opposte l'una rispetto all'altra;
- le letture paired-end devono provenire da un protocollo di preparazione del campione stranded (cioè specifico per lo strand) o unstranded.

FusionCatcher genera in output il file *final-list_candidate_fusion_genes.txt* contenente la tabella finale con i nuovi geni di fusione candidati, in particolare la loro sequenza e i punti di giunzione, e il file *summary_candidate_fusions.txt* contiene una sintesi dei geni di fusione candidati trovati ed è destinata a essere letta direttamente dai medici o dai biologi.

3 Implementazione o Sviluppo

In questo capitolo, descriveremo il processo di sviluppo e implementazione delle varie pipeline progettate per semplificare tutte le operazioni di generazione delle reads chimeriche o meno e l'utilizzo dei vari tool. Include l'automatizzazione di alcuni processi di download e di creazione dei file utili ai vari tools (fusim, art, fusioncatcher).

Verrà anche presentato un modello fully-connected addestrato sul dataset di embeddings prodotto dalla pipeline.

Il flusso di lavoro è stato realizzato utilizzando il linguaggio Python, scelto per la sua versatilità, semplicità di esecuzione su macchine eteogenee e l'ampio supporto di librerie per l'ambito della bioinformatica.

L'implementazione di queste pipeline ha richiesto una progettazione delle fasi con l'obiettivo di garantire efficienza, in quanto i dataset generati sono di grandi dimensioni. Di seguito, verranno esplorati i componenti principali delle pipeline, illustrandone le funzionalità e le scelte implementative adottate.

3.1 Pannello dei Geni

L'intera pipeline Python realizzata fa uso di un file di testo, all'interno del quale vanno inseriti i nomi dei geni dei quali si vuole effettuare il download e/o la generazione delle fusioni. Un esempio del file è il seguente:

Listing 1: gene_panel.txt

```
RUNX1
ETV6
...
CTCF
KMT2A
```

3.2 Generazione delle reads

L'intero processo di generazione e simulazione di fusioni geniche è automatizzato da questa pipeline. Il risultato finale è una raccolta di letture simulate per ciascuna combinazione genica. Questi sono molto utili per qualsiasi tipo di studio bioinformatico, come analizzare l'espressione delle fusioni geniche in varie condizioni o testare gli algoritmi di rilevamento delle fusioni geniche.

La pipeline realizzata è composta da due moduli principali: nella prima fase vengono generate sequenze di fusioni geniche; nella seconda fase vengono simulate letture di sequenze a partire dalle fusioni prodotte precedentemente.

3.2.1 FUSIM

La funzione principale è *run_fusion()*, questa funzione prende in input due geni (gene1 e gene2) e genera una fusione tra questi utilizzando il programma JAVA (esterno) Fusim. I passaggi posso essere riassunti come segue:

- La funzione chiama **fusim.jar** con dei parametri specifici: *–gene-model* definisce il modello genico di riferimento, *–fusions* il numero di fusioni da generare, *–reference* il genoma umano di riferimento (hg19), *–fasta-output* e *–text-output* definiscono i file di output, *–cds-only* e *–auto-correct-orientation* limitano la fusione alle regioni codificanti (CDS) e correggono automaticamente l'orientamento;

- Il programma legge i nomi dei geni dal file *genes_panel.txt*, si calcolano tutte le combinazioni possibili tra i geni (ogni gene può combinarsi con tutti gli altri, incluso se stesso). Si utilizza ThreadPoolExecutor per eseguire più fusioni in parallelo. L'output di questa fase è costituito da file *.fasta* e *.txt* per ogni coppia di geni, archiviati in cartelle specifiche.

3.2.2 ART-ILLUMINA

La funzione principale è *run_art_illumina()*, questa funzione riceve come input il nome di un file *.fasta*, simula letture "Illumina" e le salva in file di output, tutto questo utilizzando il programma (esterno) **ART-Illumina**.

