Sommario

Negli ultimi anni l'introduzione delle tecnologie per il sequenziamento dell'intero genoma di organismi biologici ha rivoluzionato l'approccio allo studio delle funzionalità biologiche degli esseri viventi. Il problema si è spostato da come generare i dati a come gestire e sfruttare la mole di informazioni prodotta. Una delle soluzioni proposte è il concetto di annotazione, cioè l'associazione di informazioni utili a sequenze molecolari che descrivono la conoscenza disponibile delle caratteristiche strutturali e funzionali di tali sequenze. L'interpretazione dei moderni esperimenti di biologia molecolare dipende fortemente dalla consistenza e completezza delle annotazioni esistenti. Le annotazioni sono inserite in banche dati da degli esperti di settore che analizzano la letteratura disponibile, attività lenta e soggetta ad errori; pertanto tali banche dati sono per definizione incomplete e, a volte, incorrette. Per arginare questo problema, la creazione di strumenti informatici per facilitare e velocizzare il processo di inserimento di nuove annotazioni risulta importante e molto utile.

Alcuni algoritmi di predizione sono stati proposti in letteratura, ma essi presentano limiti relativi ai tipi di dato utilizzabili, alle prestazioni, all'affidabilità e alla mancanza di meccanismi che li rendano pienamente automatizzati. Altri strumenti invece utilizzano dei modelli matematici molto generali, mutuati dagli ambiti di Elaborazione del Linguaggio Naturale e dell'analisi semantica di documenti. Ne è un esempio *AnnotationPredictor* sviluppato nel Gruppo di Basi di Dati e Bioinformatica del Politecnico di Milano, un applicativo per la predizione di annotazioni basato su tecniche di analisi semantica vettoriale. Nella presente tesi intendo descrivere il lavoro che ho svolto per estenderne l'architettura con ulteriori algoritmi, illustrando le modifiche architetturali implementate per permette l'integrazione dell'algoritmo *pLSA* e delle funzioni di peso *tf-idf*. Procederò quindi alla valutazione delle prestazioni delle nuove soluzioni proposte.

I risultati ottenuti mostrano la validità dei nuovi metodi ma evidenziano anche problematiche legate all'onere computazionale.

1 Introduzione

Nel proseguo del presente capitolo introdurrò il contesto in cui è stata elaborata questa tesi e le principali definizioni necessarie ad una corretta interpretazione dell'elaborato; descriverò i tipi di dato utilizzati e le relative problematiche. Infine darò una indicazione di come è organizzata la presentazione del lavoro.

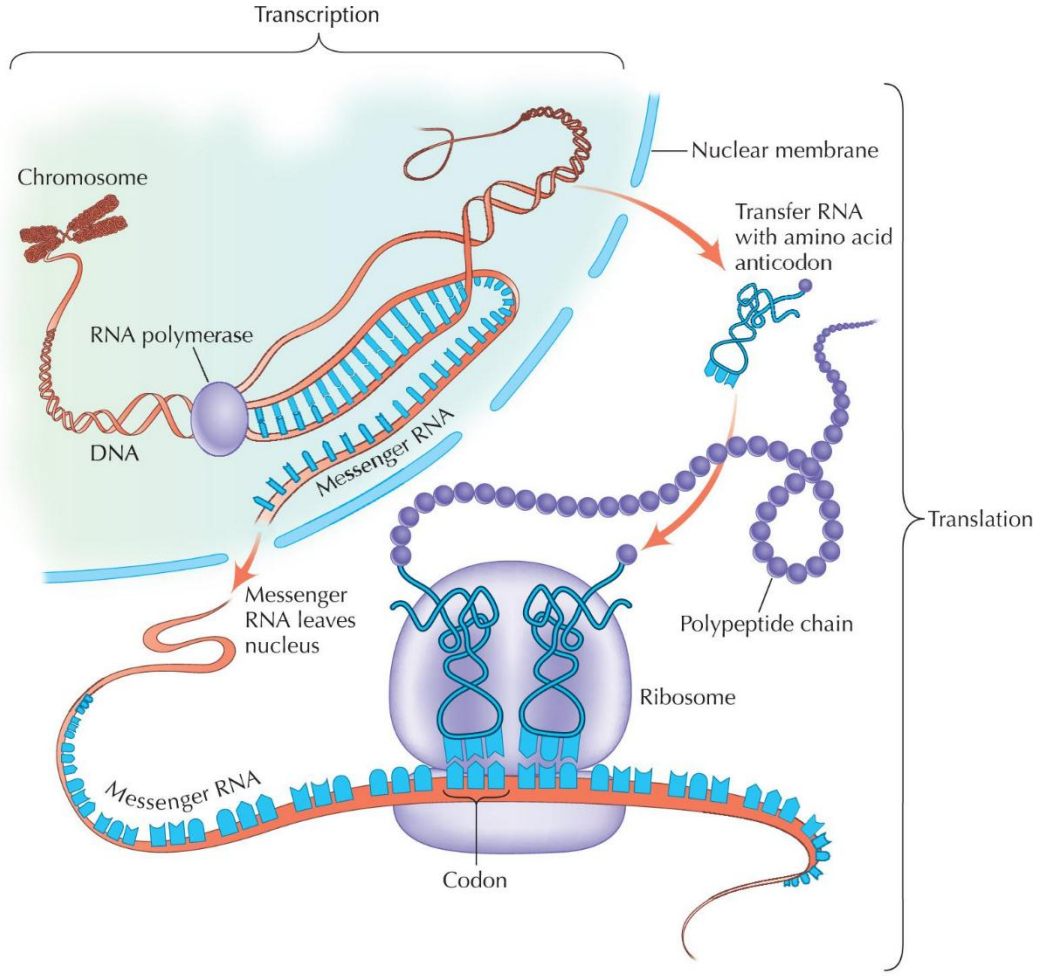
1.1 Geni e proteine

Il DNA (acido desossiribonucleico) è una molecola presente in quasi tutte le cellule di un organismo biologico che, con la sua struttura, codifica tutte le informazioni necessarie per lo sviluppo e il funzionamento di un essere vivente. Nel 1865 lo scienziato ceco Gregor Mendel scoprì che i tratti individuali di ogni essere vivente sono determinati da fattori discreti, che più tardi furono chiamati geni, che sono ereditati dai genitori; ma solamente nel 1953 fu isolata per la prima volta da James Watson e Francis Crick la molecola del DNA e fu scoperta la sua ben nota struttura a doppia elica. Tale struttura codifica tutte le informazioni genetiche di un individuo – il suo genoma - come una sequenza di quattro diverse basi azotate su una struttura molecolare di zucchero: il desossiribosio. La sequenza delle quattro basi azotate, (A) Adenina (T) Timina (G) Guanina e (C) Citosina, costituisce il codice genetico. Ogni base azotata si chiama anche nucleotide.

Possiamo paragonare il DNA ad un grande libro, nel quale sono scritte tutte le informazioni che compongono il patrimonio genetico di ogni essere vivente. Ogni base azotata è paragonabile ad una lettera dell'alfabeto con cui è scritto questo libro e ogni tripletta di basi azotate, chiamata codone, è paragonabile ad una parola; ciascuna di queste "parole" codifica un aminoacido, l'elemento base costitutivo di ogni proteina. Le combinazioni possibili di triplette sono in totale sessantaquattro le quali sono tradotte in venti aminoacidi, quindi più triplette possono codificare lo stesso aminoacido. I geni sono costituiti da lunghe sequenze di nucleotidi e possiamo paragonarli alle frasi del codice genetico. L'informazione codificata da ogni gene viene interpretata all'interno dei ribosomi e utilizzata per la sintesi di una proteina. Le proteine sono i

cosiddetti “mattoni della vita”, sono composte di sequenze di aminoacidi e costituiscono l’elemento base di ogni struttura biologica vegetale o animale.

Nelle cellule umane, il DNA è organizzato e compresso in ventitré paia di cromosomi, dove ogni membro di ogni paio è ereditato da ogni genitore. La maggior parte del DNA (il 98,5%) è considerata “inutile”, o meglio la sua funzione è sconosciuta, mentre la parte rimanente costituisce i geni, coinvolti nel processo di trascrizione, cioè nel trasferimento delle informazioni dal DNA nel nucleo della cellula, ai ribosomi per la sintesi proteica. Le sequenze di basi che codificano informazioni (gli esoni), sono inframmezzate da pezzi di DNA che non codificano nulla, gli



introni, la c

Figura 1: Schema della sintesi proteica; sono rappresentate le due fasi di trascrizione e traduzione

cellule, simile al DNA è l’RNA (l’acido ribonucleico) che differisce dal DNA in quanto ha una

struttura a nastro singolo e la base azotata Timina è sostituita dall'Uracile. Tale acido nucleico serve per trasportare l'informazione contenuta nel DNA (che rimane sempre all'interno del nucleo della cellula) fino ai ribosomi, piccoli organuli situati nel citoplasma della cellula.Il passaggio dall'RNA nucleare all'RNA messaggero (mRNA) è ottenuto concatenando solamente gli esoni (cioè le parti che codificano informazioni). In pratica l'RNA serve per copiare le informazioni codificate sul DNA e trasportarle ai ribosomi, dove avviene la sintesi proteica (Figura 1).

1.2 La ricerca genomica e proteomica

Il funzionamento della sintesi proteica in precedenza descritto non è esente da problemi ed errori. Le informazioni contenute nei cromosomi possono essere alterate da condizioni ambientali avverse, quali eccessivo calore, radiazioni elettromagnetiche di elevata intensità, ecc., oppure possono subire mutazioni spontanee. Anche i processi di trasporto e trascrizione, dal nucleo ai ribosomi, sono soggetti ad errori, come qualunque messaggio che viaggia su di un canale affetto da rumore. Tutti questi errori possono causare problemi alle cellule, malattie all'organismo, e nei casi più gravi possono portare alla morte dell'individuo.

Per fronteggiare questo tipo di malattie è nata la ricerca genetica, una scienza che si occupa di studiare il comportamento delle cellule, la loro esposizione a rischi di malattie genetiche e la loro reazione a determinati interventi farmacologici. Questa scienza, come molte altre nate in epoche recenti, è fortemente interdisciplinare e coinvolge direttamente altre scienze più consolidate come la biologia, la medicina, la chimica, l'elettronica, la statistica e l'informatica. Quest'ultima si propone di fornire strumenti e metodi di analisi, indispensabili per orientarsi in un ambito che attualmente contempla enormi quantità di dati sperimentali e informazioni. Per guidare le condizioni in cui gli esperimenti sono stati effettuati e per aiutare lo scienziato a valutarne i risultati sono stati creati strumenti efficaci ed efficienti in grado di organizzare la conoscenza biologica. Tra questi strumenti ci sono i vocabolari controllati, le ontologie e le banche date di annotazioni funzionali.

1.3 Vocabolari controllati, ontologie e annotazioni funzionali

Nel proseguimento della corrente tesi tenderò a limitare i termini quali gene o proteina, in quando gli argomenti trattati varranno per entrambi gli oggetti; preferirò quindi l'espressione "entità biomolecolare" per includere entrambi i termini in un unico soggetto.

Un "vocabolario controllato" fornisce una metodologia per organizzare la conoscenza in maniera ordinata, facilmente leggibile e preferibilmente in modo univoco affinché sia utilizzabile semplicemente e rapidamente da un calcolatore elettronico. Un vocabolario controllato è composto di un insieme di termini preselezionati e autorizzati dal progettista dello stesso; ciò è in contrasto con il linguaggio naturale in quanto in quest'ultimo non vi sono restrizioni sul vocabolario utilizzato e quindi è resa possibile la presenza omografie, sinonimi e polisemie; tali fattori sono fonte di ambiguità e per questo motivo si sceglie di non permettere la loro presenza in un vocabolario controllato.

Ogni termine rappresentato in un vocabolario controllato è generalmente identificato univocamente da un codice alfanumerico e rappresenta un particolare concetto o caratteristica.

Un esempio di vocabolario controllato è il Medical Subject Heading (MeSH); ideato con l'obiettivo di indicizzare la letteratura scientifica in ambito biomedico, MeSH è composto da oltre 24.000 termini organizzati gerarchicamente dal più generale al più specifico. Un documento è indicizzato con 10-15 termini di cui solo uno o due di essi sono principali; il ricorso ad un vocabolario controllato migliora il risultato delle operazioni di ricerca poiché riduce il rumore che si avrebbe se si utilizzassero solamente le parole libere del linguaggio comune.

In ambito biomedico i vocabolari controllati sono ampiamente utilizzati anche per memorizzare le annotazioni funzionali genomiche, cioè delle relazioni tra una entità biomolecolare e uno o più concetti. Molte banche dati contengono questo tipo di informazioni, possiamo dividerle in due classi:

- Vocabolari controllati *flat* o terminologie: i termini del vocabolario non sono strutturati ne collegati tra loro tramite relazioni;
- Vocabolari controllati strutturati o ontologie: i termini sono organizzati gerarchicamente dal più generale al più specifico tramite relazioni caratterizzate da un particolare significato semantico. Le ontologie sono supportate da una buona quantità di strumenti informatici per il loro utilizzo.