Il programma viene chiamato con dei parametri specifici di simulazione: *-l* lunghezza delle letture a 150 bp, *-f 10*: copertura pari a 10X. *-p*: letture in modalità *paired-end*, *-m 400* e *-s 10* media e deviazione standard sono impostate rispettivamente a 400 e 10.

Il main infine si occupa di chiamare la funzione di run su tutti i file *.fasta* contenuti nella folder specificata.

3.3 Creazione del Dataset delle fusioni

Questa pipeline è stata progettata per simulare fusioni geniche e creare sequenze che rappresentano i punti di fusione. Utilizza come strumento Fusim, visto nel capitolo precedente §3.2, per generare delle reads, si procederà poi

con un processo di etichettatura per la classificazione (es. chimeriche o non chimeriche). Tale pipeline facilita il lavoro e risulta molto importante nel momento in cui si vogliono realizzare numerose sperimentazioni su reads chimeriche o non chimeriche, addestramento di modelli di "predizione del gene", oppure modelli di "predizione dei geni fusi".

Le componenti della pipeline sono 3: le prime due componenti riguardano Fusim, mentre la terza parte riguarda quella che è la generazione del dataset utilizzando pandas.

La funzione più importante è *get_sequences_from_fusim_fasta()*, riceve come parametro un file FUSIM e lo converte in una lista di sequenze, facilitando quelle che saranno le future manipolazioni dei dati.

Per effettuare la creazione del DataFrame è stato necessario effettuare l'analisi e la manipolazione delle informazioni presenti all'interno dei file .txt generati da fusim, questa operazione è necessaria per ottenere il gene di fusione e le basi dell'esone.

Le informazioni sono state poi convertite in formato tabellare, rendendole adatte all'importazione in un DataFrame, che è poi usato per creare un dataset .csv. Ogni riga del dataset contiene quindi il gene di fusione, le basi dell'esone (caratterizza il punto esatto di giunzione tra i due geni) e la sequenza completa.

Listing 2: Dataset Fusion-Gene.csv

```
FUSION_GENE,    EXON_BASE,  SEQUENCE
TAL1-MEF2D,     312,        CGGGTTCTTT...
PBX1-PMEL,      174,        ATGAGGCGCA...
...,            ...,        ...
ETV6-PAX5,      67,         ATGTCTGAGA...
```

3.4 Creazione Dataset Embeddings e addestramento

In questa sezione, descriviamo in dettaglio la pipeline utilizzata per preparare i dati a partire dai file di reads fino alla generazione degli embeddings, che verranno infine utilizzati come input per un modello di apprendimento automatico basato su BERT.

3.4.1 Conversione file di Reads in DataFrame

Il primo passaggio della pipeline consiste nella lettura e conversione dei dati di reads in un formato più facilmente manipolabile. Partendo da un file `.aln` che contiene le reads, utilizziamo la funzione denominata `art_to_dataframe(art_filepath)`, che riceve il file in input e lo converte in un DataFrame di pandas. Per effettuare la creazione del DataFrame è stato necessario effettuare l'analisi e la manipolazione dell'intestazione ART.

Questo DataFrame rappresenta una struttura tabellare in cui ogni riga contiene le informazioni dettagliate di ciascuna read, facilitando le operazioni di manipolazione e filtraggio necessarie per l'elaborazione dei dati.

La funzione `art_to_dataframe` genera un DataFrame con le seguenti colonne:

- `seqId`: Identificatore univoco della sequenza;
- `seqCount`: Numero di volte che la sequenza compare nel file;
- `start`: Posizione di inizio della sequenza;
- `strand`: Informazione relativa allo strand (filamento) della sequenza;
- `read1` e `read2`: Sequenze delle reads.