Negli ultimi anni le ontologie stanno prendendo il sopravvento sui vocabolari *flat* e molti di essi sono stati tradotti in ontologie. Resistono comunque ancora alcuni vocabolari controllati, tra questi vanno citati OMIM (Online Mendelian Inheritance in Man) che cataloga le patologie aventi una componente genetica e Reactome, una raccolta di *pathway* biologiche (insieme delle reazioni chimiche coinvolte in un processo all'interno di una cellula).
Per quanto riguarda le ontologie, oltre a GO (Gene Ontology) che verrà descritta nel seguito, possiamo citare ChEBI (Chemical Entities of Biological Interest) concentrata sui componenti chimici, SBO (System Biology Ontology) concentrata sulla modellazione computazionale in ambito biologico oltre all'ontologia MeSH descritta precendetemente.
I vocabolari presi singolarmente hanno poco significato intrinseco e le ontologie, fino a pochi anni fa, potevano essere considerate degli esercizi di stile. Negli ultimi anni si è assistito ad un'inversione di mentalità. Sempre più organizzazioni si concentrano sui vocabolari controllati, i quali termini vengono utilizzati per descrivere la conoscenza riguardanti le entità biomolecolari.

1.3.1 Il concetto di annotazione

Una annotazione è una associazione tra una sequenza di nucleotidi o aminoacidi e delle informazioni utili; molti tipi di annotazioni sono usate nella pratica. Nel corso di questa tesi ci si riferirà ad annotazioni funzionali, cioè annotazioni tra una entità biomolecolare e una caratteristica.

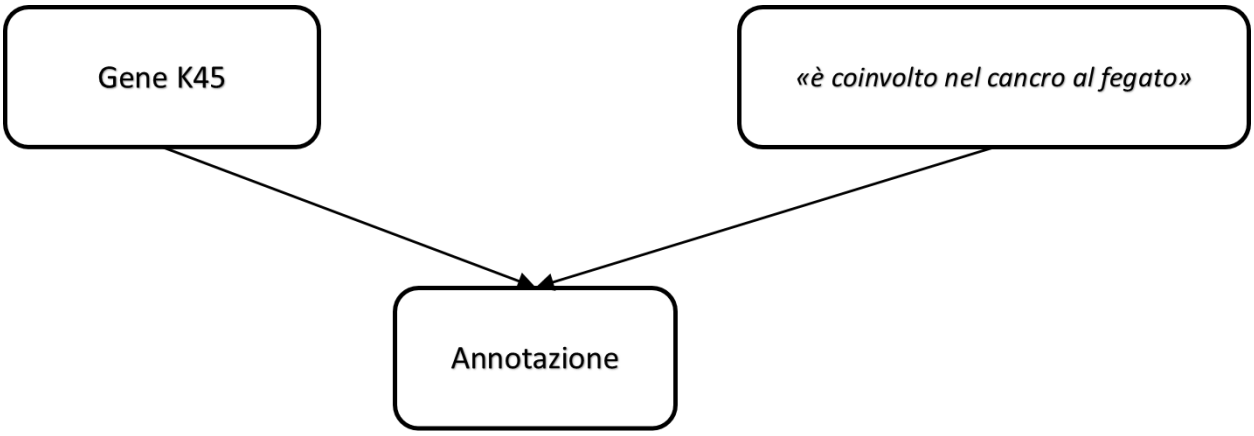


Figura 2: esempio di annotazione funzionale tra un gene (K45) e la caratteristica "è coinvolto nel cancro al fegato"

1.4 Banche dati biomolecolari

Al giorno d'oggi molti ci sono molti gruppi privati o pubblici che sequenziano il genoma di vari esseri viventi; inoltre i nuovi metodi di sequenziamento ed analisi permettono di produrre una quantità enorme di dati. Questa mole di informazione viene memorizzata e resa disponibile in database eterogenei e distribuiti su scala planetaria (*Figura 2*).

L'informazione contenuta in questi database è eterogenea; possiamo dividere queste banche dati in due categorie:

- Primarie: contengono le sequenze biomolecolari (DNA, proteine, carboidrati, strutture tridimensionali, ecc.); le informazioni contenute sono generiche ;
- Derivate: contengono gli stessi dati delle banche dati primarie ma rivisitati ed annotati con ulteriori funzionalità. Ci sono molte possibilità riguardo al tipo di informazioni aggiuntive che esse possono contenere e alla loro provenienza: curate da un esperto, inferte computazionalmente o una combinazione delle due precedenti.

Inoltre bisogna distinguere queste banche dati anche sulla base dei metodi di accesso che rendono disponibili; pagine HTML, FTP o, più raramente, accesso diretto.

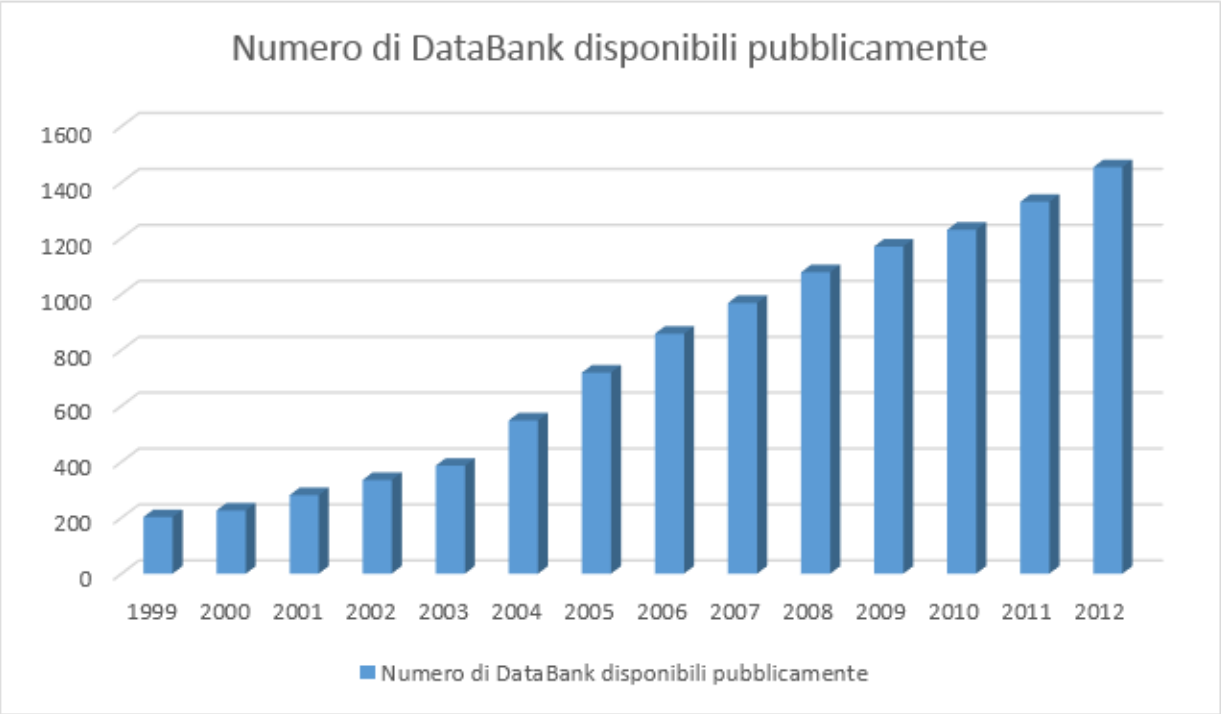


Figura 3: Crescita delle banche dati di dati biomolecolari disponibili pubblicamente online

Un grande problema di questa tecnologia è che le informazioni sono contenute su banche dati distinte, non eterogene tra di loro: si rende quindi necessario lo sviluppo di strumenti per l'integrazione dell'informazione. Un esempio è fornito dal database integrato GPDW (Genomic and Proteomic Data Warehouse) [1], realizzato al Politecnico di Milano, da cui sono estratti i dati utilizzati in questo elaborato.

1.5 Il Progetto Gene Ontology

Gene Ontology è un progetto bioinformatico molto importante e molto usato; si prefigge lo scopo di mantenere un vocabolario controllato organizzato ontologicamente ed indipendente dalla specie, di annotare geni e prodotti di geni ai termini del vocabolario e di fornire strumenti che permettano l'accesso all'informazione rappresentata.

Il GO Consortium fornisce tre differenti vocabolari controllati ontologici per descrivere attributi delle entità biomolecolari: l'ontologia Cellular Component (*componente cellulare*), composta da 3.050 termini, descrive la locazione delle entità molecolari a livello di strutture subcellulari o complessi macromolecolari; l'ontologia Molecular Function (*funzione molecolare*), composta da

9.460 termini, rappresenta le attività che un prodotto genico compie o le sue "abiltà" come ad esempio legarsi o trasportare qualcosa oppure trasformare una cosa in un'altra; Biological Process (*processo biologico*) con circa 23.900 termini è l'ontologia più grande e descrive i processi cellulari, cioè una serie di eventi e di funzioni molecolari con un ben definito stato iniziale e finale. Le tre ontologie sono state sviluppate al fine di catturare aspetti ortogonali di entità biomolecolari, in altre parole ognuna tenta di rappresentare informazioni appartenenti a domini di conoscenza diversi.

Ogni ontologia è strutturata come un grafo diretto aciclico (*directed acyclic graph, DAG*) di termini relazionati tra loro tramite associazioni di partizione (*part_of*), associazioni di ereditarietà (*is_a*) o relazioni funzionali (*regulates*). La struttura dell'ontologia è simile a quella di un albero, infatti non possono esserci cicli di relazioni tra i termini, ma a differenza di quello che succede in un albero un nodo può avere più di un padre e quindi anche essere contemporaneamente su più livelli diversi (*Figura 3*).

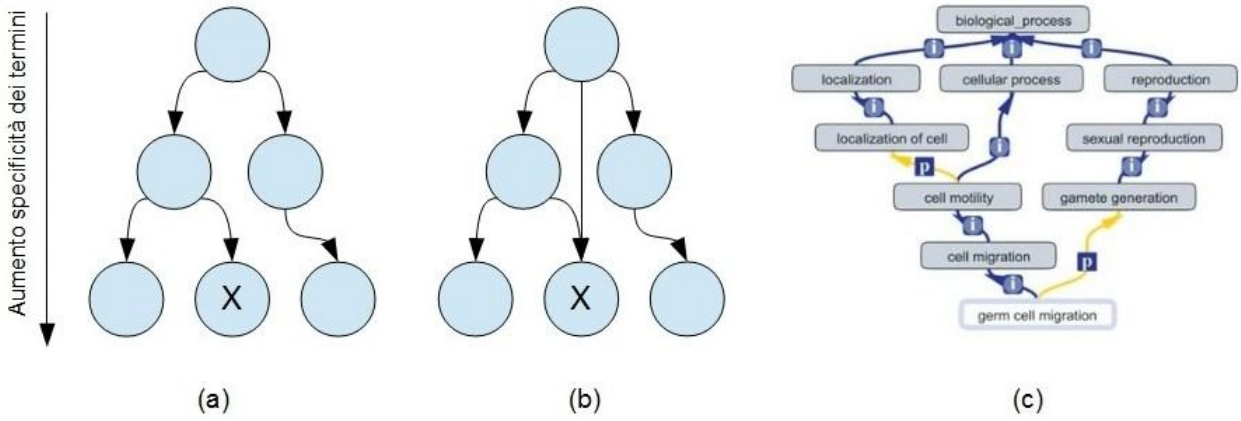


Figura 4: (a) un albero; (b) un DAG di una ontologia, rispetto all'albero si può notare come il nodo X abbia due padri e sia contemporaneamente al secondo e terzo livello di profondità; (c) un esempio di DAG di GO, ontologia Biological Process

Oltre alla struttura del vocabolario controllato il GO Consortium offre pubblicamente annotazioni a entità biomolecolari di cinquanta organismi tra animali, piante o funghi.

Una annotazione è una associazione tra un gene o un suo prodotto ed un termine della ontologia arricchita con ulteriori informazioni, come riportato in *Tabella 1*.

Tabella 1: Campi di una annotazione GO riferiti allo standard GAF 2.0

Column	Content	Required?	Cardinality	Example
1	DB	**required**	1	UniProtKB
2	DB Object ID	**required**	1	P12345
3	DB Object Symbol	**required**	1	PHO3
4	Qualifier	optional	≥ 0	NOT
5	GO ID	**required**	1	GO:0003993
6	DB:Reference	**required**	≥ 1	PMID:2676709
7	Evidence Code	**required**	1	IMP
8	With (or) From	optional	≥ 0	GO:0000346
9	Aspect	**required**	1	F
10	DB Object Name	optional	0 or 1	Toll-like receptor 4
11	DB Object Synonym	optional	≥ 0	hToll
12	DB Object Type	**required**	1	Protein
13	Taxon	**required**	1 or 2	Taxon:9606
14	Date	**required**	1	20090118
15	Assigned By	**required**	1	SGD
16	Annotation Extension	optional	≥ 0	Part_of(CL:0000576)
17	Gene Product Form ID	optional	0 or 1	UniProtKB:P12345-2

Le colonne 1, 2 e 3 identificano univocamente un gene o un suo prodotto mentre i campi 10, 11 e 12 forniscono ulteriori informazioni sulla entità biomolecolare coinvolta nella annotazione. Il campo 13 (*tassonomia*) specifica a quale organismo il gene appartenga.

La colonna 4, (*Qualifier*) è particolarmente importante, infatti cambia il modo in cui un annotazione deve essere interpretata; può assumere uno o più dei seguenti valori:

- NOT: usato per indicare esplicitamente che l'entità biomolecolare non è annotata con il termine specificato. Questo è particolarmente importante quando l'associazione entità biomolecolare-termine dovrebbe essere evitata; Per esempio, una proteina avente similarità nella sua sequenza amminica con un enzima ma è dimostrabile sperimentalmente

che tale proteina non ha attività enzimatica, allora è possibile aggiungere un'annotazione con tale flag per specificare che l'informazione non è effettivamente associata ad essa.

- *Colocalizes_with*: utilizzato solamente con termini appartenenti all'ontologia *Cellular Component*, è usato per esprimere che l'informazione potrebbe essere transiente nel ciclo di vita di un'entità biomolecolare
- *Contributes_to*: utilizzato solo con termini appartenenti all'ontologia *Molecular Function*, esprime che l'entità biomolecolare entra a far parte di una funzione attribuibile ad un complesso.

Un altro *flag* rilevante è il 7 (*Evidence Code*) che rappresenta l'evidenza scientifica con la quale una annotazione è stata introdotta nel database. Gli *Evidence Code* possono essere suddivisi in quattro categorie:

- Sperimentali: sono ritenute essere le annotazioni con maggiore qualità in quanto derivano dalla presenza di pubblicazioni che supportano la caratterizzazione dell'entità biomolecolare. In tale categoria ricadono i seguenti evidence code:
  - *Inferred from Experiment* (EXP),
  - *Inferred from Direct Essay* (IDA),
  - *Inferred from Physical Interaction* (IPI),
  - *Inferred from Mutant Phenotype* (IMP),
  - *Inferred from Genetic Interaction* (IGI)
  - *Inferred from Expression Pattern* (IEP).
- Computazionali: annotazioni derivanti da analisi in silico della sequenza dell'entità biomolecolare e/o altri dati riguardo essa. In tale categoria ricadono:
  - *Inferred from Sequence or Structural Similarity* (ISS),
  - *Inferred from Sequence Orthology* (ISO),
  - *Inferred from Sequence Alignment* (ISA),
  - *Inferred from Sequence Model* (ISM),
  - *Inferred from Genomic Context* (IGP)
  - *Inferred from Reviewed Computational Analisys* (RCA).
- Author statement: annotazioni derivanti da citazioni. In tale categoria ricadono*: Traceable Author Statement* (TAS) e *Non-traceable Author Statement* (NAS).

- Da curatore: indica il fatto che il curatore ha ritenuto opportuno aggiungere tale annotazione senza ricadere in tutti gli altri casi sopra citati. In tale categoria ricadono; *Inferred by Curator* (IC) e *No biological Data available* (ND). Quest'ultimo *evidence code* indica la situazione in cui il curatore non ha trovato alcuna informazione riguardante l'entità biomolecolare. Tale *evidence code* è quindi utilizzato solo con associazioni a termini *root* delle ontologie.

Tutti gli *evidence code* sopra citati risultano in ogni caso assegnati da un curatore. L'unico evidence code non attribuito da un curatore è *Inferred from Electronic Annotation* (IEA), il cui metodo di attribuzione è usato internamente al gruppo di curatori (ricerche BLAST, mapping con keyword di Swiss-Prot, ecc.) ma non confermato manualmente ed è ritenuto, quindi, il meno affidabile. Come si può osservare è difficile esprimere in una scala ordinata, quali *Evidence Code* risultano più o meno affidabili; questo e altre problematiche riguardanti le annotazioni inficiano la bontà del lavoro svolto.

1.6 Organizzazione dell'elaborato

Il lavoro è organizzato come segue: nel Capitolo 2 si evidenziano i problemi di tali banche dati a cui questa tesi si prefigge di dare una soluzione. Nel Capitolo 3 descrivo l'architettura del software evidenziando in modo particolare le tecniche statistiche che implementa. Nel Capitolo 4 si espongono gli obiettivi e le motivazioni di questa tesi. Le modifiche puntuali necessarie all'implementazione delle nuove tecniche proposte saranno descritte nel Capitolo 5. Il Capitolo 6 raccoglie le descrizioni teoriche delle analisi statistiche proposte; evidenzierò le differenze dalle soluzioni già disponibili e le questioni irrisolte di questi metodi. Nel Capitolo 7 riporto i risultati sperimentali ottenuti dall'applicazione degli algoritmi implementati; alcuni risultati verranno utilizzati per scegliere il valore di certi parametri, mentre altri valgono come confronto tra le tecniche e valutazione delle prestazioni dei modelli predittivi. Le conclusioni e gli spunti per possibili sviluppi futuri sono riportati rispettivamente nel Capitolo 8 e nel Capitolo 9. Infine nel Capitolo 10 riporto i riferimenti bibliografici.

2 Il problema

Le banche dati che ho descritto finora raccolgono la conoscenza dei fenomeni biologici e sono fondamentali ai ricercatori per interpretare i risultati degli esperimenti. Malgrado la loro indiscutibile utilità questi strumenti non sono esenti da difetti; nelle prossime sezioni presenterò le principali problematiche legate a queste tecnologie.

2.1 Difficoltà nell'efficace utilizzo delle informazioni biomolecolari disponibili

Un primo problema nasce dalla distribuzione e dalla numerosità di queste banche dati che, sebbene cerchino di essere ortogonali tra di loro, spesso presentano ridondanza di informazioni e relazioni tra i termini dei vocabolari, specialmente se sono curate da associazioni diverse. Per questa ragione negli ultimi anni si sono fatti molti sforzi per la realizzazione di strumenti di integrazione che permettano di raccogliere informazioni da più fonti e di mantenerle consistenti ed aggiornate. Il problema è tutt'altro che banale considerata la mole di dati disponibili e la frequenza di aggiornamento; in questa ottica sono nati GPDW e altri sistemi informatici.

Un altro problema è l'incompletezza e la possibile non correttezza dell'informazione contenuta nelle banche dati: per ogni organismo sequenziato solo un sottoinsieme dei suoi geni è conosciuto e solo un sottoinsieme di questi è annotato e, con molta probabilità, anche per questi geni non tutte le funzioni/caratteristiche sono conosciute e ancora meno sono già state annotate; inoltre la maggior parte delle annotazioni è introdotta da dei curatori che esaminano la letteratura, procedimento non certo privo di errori.

Ad esempio, nella Gene Ontology, capita che ci siano geni annotati ad una particolare funzione molecolare ma non al rispettivo processo biologico: questo potrebbe essere solo un piccolo problema per un ricercatore o un biologo interessato alle annotazioni di un singolo gene poiché un umano può facilmente fare delle estrapolazioni ovvie. Tuttavia questo non è il modo in cui queste banche dati vengono generalmente usate, ma solitamente un ricercatore tenta di analizzare il risultato di un esperimento ad alta capacità (*high-throughput*) usando un software che esegue analisi ontologiche; questi software interrogano il database delle annotazioni e calcolano valori statistici in base all'informazione in esso contenute, senza la possibilità di eseguire deduzioni.

È importante sottolineare come queste limitazioni siano intrinseche alla tecnologia delle banche dati, infatti le annotazioni, rappresentando il livello di conoscenza biologica in un istante di tempo, hanno una natura dinamica e sono destinate ad essere aggiunte, modificate o eliminate e quindi difficilmente potranno mai essere considerate definitive e complete.

2.2 Motivazioni dell'utilizzo delle tecniche per la predizione

Ho già accennato alle ragioni che spingono a sviluppare delle applicazioni per l'integrazione dei dati biomolecolari provenienti da fonti eterogenee e citato come esempio GPDW. Purtroppo la mera integrazione dei dati nulla può per limitare l'incompletezza degli stessi. Molte analisi bioinformatiche eseguite oggigiorno sono basate sulle annotazioni di geni e prodotti genici; in questo scenario aumentare tali annotazioni sia in quantità e copertura che in qualità è indispensabile per ottenere risultati migliori in queste analisi. Per questa ragione negli ultimi anni sono stati sviluppati degli strumenti informatici in grado di stabilire la rilevanza di una annotazione o di generare una lista di annotazioni mancanti ordinata secondo il livello di probabilità per velocizzare il lavoro dei curatori. Questi strumenti possono essere utilizzati sia per accrescere l'accuratezza di un insieme di annotazioni in vista di una analisi statistica sia per suggerire nuove ipotesi sperimentali.

Bisogna comunque tenere presente che questi strumenti non vogliono sostituirsi all'attività di ricerca scientifica e che i risultati prodotti necessitano di essere validati sperimentalmente da un esperto che ne garantisca la bontà.

3 La soluzione proposta

Diversi metodi sono stati sviluppati per l'annotazione automatica di geni e termini nei vocabolari controllati: alcuni di essi suggeriscono annotazioni basandosi su associazioni tra termini di vocabolari distinti, altri usano tecniche di allineamento tra le sequenza nucleotidiche (geni) o peptidiche (proteine) per associare ad una sequenza biomolecolare i termini a cui sono associate le sequenze simili.

Altri metodi ancora si affidano ad una analisi semantica di un insieme di annotazioni; ad esempio King et al. [2] hanno proposto l'utilizzo di reti Bayesiane e alberi decisionali per predire annotazioni imparando dei motivi da un insieme di annotazioni disponibili; successivamente Tao et al. [3] hanno proposto l'utilizzo dell'algoritmo *k-nearest neighbor* (k-NN) in cui un gene eredita le annotazioni che sono comuni tra i suoi vicini; Khatri et al. [4] hanno invece proposto l'utilizzo di strumenti di algebra lineare (in particolare la decomposizione a valori singolari o SVD, *singular values decomposition*) per analizzare la matrice delle associazioni gene-termine.

In questo capitolo introdurrò le tecniche di analisi semantica che saranno usate come punto di partenza nello sviluppo di questa tesi; in particolare verrà descritta nel dettaglio la tecnica LSI che sarà nel seguito arricchita con delle tecniche statistiche chiamate "funzioni di peso". Nelle ultime sezioni descriverò l'architettura del software *AnnotationPredictor*, inizialmente sviluppato dal Gruppo di Basi di Dati e Bioinformatica del Politecnico di Milano [5], che andrò ad estendere con nuove funzionalità.

3.1 Rappresentazione di annotazioni biomolecolari come matrice delle occorrenze

L'insieme delle annotazioni di un organismo può essere rappresentato utilizzando una matrice GF con g righe e f colonne. Ogni riga rappresenta una entità biomolecolare mentre ogni una funzionalità (un termine del vocabolario controllato). Un elemento gf_{ij} è 1 se l'i-esimo gene è annotato al j-esimo termine:

$$GF = \{gf_{ij}\} = \begin{cases} 1, & se\ g_i \text{ è } annotato\ a\ f_j \\ 0, & altrimenti \end{cases}.$$

Considerando anche il valore del campo *Qualifier* delle annotazione possiamo aggiungere un ulteriore valore al dominio degli elementi di GF . Un elemento gf_{ij} è uguale a -1 se il esiste una annotazione tra il gene g_i e il termine f_j con valore di *Qualifier* uguale a *NOT*. Una annotazione negativa è più forte di una annotazione nulla nel senso che la prima significa che non sussiste una relazione tra la entità biomolecolare ed il termine del vocabolario, mentre la seconda che non esistono informazioni a riguardo.

Quando un vocabolario controllato ha una struttura ontologica spesso capita che un gene sia annotato solamente al termine più specifico (annotazione diretta) ma non con quelli più generali.

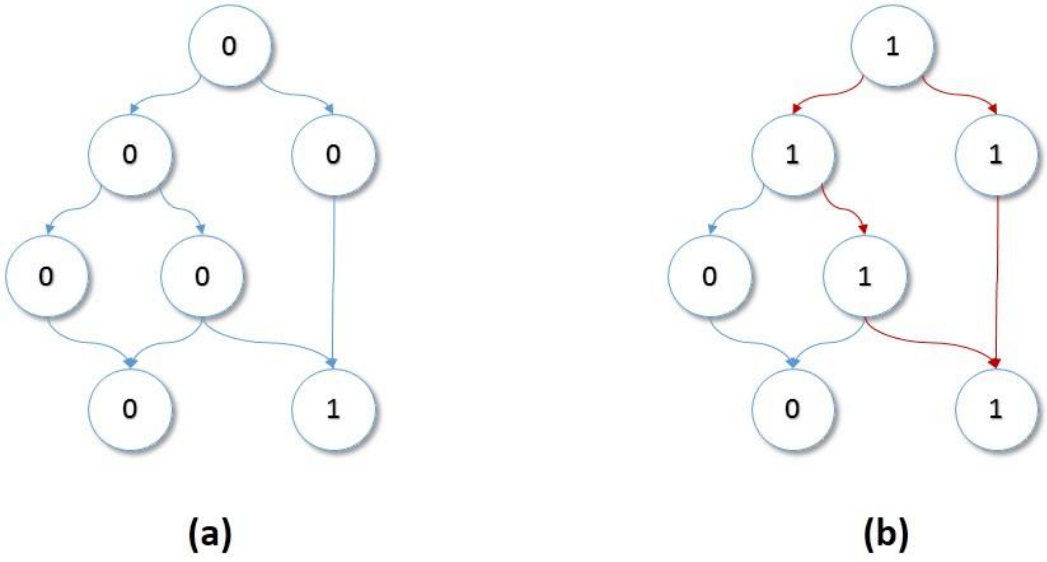


Figura 5: (a) DAG delle annotazioni dirette; (b) DAG delle annotazioni "unfoldate" lungo le direzioni in rosso

Per la natura delle ontologie possiamo considerare un gene annotato ad un termine specifico f associato anche a tutti i termini ascendenti a f, un esempio in Figura 4; questa operazione di estensione della matrice delle annotazioni è chiama *unfolding* (che possiamo tradurre in italiano con il termine esplicitazione).