3.4.2 Identificazione del Punto di Fusione

Successivamente, viene aperto un file di tipo `.txt` generato con FUSIM, il cui scopo è fornire informazioni sul punto di fusione noto come *exonbase*. Questo dato permette di etichettare ogni read in base alla sua appartenenza ad una fusione genetica, distinguendo tra reads chimeriche e non chimeriche. L'etichettatura delle reads rappresenta la base per il training di modelli di classificazione.

3.4.3 Creazione del Dataset Finale

Avendo a disposizione i dati elaborati nei passaggi precedenti, procediamo alla creazione del dataset finale, che include le seguenti informazioni per ciascuna read:

- `read`: Sequenza della read;

- `fusionGenes`: Nomi dei geni di fusione associati;
- `label`: Etichetta binaria che indica se la read è chimerica (`1`) o non chimerica (`0`).

Il dataset risultante viene quindi salvato in un file `.csv`.

3.4.4 "K-Merizzazione" del Dataset

Una volta ottenuto il dataset `.csv`, si procede con un'operazione di -*k-merizzazione* delle sequenze. La k-merizzazione consiste nella suddivisione delle sequenze in *k-mer*, ovvero sottosequenze di lunghezza fissata `k`, che in questo caso è impostata a 4. Questo permette di rappresentare ogni read come una sequenza di parole di lunghezza 4, agevolando l'analisi testuale con modelli di NLP.

Per ciascuna read nel dataset, la sequenza viene scomposta in k-mer consecutivi, generando 145 parole di lunghezza 4. Le sequenze k-merizzate sono quindi associate alla loro etichetta di appartenenza (chimerica o non chimerica), e vengono utilizzate come input per i passaggi successivi della pipeline.

Listing 3: Esempio K-merizzazione

```
sequenza iniziale:
GGCTGGCTGTGGCTGGCTGT

k-mer:
GGCT GCTG CTGG TGGC GGCT GCTG CTGT ... CTGT
```

3.4.5 Estrazione embeddings BERT

Le sequenze k-merizzate di ciascuna read vengono trasformate in una riga composta da 145 parole, che rappresenta l'input per il modello BERT. Il modello BERT §2.4 viene utilizzato per generare embeddings a partire dalle sequenze genomiche.

Ogni sequenza k-merizzata di 145 parole viene inviata a BERT per poter poi effettuare l'estrazione degli embeddings, che rappresentano una mappatura delle parole in uno spazio numerico multidimensionale. Questo spazio permette di catturare le relazioni semantiche tra le sequenze, facilitando il compito di classificazione delle reads.

Il risultato dell'estrazione degli embeddings dell'elaborazione con BERT è un file `.csv` in cui ogni riga contiene:

- Una lista contenente embedding sotto forma di vettori numerici.
- L'etichetta binaria associata (`0` o `1`), che indica se la sequenza è chimerica.

Listing 4: Dataset embeddings.csv

```
EMBEDDINGS,                                        LABEL
[[5325, 7764, ... , 4365],[3245, ... , 5981]],  0
...
[[4765, 1435, ... , 9034],[6650, ... , 3209]],  1
```

3.4.6 Output Finale: Dataset degli Embeddings

Il dataset finale degli embeddings, contenente le rappresentazioni numeriche delle reads e le rispettive etichette, costituisce la base per il training di modelli di apprendimento automatico avanzati. Grazie alla pipeline implementata, abbiamo quindi un dataset pronto per l'addestramento, dove ogni sequenza è rappresentata da un embedding derivato da BERT e associato all'informazione di classificazione, che permette di distinguere tra reads chimeriche e non chimeriche.

3.4.7 Addestramento del Modello RNN

Per il compito di classificazione delle reads, è stato implementato un modello di machine learning basato su RNN (Recurrent Neural Network). L'input layer del modello RNN utilizza il dataset di embeddings prodotto nella pipeline descritta in precedenza, in cui le sequenze genomiche sono rappresentate come vettori numerici, e ogni read è associata a un'etichetta binaria.