Tenendo conto delle due modifiche appena descritte possiamo ridefinire la matrice delle annotazioni come:

$$GF = \{gf_{ij}\} = \begin{cases} 1, & se\ g_i \text{ è } annotato\ a\ f_j\ o\ a\ un\ suo\ ascendente \\ -1, & se\ g_i \text{ è } direttamente\ annotato\ a\ f_i\ con\ Qualifier = NOT \\ 0, & altrimenti \end{cases}.$$

Interpretando la matrice GF abbiamo ogni riga rappresenta tutte le funzionalità note avere una relazione con un determinato gene/proteina, mentre ogni colonna rappresenta tutte le entità biomolecolari per cui è presente una annotazione ad una specifica funzionalità.
La matrice delle annotazioni è costruita a partire da una istanza del database delle annotazioni, quindi dalla conoscenza contenuto in esso in un qualche istante di tempo; per questa ragione conserva tutte le caratteristiche di incompletezza e incorrettezza discusse nel capitolo precedente.

3.2 Latent Semantic Indexing

Lo scopo di questa sezione è quello di introdurre dal punto di vista teorico la tecnica LSI (Latent Semantic Indexing), conosciuta anche come LSA (Latent Semantic Analysis), utilizzata nel campo delle Elaborazioni del Linguaggio Naturale (NLP, *Natural Language Processing*) per analizzare le relazioni tra un insieme di documenti e le parole in essi contenute, generando un insieme di concetti. Tuttavia, data la natura matematica e generale dell'approccio utilizzato, LSI è indipendente dal contesto e può essere utilizzata oltre che con parole del linguaggio naturale, con qualsiasi altro tipo di stringa, termine o simbolo.
Sebbene non sia l'unica opzione disponibile, solitamente si sceglie di basare la tecnica LSI sulla fattorizzazione a valori singolari SVD (*Singular Values Decomposition*), che verrà introdotta nel seguito.

3.3 Singular Values Decomposition

In uno spazio vettoriale $\mathbb{R}^n$ una trasformazione lineare è una funzione che possiamo scrivere nella forma:

$$\boldsymbol{y} = A\boldsymbol{x}$$

Dove $\boldsymbol{x}, \boldsymbol{y}\ \in \mathbb{R}\hat{}n$ mentre $A \in\ \mathbb{R}^{n\times n}$ è una matrice quadrata.
Un vettore non nullo $\boldsymbol{t}$ è detto essere autovettore per la trasformazione lineare A se sussiste la seguente:

$$\boldsymbol{Ax} = \boldsymbol{\lambda x}$$

per qualche scalare λ. In questa situazione lo scalare λ viene chiamato autovalore di A corrispondente all'autovettore $\boldsymbol{t}$. Informalmente si può dire che un autovettore di una trasformazione è un vettore che, quando applicato a tale trasformazione, cambia in lunghezza ma non in direzione (Figura 5).

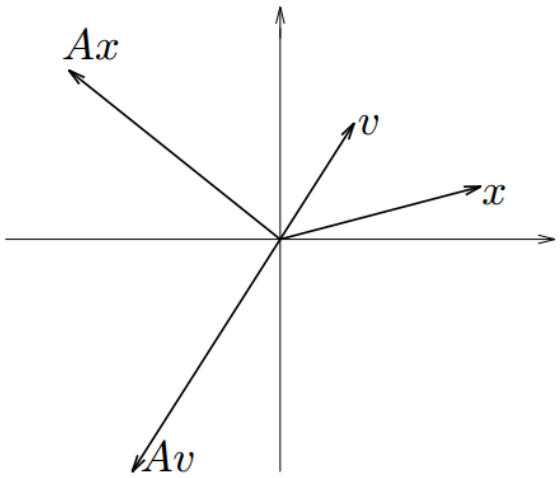


Figura 6: il vettore v è un autovettore e quando applicato alla trasformazione lineare A viene semplicemente scalato; il vettore x non è autovettore e quando applicato ad A cambia in direzione

Ogni matrice quadrata A può essere scritta come:

$$A = Q\Lambda Q^T$$

dove la matrice Q è una matrice quadrata delle stesse dimensioni di A e tale per cui $QQ^T = I$ (ortonormalità), mentre la matrice $\Lambda = diag(\lambda_1, \dots, \lambda_n)$ è la matrice diagonale formata dagli autovalori di A.

Riscrivendo:

$$\boldsymbol{y} = A\boldsymbol{x} = Q\Lambda Q^T\boldsymbol{x}$$

possiamo interpretare l'applicazione della trasformazione lineare A ad un vettore $\boldsymbol{x}$ come l'applicazione di tre operazioni:

i. Prendiamo i coefficienti di $\boldsymbol{x}$ lungo le direzioni definite dagli autovettori di A: $\boldsymbol{v} = Q^T\boldsymbol{x}$;

ii. Scaliamo coefficienti per i valori λ_i: $\boldsymbol{u} = \Lambda Q^T \boldsymbol{x}$;

iii. Ricostruiamo il vettore lungo le direzioni degli autovettori: $\boldsymbol{y} = Q\Lambda Q^T \boldsymbol{x}$.

Un esempio è fornito in Figura 6.

Anche se ininfluente ai fini di questa trattazione si precisa che una matrice potrebbe avere autovalori e autovettori complessi, ma si può dimostrare che una matrice simmetrica (Hermitiana in generale) ha autovalori e autovettori reali.

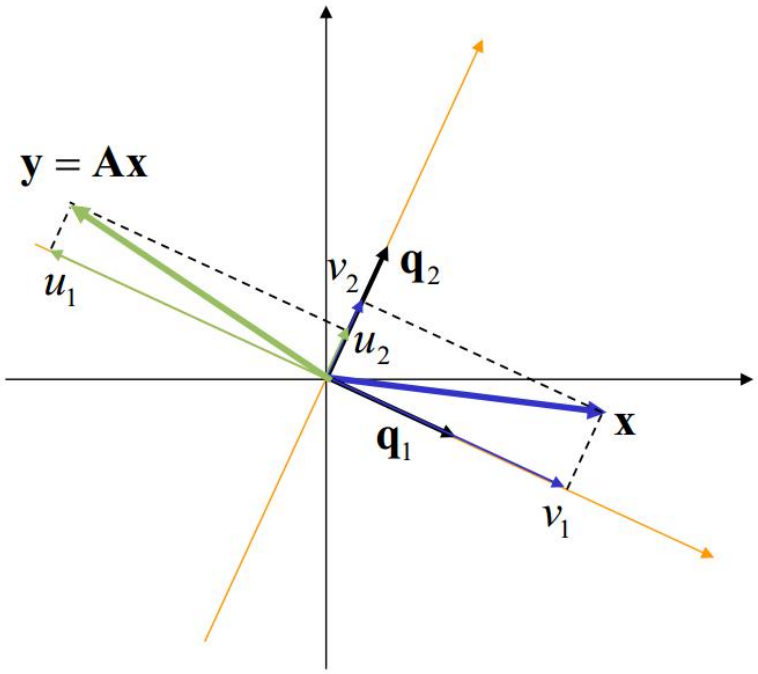


Figura 7: interpretazione dell'applicazione della trasformazione lineare A; q_1 e q_2 sono gli autovettori della matrice A.

In quanto detto finora si è sempre fatta l'assunzione che la matrice A fosse quadrata; infatti è facile verificare che una matrice rettangolare non può avere autovettori per il fatto che, nella applicazione:

$$\boldsymbol{y} = A\boldsymbol{x}$$

con $A \in \mathbb{R}^{n \times m}$ i vettori $\boldsymbol{y}$ ed $\boldsymbol{x}$ hanno dimensioni differenti.

Si vogliono trovare delle quantità di una matrice regolare che possiedono un significato comparabile a quello degli autovalori e autovettori; tali elementi sono i valori singolari e i vettori singolari.

Data una matrice rettangolare $A \in \mathbb{R}^{n \times m}$ la sua fattorizzazione SVD è:

$$A = U\Sigma V^T$$

dove:

i. $r \in \mathbb{N}$ è minore sia di n che di m;

ii. $U \in \mathbb{R}^{n \times r}$ è una matrice tale che $UU^T = I$, i cui vettori-colonna $\boldsymbol{u}_i$ sono i vettori singolari sinistri (o di output);

iii. $\Sigma = diag(\sigma_1, \dots, \sigma_r) \in \mathbb{R}^{r \times r}$, con $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_r$ è una matrice quadrata i cui elementi σ_i della diagonale principale sono i valori singolari non nulli;

iv. $V \in \mathbb{R}^{r \times m}$ è una matrice tale che $VV^T = I$, i cui vettori-colonna $\boldsymbol{v}_i$ sono i vettori singolari destri (o di input).

Si può osservare che:

$$AA^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma^2 U^T$$

ovvero i vettori $\boldsymbol{u}_i$ sono gli autovettori della matrice AA^T, e che:

$$A^T A = (U\Sigma V^T)\text{^}T(U\Sigma V^T) = V\Sigma^2 V^T$$

ovvero i vettori $\boldsymbol{v}_i$ sono gli autovettori della matrice $A^T A$, e che quindi gli elementi σ_i della diagonale di Σ sono le radici quadrate degli autovalori di $A^T A$ o AA^T.

Seguendo lo stesso procedimento proposto in precedenza, l'applicazione lineare:

$$\boldsymbol{y} = A\boldsymbol{x}$$

può essere scomposta in tre passi:

i. Prendiamo i coefficienti di $\boldsymbol{x}$ lungo le direzioni definite dalle direzioni di input $\boldsymbol{v}_i$;

ii. Scaliamo i coefficienti per i valori σ_i;

iii. Ricostruiamo il vettore lungo le direzioni di output: $\boldsymbol{u}_i$.

La differenza principale rispetto al caso delle matrici quadrate è che nella decomposizione SVD le direzioni di input e di output sono differenti; inoltre poiché i valori singolari nella matrice Σ sono in ordine decrescente il vettore $\boldsymbol{v}_1$ corrisponde alla direzione di ingresso più sensibile e la direzione $\boldsymbol{u}_1$corrisponde alla direzione di uscita con guadagno maggiore.

La decomposizione SVD ha tante applicazioni pratiche; una delle più note è la determinazione di una approssimazione a rango ridotto della matrice data.

Infatti (Teorema di Eckart e Young) sia $A \in \mathbb{R}^{n \times m}$ una matrice tale the:

$$rango(A) = r \leq p = \min(n, m)$$

sia:

$$A = U\Sigma V^T$$

la sua decomposizione SVD. Allora la matrice:

$$A_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i \, , \qquad con\ k < r$$

è tale per cui:

$$\min_{B\,:\,rank(B)=k} \| A - B \|_F = \| A - A_k \|_F$$

dove la norma matriciale utilizzata è la norma di Frobenius, definita come:

$$\| X - Y \|_F = \sqrt{\sum\nolimits_{i,j} \left(x_{ij} - y_{ij}\right)^2}$$

In altre parole, se usiamo come funzione obiettivo la norma di Frobenius, la matrice A_k è la migliore approssimazione di rango k della matrice A. Questo metodo viene anche chiamato decomposizione SVD troncata.

Infine, dal punto di vista pratico, esistono alcuni algoritmi per calcolare la fattorizzazione SVD di matrici i grandi dimensioni; uno di questi, quello che sarà usato nel seguito, è l'algoritmo proposto da Lanczos [12].

3.2.1 L'algoritmo

Per semplicità di notazione la tecnica Latent Semantic Indexing verrà introdotta nel contesto del Natural Language Processing; il passaggio all'ambito biomolecolare è molto semplice, basta sostituire alla matrice delle occorrenze documento-termine la matrice delle associazioni GF introdotta in precedenza facendo coincidere i documenti alle entità biomolecolari e i termini alle funzionalità.

Dato un insieme finito $\Delta = \{d_1, d_2, \dots, d_D\}$ di documenti e un insieme finito $\Omega = \{\omega_1, \omega_2, \dots, \omega_T\}$ di vocaboli permessi si definisce matrice delle occorenze, o matrice documento-termine, la matrice $X \in \mathbb{N}^{D \times T}$, con D righe e T colonne i cui elementi sono definiti come:

$$x_{ij} = numero\ di\ occorrenze\ del\ termine\ \omega_j\ nel\ documento\ d_i \, .$$

Si noti che nel caso delle annotazioni biomolecolari la matrice X ha valori in $\mathbb{Z}$ poiché l'elemento -1 è consentito.

La LSI è una tecnica di fattorizzazione della matrice documento-termine, nel senso che, a meno di una ri-normalizzazione delle righe e delle colonne, X può essere riscritta come il prodotto di due matrici:

$$X = AB^T$$

dove $A \in \mathbb{R}^{D \times s}$ e $B \in \mathbb{R}^{T \times s}$ e tali per cui:

$$(A, B) = \operatorname{argmin} \| X - AB^T \|_F \qquad \text{(Eq. 1)}$$

3.2.2 LSI tramite SVD

Risulta evidente che l'equazione (Eq. 1) ha più di una soluzione; infatti trovata una soluzione valida moltiplicando e dividendo rispettivamente A e B per uno stesso scalare si ottiene una altra soluzione ugualmente valida.