Prima di procedere all'addestramento, il dataset è stato diviso in due insiemi: uno per il *training* e uno per il *testing*. Questa suddivisione è fondamentale per valutare le prestazioni del modello in termini di accuratezza e generalizzazione. Durante l'addestramento, la colonna `EMBEDDINGS` è stata utilizzata come feature di input, mentre la colonna `LABEL` (con valori `0` e `1`, rispettivamente per reads non chimeriche e chimeriche) è stata utilizzata come target.

Il modello RNN, grazie alla sua capacità di apprendere dipendenze sequenziali e di gestire dati in sequenza, è particolarmente adatto per analizzare pattern all'interno delle sequenze genomiche e per migliorare la classificazione delle reads. Al termine dell'addestramento, il modello è stato valutato sul set di test per determinare la sua efficacia nel distinguere tra reads chimeriche e non chimeriche.

3.5 Creazione del Dataset "Trigrams"

Questa pipeline è stata progettata per ottenere come output un dataset costituito da sequenze consecutive di k-mer derivanti da una sequenza di un gene specificato. All'interno di questa pipeline viene utilzzato il tool esterno ***gt_shredder*** per trasformare file FASTQ di sequenze in file *.reads* con frammenti di sequenze di lunghezza fissata a 150 bp. Il codice sviluppato esegue una pipeline di pre-elaborazione su sequenze genetiche per renderle compatibili con modelli NLP (Natural Language Processing) utilizzando k-mer e *trigrams*, tecniche comunemente utilizzate per la rappresentazione di sequenze di DNA. La pipeline si articola in più passaggi:

- Frammentazione dei file FASTQ: con *gt_shredder_df()*, vengono letti i file .FASTQ contenenti sequenze di DNA, questi verranno frammentati in sequenze di lunghezza definita (150 bp) utilizzando il tool esterno *gt_shredder* in modo da agevolare le operazioni successive;

- Conversione delle sequenze ottenute in k-mer: la funzione *seq2kmer()*, utilizzata a sua volta dalla funzione *kmerize_seq()* trasforma ogni sequenza di DNA in una lista di k-mer, ovvero sottosequenze consecutive di lunghezza k, per rendere le sequenze rappresentabili come "parole" che il modello NLP possa interpretare;

- Creazione di file k-mer: il codice elabora ogni frammento per generare k-mer che vengono salvati come file separati nella directory designata; la funzione *create_kmers()* si occupa di questo passaggio.

- Costruzione del DataFrame: la funzione *set_df_kmers()* crea un DataFrame in cui ogni riga rappresenta una sequenza di k-mer associata a una specifica etichetta di classe, corrispondente al gene di origine, facilitando così la gestione e la manipolazione dei dati per la fasi successive.

- Costruzione di "trigrams" e salvataggio: infine, con *create_tr_dataset()*, il codice frammenta ogni sequenza di (3) k-mer in "trigrams", creando appunto "frasi" di tre k-mer consecutivi che, associate all'etichetta di classe, vengo-

no salvate in un file CSV per essere utilizzate nei vari modelli di intelligenza artificiale.

Questo approccio consente una **rappresentazione strutturata** delle sequenze di DNA, facilitando l'identificazione di pattern rilevanti per la classificazione di fusioni geniche.

3.6 Docker Fusion-Catcher

È stato realizzato un container Docker contenente FusionCatcher, uno strumento della bioinformatica. L'immagine Docker è stata caricata all'interno dell'hub docker in modo da essere resa disponibile online. È stata configurata per includere FusionCatcher già "installato" e pronto all'uso, completo di tutte le dipendenze necessarie e in un ambiente operativo preconfigurato.

Questo metodo rende lo strumento molto più semplice da utilizzare, eliminando la necessità di installazioni o configurazioni complicate che devono essere fatte manualmente e consentendo un'esecuzione replicabile e immediata su qualsiasi sistema dotato di Docker. Inoltre, rendende più accessibile l'analisi delle fusioni geniche, l'uso di un container Docker garantisce la portabilità e la riproducibilità.

4 Conclusioni

Questo progetto ha introdotto una pipeline automatizzata per la generazione di letture chimeriche sintetiche, con l'obiettivo di facilitare l'analisi delle fusioni geniche e migliorare l'efficienza dei flussi di lavoro bioinformatici. La pipeline, attraverso l'integrazione di strumenti come *Fusim* e *ART-Illumina* e l'utilizzo di tecniche avanzate di *k-merizzazione* e tokenizzazione, si è dimostrata efficace nella produzione di dataset, pronti per l'addestramento e il benchmarking di modelli di intelligenza artificiale.

4.1 Risultati Ottenuti

1. **Ottimizzazione della creazione di dataset**: La pipeline ha semplificato e automatizzato la generazione di letture chimeriche e non chimeriche, riducendo i tempi e le risorse computazionali richiesti rispetto ai metodi manuali. I dataset prodotti sono risultati adatti per

il benchmarking di strumenti bioinformatici e per l'addestramento di modelli di machine learning.

2. **Preparazione efficace di dati per modelli IA**: Utilizzando metodi di tokenizzazione e rappresentazione con "k-mer" e "trigrammi", la pipeline ha permesso la conversione delle sequenze in formati compatibili con modelli IA. Gli embeddings generati tramite BERT hanno fornito rappresentazioni numeriche delle sequenze genomiche, dimostrando utilità per compiti di classificazione e predizione.

3. **Sviluppo di un modello RNN per la classificazione delle letture**: È stato realizzato un modello RNN ed è stato addestrato sui dataset di embeddings ottenuti dalle letture chimeriche e non chimeriche.

4.2 Sviluppi Futuri

Per proseguire il lavoro, si identificano diversi potenziali sviluppi:

1. **Integrazione di modelli di deep learning più avanzati**: Esplorare l'uso di modelli transformer di ultima generazione, come DNABERT, che sono specificamente ottimizzati per le sequenze di DNA, potrebbe migliorare ulteriormente le prestazioni di classificazione e predizione.

2. **Realizzazione di un DNABERT custom**: Esplorare la possibilità di realizzare un modello di DNABERT personalizzato, per migliorare le prestazioni dei modelli e per fornire degli embeddings più precisi.

In sintesi, i risultati di questo progetto dimostrano l'efficacia di un approccio automatizzato e strutturato per la generazione e l'analisi delle letture chimeriche, offrendo strumenti e metodologie per sostenere i progressi nel campo della genomica computazionale e delle applicazioni IA in bioinformatica. La pipeline presentata rappresenta un passo avanti nell'ottimizzazione delle risorse bioinformatiche, con significativi benefici per la ricerca scientifica.

Riferimenti bibliografici

[1] Andrew E Bruno et al. «FUSIM: a software tool for simulating fusion transcripts». In: *BMC Bioinformatics* 14.1 (2013), pp. 1–13. DOI: 10.1186/1471-2105-14-13.

[2] Jacob Devlin et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: *arXiv preprint arXiv:1810.04805* (2018). URL: https://arxiv.org/abs/1810.04805.

[3] Weifeng Huang et al. *ART: A Next-Generation Sequencing Read Simulator*. Version 2.5.8, https://www.niehs.nih.gov/research/resources/software/biostatistics/art/index.cfm. 2012.

[4] Yanrong Ji et al. «DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome». In: *Bioinformatics* 37.15 (2021), pp. 2112–2120. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab083. eprint: https://academic.oup.com/bioinformatics/article-pdf/37/15/2112/57195892/btab083.pdf. URL: https://doi.org/10.1093/bioinformatics/btab083.

[5] Daniela Nicorici et al. «FusionCatcher: A Tool for Finding Somatic Fusion Genes in Paired-End RNA-Sequencing Data». In: *Bioinformatics* 30.12 (2014), pp. 1774–1780. DOI: 10.1093/bioinformatics/btu121.

[6] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.