Solitamente si sceglie di prendere una decomposizione a basi ortogonali ordinate per valore decrescente del rispettivo valore singolare, cioè la decomposizione SVD, definendo $A = U\Sigma$ e $B = V$.

Se invece della decomposizione intera della matrice, prendiamo la decomposizione troncata al k-esimo otteniamo una approssimazione della matrice documento-termine:

$$\hat{X} = U_k \Sigma_k V_k^T$$

in cui appaiono solo le relazioni maggiormente rappresentate all'interno della matrice di partenza; questo può aiutare a ripulire i dati dal rumore.

Possiamo interpretare geometricamente i risultati di LSI immaginando che la decomposizione SVD troncata al k-esimo livello produca k "concetti". Poniamo:

$$A_k = U_k \Sigma_k = \begin{bmatrix} d_1 c_1 & \cdots & d_1 c_k \\ \vdots & \ddots & \vdots \\ d_D c_1 & \cdots & d_D c_k \end{bmatrix}$$

dove ogni elemento $\{d_i c_j\}$ rappresenta quanto il j-esimo concetto è espresso nell'i-esimo documento; poniamo inoltre:

$$B_k^T = V_k^T = \begin{bmatrix} c_1\omega_1 & \cdots & c_1\omega_T \\ \vdots & \ddots & \vdots \\ c_k\omega_1 & \cdots & c_k\omega_T \end{bmatrix}$$

dove ogni elemento $\{c_i\omega_j\}$ è interpretato come il peso che il termine ω_j ha all'interno del concetto c_i; un elemento $\hat{x}_{ij}$ della matrice delle occorrenze approssimata è calcolato come composizione lineare dei contributi che ogni concetto dà al j-esimo termine pesati con una misura di quanto ciascun concetto sia rappresentativo per il documento d_i :

$$\hat{x}_{ij} = \sum_{n=1}^{k} d_i c_n * c_n \omega_j$$

È bene notare che i concetti sono tra loro completamente incorrelati (il che corrisponde al fatto che le loro direzioni sono ortogonali); inoltre la matrice B risulta essere indipendente dalle rotazioni dei dati in ingresso, infatti, se R è una matrice rotazionale:

$$RX = (RU\Sigma)V^T = (RA)B^T.$$

Da quest'ultima considerazione è possibile dedurre come sia in realtà difficile dare una interpretazione semantica ai concetti; essi infatti sono da intendersi come dei concetti a livello algebrico a cui non necessariamente si riesce a dare un'interpretazione semantica; l'assunzione di LSI è che i termini che appaiono spesso insieme in un documento abbiano un significato simile.

La tecnica LSI ha molte applicazioni pratiche. Un problema che può essere affrontato con la tecnica LSI è la determinazione della similarità tra due documenti; dopo il calcolo delle matrici ogni riga della matrice A_k rappresenta un documento e la similarità tra due documenti può essere stimata applicando una qualsiasi metrica, come ad esempio il coseno di similitudine. Con lo stesso procedimento si può calcolare la similitudine tra i termini decomponendo la matrice X^T. I risultati del calcolo della similitudine possono quindi essere usati per operazioni di clustering.

Un'altra applicazione pratica è la predizione di associazioni mancanti o l'identificazione di associazioni non corrette; supponiamo, per semplicità, che la matrice X sia binaria (ogni elemento indica soltanto se il termine appare nel documento, senza un conteggio delle occorrenze). Scegliamo ora una soglia $\tau \in (0,1) \subset \mathbb{R}$ e consideriamo gli elementi della matrice approssimata $\hat{X}_k$; ogni elemento cadrà in una delle seguenti quattro classi:

i. $\hat{x}_{ij} \leq \tau \ \wedge \ x_{ij} = 0$: vero negativo, l'algoritmo conferma l'assenza dell'associazione;

ii. $\hat{x}_{ij} \leq \tau \ \wedge \ x_{ij} = 1$: falso negativo, l'algoritmo suggerisce di eliminare l'associazione;

iii. $\hat{x}_{ij} > \tau \ \wedge \ x_{ij} = 0$: falso positivo, l'algoritmo suggerisce di aggiungere l'associazione;

iv. $\hat{x}_{ij} > \tau \ \wedge \ x_{ij} = 1$: vero positivo, l'algoritmo conferma l'associazione;

Nell'ambito del m*achine learning* la tecnica LSI viene usata per costruire un modello a partire da un insieme di documenti noto (chiamata solitamente *training set*); tale modello è poi utilizzato per predire associazioni in documenti nuovi (cioè non presenti nell'insieme iniziale). Il modello generato è la matrice troncata B_k ; ora si può applicare il modello ad un nuovo documento $\boldsymbol{x}$ nel seguente modo:

$$\hat{\boldsymbol{x}} = \boldsymbol{x} \, B_k B_k^T \qquad \text{(Eq. 2)}$$

ottenendo un nuovo vettore con le associazioni predette.

L'operazione nella Eq. 2 significa che il vettore di output è la composizione lineare delle proiezioni del vettore iniziale lungo le k direzioni maggiormente amplificate della matrice B.

3.2.3 Limiti di LSI

La tecnica LSI ha alcuni problemi e limitazioni. Innanzitutto non è conosciuto al momento un metodo per calcolare il livello di troncamento ottimale; solitamente il valore di k è scelto con funzioni euristiche oppure attraverso tecniche di validazione che prevedono di stimare e confrontare tanti modelli con diversi di troncamento per scegliere quello con prestazioni migliori. Un altro limite, in questo caso espressivo, di LSI è l'impossibilità di trattare correttamente i termini polisemantici, cioè quei termini che assumono un significato diverso a seconda del contesto in cui sono inseriti. Infatti dalla Eq. 2 si può vedere come una associazione in input contribuisca a determinare i valori del vettore di output in modo del tutto indipendente dal valore delle altre associazioni in input (il contesto).

3.3 L'architettura del software AnnotationPredictor

Il lavoro presentato in questa tesi è una estensione di un software sviluppato da Roberto Sarati nel 2009 [6] per la predizione di annotazioni funzionali; l'applicativo è progettato per lavorare con i dati presenti nel data warehouse GPDW. Nella sua versione originale consente di predire una lista di annotazioni da aggiungere o eliminare a partire da un insieme di annotazioni noto, applicando i

metodi LSI, SIM1 e SIM2; inoltre fornisce strumenti per la valutazione dei modelli generati e l'autotuning dei parametri.

Nel proseguo di questa sezione verranno descritte alcune delle caratteristiche principali del software e i protocolli utilizzati; per i dettagli tecnici si rimanda alla tesi dell'autore.

3.3.1 Costruzione della matrice delle annotazioni

La matrice delle annotazioni su cui si calcola il modello è ottenuta dal database GPDW tramite una interrogazione che l'utente può personalizzare scegliendo la tassonomia e uno o più vocabolari da cui estrarre i dati. L'interrogazione esclude le annotazioni relative a geni o termini marcati come obsoleti e quelle con un grado di evidenza uguale a IEA (annotazioni predette elettronicamente) oppure ND (annotazioni senza dati biologici di supporto, tipicamente annotazioni alla root di una ontologia quando non si conoscono associazioni più specifiche con altri termini della stessa). Le annotazioni che hanno un grado di *Qualifier* impostato su *NOT* vengono inserite nella matrice con valore -1.

L'operazione fin qui descritta produce la matrice delle annotazioni dirette, cioè quelle annotazioni inserite all'interno del database. Se il vocabolario controllato da cui si estraggono i dati è un ontologia allora si esegue l'operazione di *unfolding* descritta in precedenza, cioè ogni gene viene annotato anche a tutti gli ascendenti dei termini a cui è direttamente annotato.

Dalla matrice risultante si eliminano tutte quelle colonne in cui compaiano meno di L (di default $L = 3$) elementi diversi da zero; questo operazione è detta *pruning* (potatura).

Infine si esegue una permutazione casuale dei vettori riga per eliminare eventuali dipendenze tra geni "vicini" all'interno del database.

La struttura dati così prodotta risulta essere una matrice molto sparsa, in cui generalmente il numero di elementi non nulli è compreso tra 1% e il 10% che viene rappresentata in memoria con una struttura dati dedicata per risparmiare spazio.

3.3.2 Anomalie e loro correzione

Per essere consistente, l'insieme delle anomalie deve rispettare la *True Path Rule* (regola del cammino coerente) cioè ogni volta che un gene è annotato ad un termine deve essere annotato anche a tutti i suoi ascendenti. Questa condizione è sicuramente vera nella matrice di input, per

costruzione; viceversa, la matrice predetta potrebbe presentare delle anomalie, ovvero delle situazioni in cui il valore di un termine

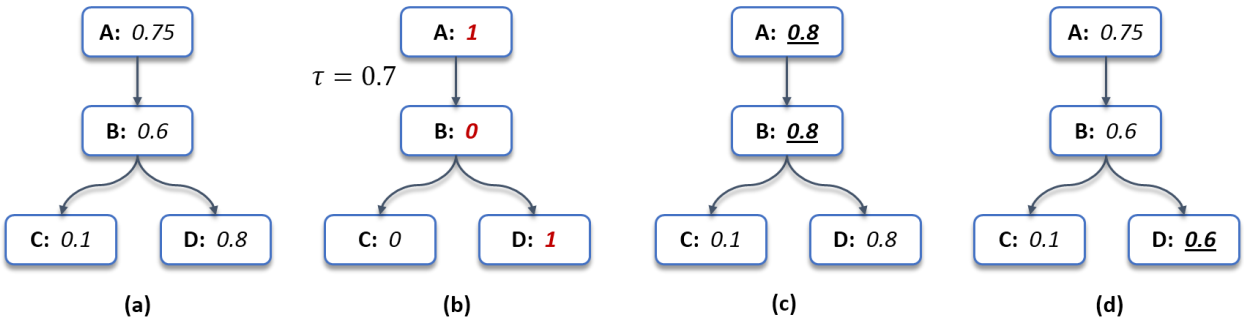


Figura 8:Ipotetico profilo di annotazioni di un gene; (a) valori predetti dall'algoritmo; (b) annotazioni predette dopo la fase di sogliatura al livello 0.7, i nodi A-B-D violano la True Path Rule; (c) correzione di anomalie From Leaves To Root, con sogliatura a 0.7 il termine D sarebbe ancora predetto; (d) correzione di anomalie From Root To Leaves, con sogliatura 0.7 verrebbe predetto solamente l'annotazione ad A.

sia maggiore di quello del padre. In questi casi per valori di soglia compresi tra il valore del padre e quello del figlio avremmo un profilo di annotazione che viola la *True Path Rule*.

Per ottenere un risultato consistente è necessario applicare un algoritmo di correzione di anomalie. Due approcci diversi possono essere adottati: a partire dai nodi foglia, si prosegue verso le radici del DAG ed ogni volta che si trova un nodo in cui un suo padre abbia valore minore si assegna a quel padre lo stesso valore del nodo; oppure, si parte dalla radice e ogni volta che si incontra un nodo con un figlio che ha un valore maggiore si assegna al nodo figlio lo stesso valore del padre. I due metodi si chiamano rispettivamente *From Leaves to Root* (dalle foglie alla radice) e *From Root to Leaves* (dalla radice alle foglie), un esempio di applicazione è riportato in Figura 7.

Nell'elaborato di Sarati si è evidenziato come il secondo procedimento sia peggiore in quanto riduca troppi valori dei nodi.

3.3.3 Dettagli implementativi

L'applicazione è composta da una parte in sviluppata in Java e da una libreria per l'esecuzione delle analisi scritta i C++; le due parti comunicano tra di loro attraverso il framework JNI (Java Native Interface), si veda la Figura 8.

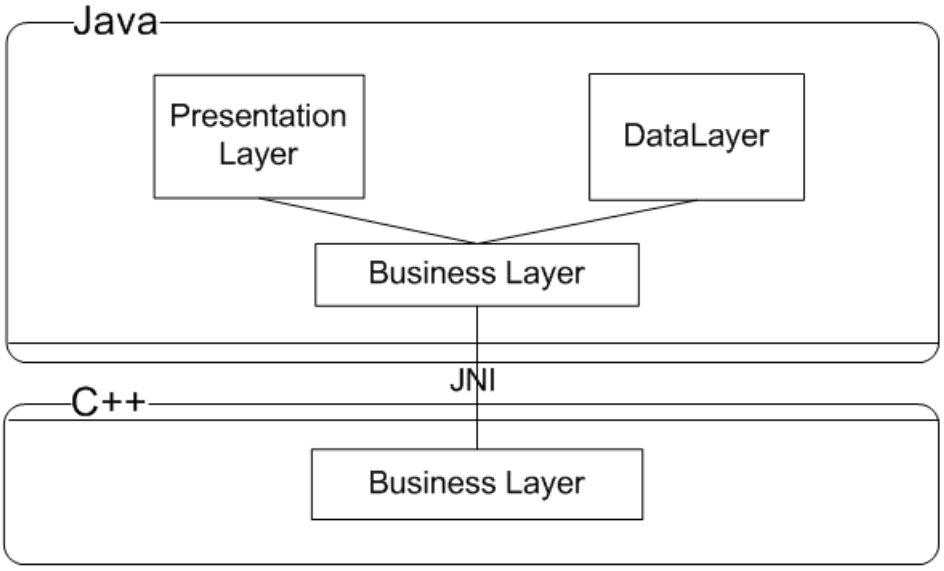


Figura 9: Architettura generale del sistema

L'interfaccia Java si occupa di elaborare l'input dell'utente, di visualizzare l'output e di interagire con il DBMS (*Data Base Management System*) . Il core computazionale C++ invece esegue le analisi e applica gli algoritmi e protocolli descritti nelle sezioni precedenti di questo capitolo.
La parte di codice scritta in C++ utilizza a sua volta delle librerie *opensource*, in particolare AMD ACML come kernel matematico, SvdLibC per il calcolo della decomposizione SVD (implementa l'algoritmo di Lanczos) e Boost per i tipi di dato.
Oltre alla interfaccia Desktop è stata sviluppata anche una interfaccia web per accedere al servizio in remoto attraverso un comune browser web.
Una descrizione dettagliata dell'implementazione del sistema la si può trovare nella tesi di Roberto Sarati, intitolata "Metodi ed architetture software per la predizione di annotazioni genomiche funzionali e la stima di similarità semantica di geni e proteine".

4 Obiettivi del progetto di tesi

Nell'elaborato di Roberto Sarati si è dimostrata la validità dell'approccio LSI per la predizione di nuove annotazioni nel data warehouse GPDW.

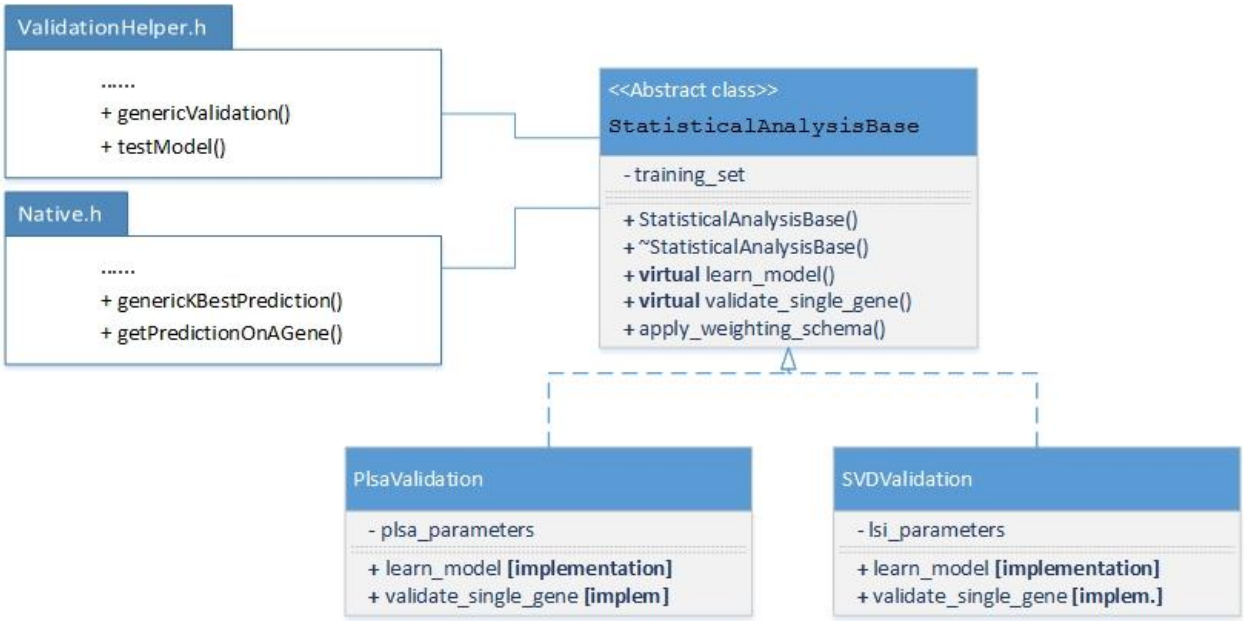
Studi condotti negli anni precedenti nel campo della analisi semantica e delle sue applicazioni al campo delle annotazioni biomolecolari hanno mostrato come sia possibile ottenere dei risultati migliori rispetto a quelli forniti da LSI. In particolare nel 1999 Hofmann [32] ha proposto un nuovo algoritmo, chiamato pLSA (*probabilistic Latent Semantic Analysis*), che perfeziona LSI utilizzando delle fondamenta probabilistiche anziché algebriche. Nel 2006 Draghici et al. [10] hanno invece proposto di migliorare il potere predittivo di LSI applicando alla matrice delle annotazioni degli schemi di peso.

In questa tesi si vuole eseguire una reingegnerizzazione dell'architettura presentata nel capitolo precedente per permette di implementare sia nuovi modelli predittivi, in particolare pLSA, che le funzioni di peso per la matrice delle annotazioni, creando una infrastruttura che permetta di confrontare i diversi modelli predittivi.

In primo luogo si progetterà una nuova struttura per il software *AnnotationPredictor* che permetta di condividere tra diversi metodi predittivi lo stesso codice per la valutazione dei risultati (validazione), la predizione di nuove annotazioni e la correzione di anomalie. In una seconda fase si implementeranno l'algoritmo pLSA e le funzioni di pesatura. L'analisi pLSA risulta essere computazionalmente onerosa, specialmente per istanze di dati grandi come nel caso delle annotazioni biomolecolari; si cercherà quindi di fare una implementazione il più efficiente possibile. Infine verrà progetta ed eseguita una campagna di test per valutare le prestazioni dei nuovi modelli, i miglioramenti rispetto a quanto già implementato e i limiti e problemi di questi nuovi approcci.

5 Reingegnerizzazione del software AnnotationPredictor

Ai fini di questo elaborato è necessario fare qualche modifica alla struttura del software originale implementato da Sarati; le modifiche si muovono nella direzione di garantire il riutilizzo del codice all'interno dell'applicazione per quelle operazioni comuni a tutti i metodi predittivi: validazione, predizione, correzione di anomalie. Questo, oltre ad essere una delle *best practice* suggerite dall'ingegneria del software, si rivela necessario per garantire la correttezza del confronto tra metodi predittivi; infatti è necessario che le condizioni e i processi in cui vengono applicati gli algoritmi siano sempre uguali. A questo scopo si è cercato di astrarre il più possibile e di separare le caratteristiche generali di un modello predittivo da quelli che invece sono i dettagli



propri del modello; l'architettura di riferimento è quella in Figura 9.

Figura 10: Schema delle nuove componenti architetturali

Le modifiche riguardano solo la parte del core matematico in C++; il nuovo metodo genericValidation() si occupa di implementare il processo di validazione su una classe astratta StatisticalAnalysisBase. Questa classe definisce dei metodi virtuali C++ che, in quanto tali, non possono essere chiamati direttamente ma devono essere implementati da una classe derivata. La classe StatisticalAnalysisBase contiene tutti gli attributi necessari a qualsiasi metodo predittivo: training set, numero di caratteristiche (feature), numero di geni, numero massimo di thread da

usare. Questa classe inoltre definisce ed implementa gli schemi di peso; infatti la funzione di peso può essere vista come un metodo statistico particolare, non alternativo agli altri ma addizionale. È quindi una parte di codice che deve essere condivisa tra tutti i metodi predittivi.
Nelle classi PlsaAnalysis e SVDAnalysis invece vengono memorizzati tutti gli attributi necessari al particolare metodo predittivo, ad esempio il livello di troncamento nel caso di SVD. Inoltre vengono implementati i metodi virtuali learn_model() e validate_single_gene().

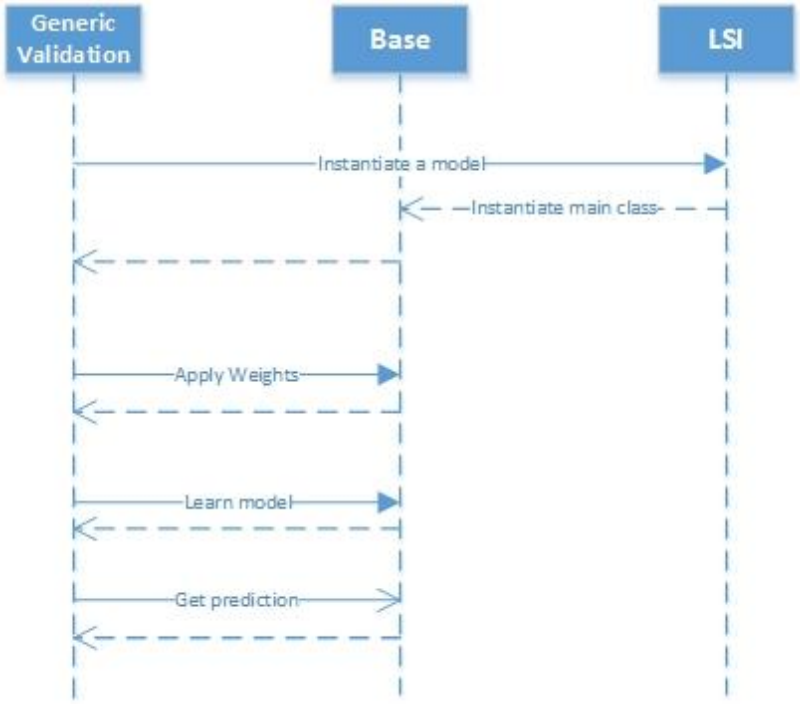


Figura 11: Processo di validazione, esempio nel caso di modello LSI

Il processo di validazione avviene come rappresentato nel diagramma in *Figura 10*; la funzione GenericValidation() istanzia una classe SVDAnalysis a cui si riferisce tramite un riferimento alla classe padre StatisticalAnalysisBase (come permesso dal linguaggio C++). La funzione GenericValidation() durante il processo di validazione invoca metodi di StatisticalAnalysisBase; alcuni di questi metodi sono implementati nella classe StatisticalAnalysisBase (come ad esempio l'applicazione degli schemi di peso). Le chiamate a quei metodi che invece sono dichiarati all'interno della classe StatisticalAnalysisBase come puramente virtuali vengono ridiretti sulla implementazione della classe figlia particolare (in *Figura 10*, ad esempio, il metodo

learn_model() viene invocato sulla classe StatisticalAnalysisBase, ma essendo un metodo virtuale viene ridiretto sulla implementazione della classe specifica).

In questo modo il codice contenuto in GenericValidation() distingue tra diversi metodi solo durante la creazione (dove questo è necessario, poiché ad esempio ogni metodo ha dei parametri differenti) ma nel seguito della esecuzione non distingue tra i diversi metodi; questo assicura che il codice, e dunque la procedura, di validazione sia uguale con qualsiasi algoritmo predittivo, rendendo così possibile un confronto oggettivo tra i metodi.

A differenza di quanto avveniva nella versione precedente del software ora il codice per la creazione della lista delle predizioni è lo stesso usato per la validazione; la differenza è solo nel dataset usato come training set che nel caso delle validazione è una porzione della matrice delle annotazioni invece nel caso della predizione è la matrice intera. Anche questa modifica è stata fatta per uniformare il codice, ridurre le duplicazioni e garantire che il codice testato nella validazione sia lo stesso che si userà per la predizione.

Per quanto riguarda l'interfaccia della libreria si è scelto di non apportare modifiche rispetto alla versione precedente; i metodi GenericValidation() e GenericKBestPrediction() non sono esportati, cioè non sono invocabili all'esterno delle libreria. Il codice Java continua ad invocare i vecchi metodi PlsaValidation(), SvdValidation(), PlsaKBestPrediction(), ecc., ma questi ora corrispondo solo a un *wrapper* verso i metodi generici. Si è ritenuto che questa opzione fosse migliore rispetto che a passare il tipo di modello come parametro; in caso contrario si sarebbe dovuta mantenere una lista delle corrispondenze simbolo-metodo, un idioma non molto comodo e difficilmente scalabile. In Figura 11 è riportato un esempio dello scenario di utilizzo del software per la predizione di nuove annotazioni.

5.1 Modifiche puntuali

Per permettere l'applicazione degli schemi di peso si è dovuta modificare la classe SparseMatrix che definisce la struttura dati contenente il *training set*. In particolare nella versione originale questa struttura dati era per la sola lettura, cioè una volta creata non poteva essere modificata. È stato necessario implementare un metodo che consentisse di modificare i valori.

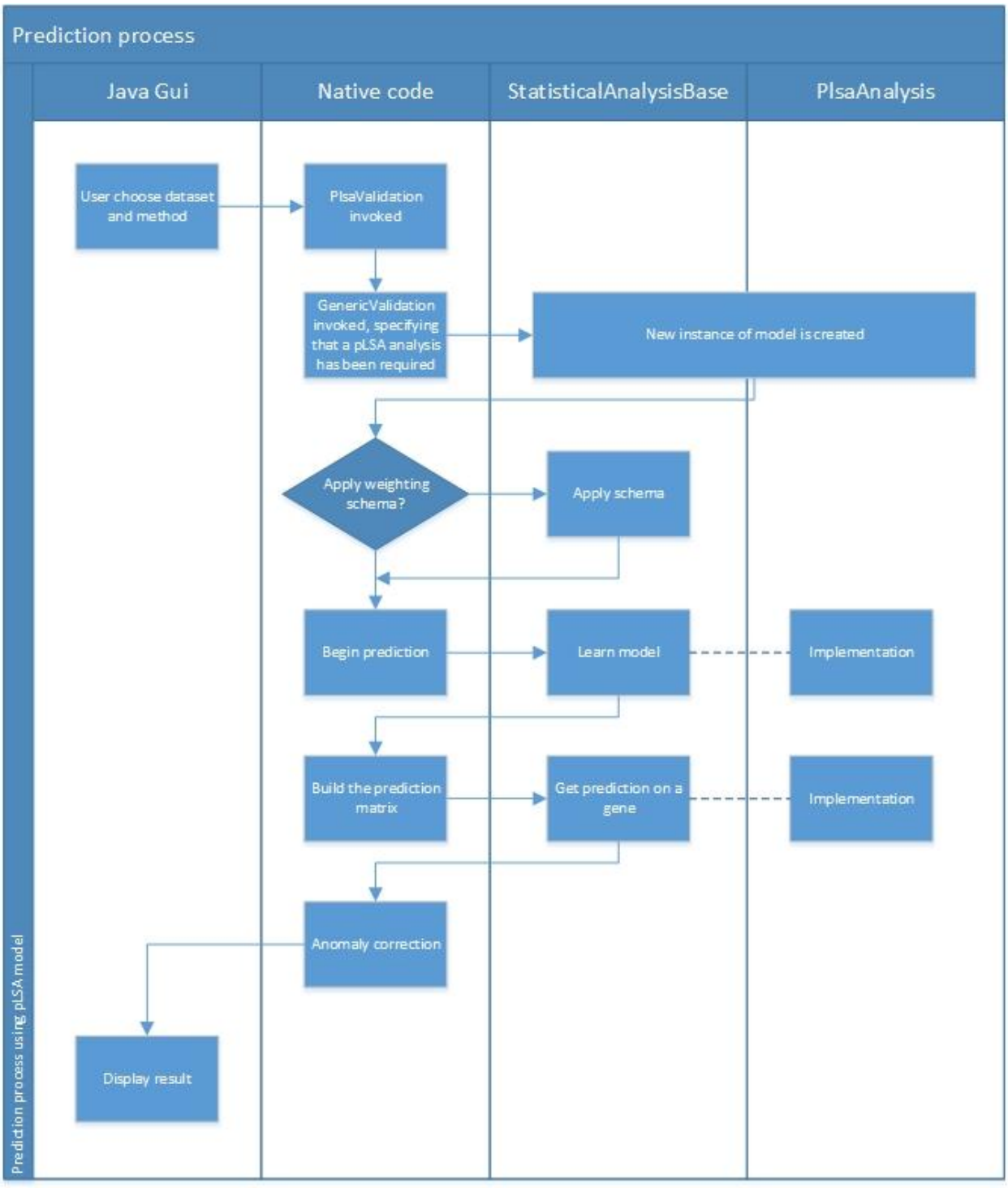


Figura 12: Workflow del processo di predizione, nell'esempio si usa il metodo pLSA

6 Implementazione di nuove tecniche per l'analisi semantica

In questo capitolo verranno descritte le nuove tecniche semantiche che si intende implementare nel software; inoltre si proporrà un confronto tra l'algoritmo LSI e l'algoritmo pLSA.

6.1 Probabilistic Latent Semantic Analysis

Probabilistic Latent Semantic Analysis (pLSA), nota anche come Probabilistic Latent Semantic Indexing, è una tecnica statistica per l'analisi di una matrice delle co-occorrenze. È stata introdotta da Thomas Hoffman nel 1999 [7]. In contrasto a LSI usa un approccio statistico basato sulla funzione di verosimiglianza; inoltre definisce un modello generativo per i dati e questo facilita l'interpretazione dei risultati. Rispetto a LSI permette di trattare con la polisemia, distinguendo esplicitamente tra diversi significati di uno stesso termine.

L'algoritmo pLSA è stato usato in passato nell'ambito dell'*Information Retrieval* per l'indicizzazione di documenti testuali [13] e per la determinazione dell'argomento deli articoli nei *blog* [14] e nell'ambito della Visione Artificiale la classificazione di immagini [15].

Il modello probabilistico su cui si basa pLSA è stato chiamato *Aspect Model*. Si tratta di un modello basato su delle variabili latenti (chiamate *topic*) che permette di rappresentare qualsiasi tipo di informazione esprimibile come matrice delle co-occorrenze.

6.1.1 Il concetto di *topic*

Prendendo come esempio l'ambito dell'Elaborazione del Linguaggio Naturale un *topic* è un "argomento" che può essere trattato all'interno di un documento. A seconda dell'argomento scelto saranno più frequenti alcuni termini piuttosto che altri. Inoltre un argomento può anche determinare il significato di un termine: si pensi al termine "*radice*" che può assumere significati completamente non correlati se l'argomento è "*piante*" piuttosto che "*polinomi*". In un *topic*, ad ogni termine è associato un grado di probabilità che corrisponde all'aspettativa di trovare il termine in un documento che tratta quell'argomento. Semplificando possiamo considerare un *topic* come una lista di termini ordinata per valore di probabilità o più precisamente come una distribuzione multinomiale di termini.

Il processo di generazione di un documento nel modello probabilistico di pLSA prevede i seguenti passi: si scelgono uno o più *topics* che saranno trattati nel documento; per ciascuno di essi si scelgono uno o più termini con probabilità condizionata dai *topics* scelti.

6.1.2 Il modello formale

Nel seguito presento una trattazione più formale del modello generativo di pLSA.

Si supponga di avere un insieme fissato e finito di documenti $D = \{ d_1, d_2, \dots, d_D\}$, un insieme fissato e finito di variabili nascoste (i *topics*) $T = \{t_1, t_2, \dots, t_T\}$ e un insieme fissato e finito di possibili termini che possono apparire nei documenti $\Omega = \{\omega_1, \omega_2, \dots, \omega_\Omega\}$.

Sia θ una distribuzione di probabilità multinomiale con D eventi possibili, che descrive la probabilità di scegliere un documento all'interno del corpus.

Si definiscono poi due insiemi di distribuzioni di probabilità, sempre multinomiali:

i. $\Phi = \{\varphi_i \mid i = 1, 2, \dots, D\}$, dove $\varphi_i(k)$ rappresenta la probabilità che nell'i-esimo documento si scelga la k-esima variabile nascosta;
ii. $\Psi = \{\psi_i \mid i = 1, 2, \dots, T\}$, dove $\psi_i(k)$ rappresenta la probabilità che nell'i-esimo *topic* si scelga il k-esimo termine.

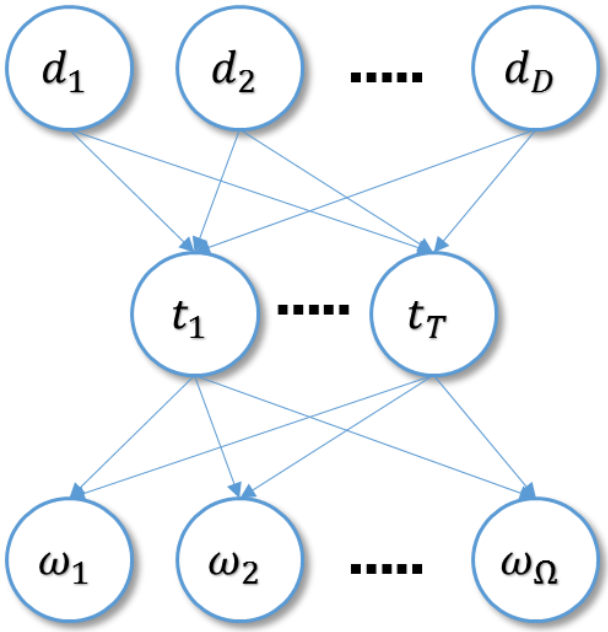


Figura 13: Rappresentazione grafica dello Aspect Model

Ogni coppia osservabile (d, ω) è generata secondo usando il modello generativo (rappresentato in *Figura 12*):

i. Si sceglie il documento d, con probabilità $P(d)$;
ii. Si sceglie ogni *topic* t con probabilità $P(t \mid d)$, data dalla distribuzione φ_d ;
iii. Per ogni *topic* si sceglie il termini ω, con probabilità $P(\omega \mid t)$, data dalla distribuzione ψ_t;

La probabilità che il termine ω compaia all'interno del documento d è quindi data da:

$$P(\omega|d) = \sum_{t \in T} P(\omega|t)P(t|d).$$

Applicando la regola di Bayes abbiamo che la probabilità della associazione (d, ω) è:

$$P(d, \omega) = P(d)P(\omega|d) = P(d) \sum_{t \in T} P(\omega|t)P(t|d).$$

Applicando nuovamente la regola di Bayes:

$$P(d)P(\omega|t) = P(t)P(t|\omega)$$

possiamo ottenere una formulazione equivalente:

$$P(d, \omega) = \sum_{t \in T} P(t)P(t|\omega)P(t|d)$$

simmetrica rispetto alle variabili latenti; nella *Figura 13* una rappresentazione grafica delle differenze tra i due modelli predittivi.

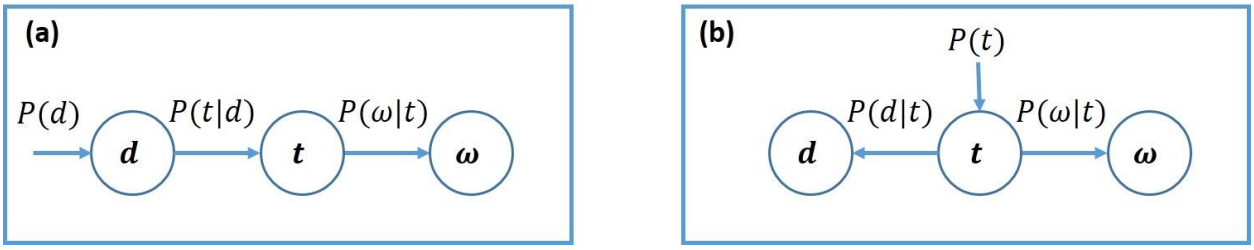


Figura 14: Paragone tra le due versioni del modello predittivo; (a) formulazione asimmetrica, (b) formulazione simmetrica.

Si può notare come la sommatoria delle probabilità condizionate di avere un vocabolo sia uguale a 1, cioè le righe della matrice $P(\omega|d)$ hanno somma unitaria, infatti:

$$\forall\, d\colon \quad \sum_{\omega} P(\omega|d) = \sum_{\omega} \sum_{t} P(\omega|t)P(t|d) = \sum_{t} P(\omega|t) \sum_{\omega} P(\omega|t) = 1$$

Questa considerazione sarà utile nel seguito.

Sia ora C il corpus formato dai D documenti. La verosimiglianza $\mathcal{L}$ di C dato il modello è uguale alla probabilità di osservare C dati i valori delle distribuzioni di probabilità:

$$\mathcal{L} = P(\mathsf{C}|\theta, \Phi, \Psi) = \prod_{d}^{D} \prod_{\omega}^{\Omega} \prod_{i}^{n(d,\omega)} P(d, \omega) =$$
$$= \prod_{d}^{D} \prod_{\omega}^{\Omega} [P(d, \omega)]^{n(d,\omega)}$$

dove $n(d, \omega)$ è il numero di volte che il termine ω appare nel documento d.

Dal punto di vista operativo risulta comodo usare il logaritmo delle verosimiglianza; definiamo quindi:

$$\ell_1 = \log \mathcal{L} = \sum_{d} \sum_{\omega} n(d, \omega) \log \left[P(d) \sum_{t} P(\omega|t) P(t|d) \right]$$

e la forma equivalente:

$$\ell_2 = \log \mathcal{L} = \sum_{d} \sum_{\omega} n(d, \omega) \log \left[\sum_{t} P(t) P(t|\omega) P(t|d) \right].$$

6.1.3 Algoritmo Expectation-Maximization

Generalmente i valori di θ, Φ, Ψ non sono noti a priori ma vanno stimati a partire dai dati osservati (la matrice delle associazioni). In particolare si assegnare alle variabili i valori che massimizzano la funzione di (log-)verosimiglianza:

$$\langle \theta, \Phi, \Psi \rangle = \arg\max \mathcal{L}(\theta, \Phi, \Psi) = \arg\max \ell_1(\theta, \Phi, \Psi) = \arg\max \ell_2(\theta, \Phi, \Psi).$$

Il problema è intrattabile in via analitica poiché richiederebbe di massimizzare una funzione obiettivo con $T(D + \Omega + 1)$ argomenti sottoposta a $(D + \Omega + 1)$ vincoli.

Il problema può essere affrontato con l'algoritmo *EM (Expectation-Maximization)* [15]; la soluzione trovata da EM tuttavia è in generale solo un ottimo locale, non per forza anche globale.

L'algoritmo EM si compone di due passi:

a. **Passo E (Expectation):** si calcola il valore atteso della funzione di verosimiglianza dati i valori stimati delle variabili all'iterazione precedente come: $Q\left(\underline{x} \middle| \underline{x}^{(t)}\right) = \mathrm{E}_{o,x^{(t)}}[\mathcal{L}_o(\underline{x})]$, dove $\underline{x}$ sono le variabili da stimare e o i dati osservati;

b. **Passo M (Maximization):** si cerca una nuovo assegnamento di valori che minimizzi la quantità trovata nel passo precedente.

Questi due passi vengono iterati a partire da un assegnamento random alle variabili che si vogliono stimare; a seconda dell'assegnamento l'algoritmo convergerà verso un minimo locale differente, quindi potrebbe essere necessario provare più inizializzazioni differenti.

L'algoritmo termina quando tutte le variabili convergono e due passi successivi non producono nessun cambiamento negli assegnamenti; nella pratica però questa condizione potrebbe essere raggiunta dopo un numero eccessivo di iterazioni e per questa ragione di solito si implementa una delle due politiche di stop:

- Si esegue un numero fissato di passi;
- Si considera raggiunta la convergenza quando il rapporto tra due passi successivi è minore di un certo valore α.

In questa tesi ho scelto di implementare la seconda politica; più avanti mostrerò come ho determinato sperimentalmente un valore di α che fosse un buon compromesso tra numero di passi necessari alla convergenza e capacità di generalizzare del modello.

A seconda che si scelga la prima o la seconda formulazione di pLSA avremo diverse formule per calcolare i parametri del modello con l'algoritmo EM. Nella prima formulazione abbiamo un passo E:

$$P(t|\omega,d) = \frac{P(\omega|t)P(t|d)}{\sum_{t'} P(\omega|t')P(t'|d)}$$

e un passo M:

$$P(d) \propto \sum_{\omega}\sum_{t} n(d,\omega)P(t|\omega,d)$$
$$P(\omega|t) \propto \sum_{d} n(d,\omega)P(t|\omega,d)$$
$$P(t|d) \propto \sum_{\omega} n(d,\omega)P(t|\omega,d)$$

dove il simbolo $\propto$ indica la proporzionalità.

Nella seconda formulazione durante il passo E calcoliamo:

$$P(t|\omega,d) = \frac{P(t)P(\omega|t)P(d|t)}{\sum_{t'} P(t')P(\omega|t')P(d|t')}$$

e durante il passo M:

$$P(t) \propto \sum_{\omega} \sum_{d} n(d,\omega) P(t|\omega,d)$$
$$P(\omega|t) \propto \sum_{d} n(d,\omega) P(t|\omega,d)$$
$$P(d|t) \propto \sum_{\omega} n(d,\omega) P(t|\omega,d).$$

I simboli di proporzionalità indicano il fatto che dopo ogni iterazione dell'algoritmo EM sia necessario svolgere una operazione di normalizzazione, per rispettare i vincoli:

$$\forall\, t, \quad \sum_{\omega} P(\omega|t) = 1;$$
$$\forall\, t, \quad \sum_{d} P(d|t) = 1;$$
$$\forall\, d, \quad \sum_{t} P(t|d) = 1;$$
$$\sum_{t} P(t) = 1; \sum_{d} P(d) = 1;$$

perché i parametri devono rappresentare delle distribuzioni di probabilità.

6.1.4 pLSAnorm: una variante di pLSA

L'algoritmo pLSA è estremamente generale; in questo elaborato propongo una sua variante pensata per meglio adattarsi al caso delle predizioni di annotazioni biomolecolari [8].

La modifica rispetto alla versione tradizionale consiste in una normalizzazione delle righe rispetto al valore massimo, cioè una volta che si è calcolata la matrice approssimata delle probabilità congiunte, per ogni riga (gene) $\bar{g}$ di tale matrice si esegue questa trasformazione:

i. $M = \max_{\omega} P(\omega, \bar{g})$;

ii. Ogni elemento del vettore $\bar{g}$ viene moltiplicato per $\frac{1}{M}$.

Questa modifica è stata fatta perché, come si è visto in precedenza, la somma degli elementi di un vettore riga della matrice stimata delle probabilità condizionate è sempre uguale a 1; in questo modo i vettori (geni) con tante occorrenze (annotazioni) avranno un valore medio per elemento minore rispetto a quelli con poche occorrenze, ovvero sarebbero svantaggiati.

Si può notare come ora, nel caso si usi la formulazione asimmetrica sia superfluo stimare il valore delle probabilità $P(d)$ perché il contributo di questo fattore sparirebbe dopo la normalizzazione. Pertanto è lecito e comodo considerare tutti i documenti uniformemente distribuiti con probabilità:

$$P(d) = \frac{1}{D}.$$

6.1.5 Applicazione di pLSA

In questa sezione spiego come si intende applicare il metodo predittivo EM alla matrice delle annotazioni *unfoldata*.

Dividerò il procedimento in due parti: addestramento del modello e uso del modello.

Per prima cosa si specifica che ai documenti corrispondono i geni mentre ai termini corrispondono le funzionalità (chiamate anche *feature*).

Nella fase di addestramento viene usata la formulazione simmetrica per stimare tutte le probabilità $P(\omega|t) = P(feature\ |topic)$; l'insieme di queste probabilità condizionate costituisce il modello.

Nella fase di utilizzo, per ogni gene (documento) $\bar{g}$ si calcola la distribuzione di probabilità:

$$P(topic\ |\bar{g})$$

utilizzando, in questo caso, la formulazione asimmetrica di pLSA.

Si calcola quindi il valore della probabilità di una annotazione (si ricordi che si trascura la probabilità del gene-documento):

$$P(feature, \bar{g}) = \sum_{topic} P(\ feature|topic)P(topic|\bar{g}).$$

Il risultato viene poi normalizzato come descritto nella sezione precedente.

6.1.6 Problemi aperti

Sebbene in letteratura sia dimostrato come utilizzando pLSA si possano ottenere risultati migliori rispetto che con LSI, questa tecnica non è esente da problemi.

Innanzitutto è computazionalmente più oneroso applicare l'algoritmo EM piuttosto che eseguire la scomposizione SVD.

Inoltre pLSA può essere usato solamente con matrici i cui elementi siano tutti non negativi; quindi non si possono rappresentare le annotazioni che hanno *Qualifier* uguale a NOT, cioè quelle

annotazioni che significano che non esiste relazione tra l'entità biomolecolare e il termine. Questa limitazione non porta tuttavia grossi inconvenienti perché, come si vedrà dai dataset usati per i test, il numero di questo tipo di annotazioni è molto ridotto.
Il metodo pLSA ha anche problemi di *overfitting* (ovvero l'eccessivo adattamento del modello statistico ai dati osservati) , infatti il numero di parametri da stimare aumenta linearmente con il numero di *topics* utilizzato.
Inoltre questo algoritmo presenta alcune questioni non ancora risolte, descritte nel seguito.

6.1.5.1 Scelta del numero di topics

La questione aperta di maggiore rilievo è la scelta del numero di *topics*. Un numero troppo piccolo di variabili nascoste può comportare una bassa capacità di generalizzazione del modello; al contrario la scelta di un numero alto di *topics* allunga il tempo necessario alla convergenza dell'algoritmo EM. Inoltre se il numero di *topics* è maggiore di quello usato dal "reale" modello generativo dei dati si va incontro a problemi di *overfitting* infatti aumentano le variabili a cui si deve assegnare un valore.
Bisogna quindi cercare un numero di *topics* che sia il minore possibile ma sufficientemente grande per poter descrivere il dataset. Purtroppo ad oggi non si conosce nessun metodo per la selezione di questo parametro, ma la scelta deve essere fatta con procedimenti euristici.
Nel Capitolo **7** descriverò il metodo usato in questa tesi per la selezione di un numero di *topics* adeguato.

6.1.5.2 Inizializzazione

L'algoritmo EM è una specializzazione del noto algoritmo generale *Gradient Ascend* con il vantaggio che non è necessario calcolare il valore esatto della funzione obiettivo (si usa una stima) e non si deve scegliere la lunghezza del passo. Partendo da un valore dei parametri da ottimizzare iterativamente si sceglie un nuovo valore muovendosi nella direzione del gradiente della funzione di verosimiglianza.

Questo procedimento, come mostrato in Figura ha però un problema: la soluzione finale dipende fortemente dal valore della inizializzazione. Due inizializzazioni differenti possono portare a due ottimi locali distinti con due valori della funzione obiettivo anche molto diversa.

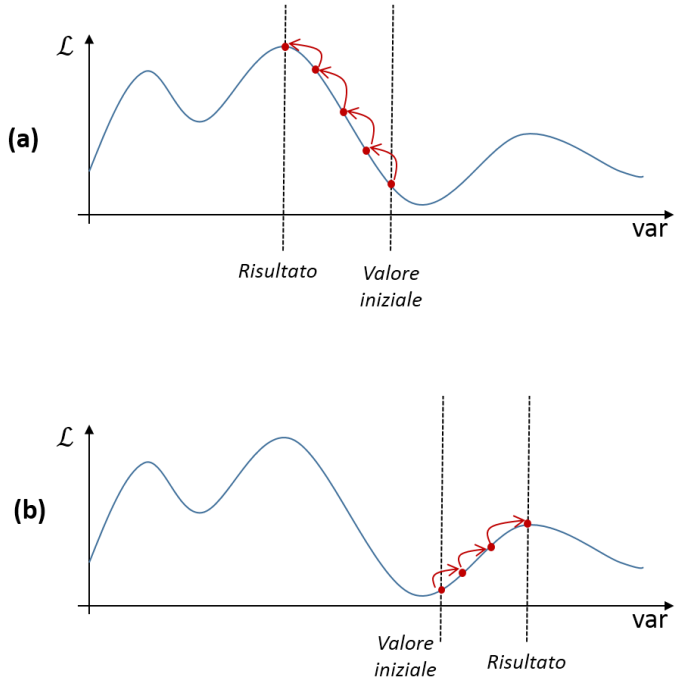


Figura 15: Esempio di applicazione dell'algoritmo EM; nell'esempio in figura l'assegnamento è ad una sola variabile. Partendo da un assegnamento casuale, ad ogni iterazione si trova una nuova soluzione migliore muovendosi nella direzione del gradiente della funzione obiettivo. Dai due esempi si può vedere come partendo da due inizializzazioni differenti si trovino due soluzioni finali con qualità diversa.

In pLSA questo problema si presenta in realtà due volte: una volta durante l'addestramento del modello e un'altra durante l'utilizzo. Non si conoscono buone prassi per la scelta di una inizializzazione, quindi si procederà generando casualmente il set iniziale di assegnamenti alle variabili.

6.1.7 Confronto tra LSI e pLSA

In questa sezione propongo un confronto tra i metodi LSI e pLSA ai fini di giustificare sotto il punto di vista teorico le migliori performance del nuovo metodo proposto.

Per generalità della trattazione si confronteranno i metodi LSI e NMF (Non-negative Matrix Factorization) mostrando come pLSA corrisponda a NMF a meno di una modifica della funzione obiettivo [9].

Sia LSI che NMF sono metodi per la decomposizione di una matrice, cioè, a meno di uno operazione di ri-normalizzazione, una matrice $X \in \mathbb{R}^{n \times m}$ viene riscritta come il prodotto:

$$X \cong AB^T$$

dove $A \in \mathbb{R}^{n \times r}$ e $B \in \mathbb{R}^{m \times r}$ sono matrici con rango minore o uguale a quello di X (quindi $r \leq \min\{n, m\}$) ed il simbolo $\cong$ indica che il loro prodotto è una approssimazione della matrice di partenza.

In caso di LSI le matrici A e B sono soluzione della equazione:

$$\min_{A,B} \| X - AB^T \|_F^2$$

dove la metrica usata è la distanza di Frobenius.

Nel caso di NMF con funzione obiettivo norma di Frobenius invece le matrici sono soluzione di:

$$\min_{A \geq 0, B \geq 0} \| X - AB^T \|_F^2.$$

Da questa ultima si può passare a pLSA tramite una modifica della funzione di costo; pLSA è soluzione della equazione:

$$\min_{A \geq 0, B \geq 0} KL(X \| AB^T)$$

dove:

$$KL(X \| Y) = \sum_{i,j} x_{i,j} \log \frac{x_{i,j}}{y_{i,j}}$$

è la norma di *Kullback-Leibler*.

È facile notare come tutte e tre le equazioni precedenti non abbiano una sola soluzione; infatti basta moltiplicare e dividere rispettivamente le matrici A e B per ottenere una nuova soluzione ugualmente valida.

Come già detto in precedenza, in LSI si sceglie generalmente una decomposizione con basi ortogonali ottenuta con la tecnica SVD, assegnanto:

$$A = U\Sigma, \qquad B = V.$$

In NMF, ed in particolar modo in pLSA, si tende invece a vincolare gli elementi di A e B ad avere somma unitaria, in modo da poter attribuire alla decomposizione una qualche interpretazione probabilistica.

Analizzando la matrice B di pLSA possiamo vedere che le sue colonne rappresentano delle distribuzioni di probabilità condizionate da una variabile nascosta e sono quindi vincolare ad avere somma unitaria. Possiamo vedere che i vettori colonna della matrice B:

- Giacciono su un simplesso;
- Appartengono all'ortante positivo.

Nel caso di LSI invece i vettori colonna di B:

- Sono versori;
- Sono ortogonali tra di loro.

In NMF si ha quindi un effetto di *clustering* sui vettori di B; questo permette di darne una interpretazione semantico, due vettori vicini avranno un significato simile.

In LSI invece questo non è possibile dove tutte le coppie di vettori hanno la stessa distanza.

Durante lo svolgimento di questa tesi ci si è chiesto se fosse possibile usare il modello pLSA direttamente come avviene in LSI, evitando di eseguire l'algoritmo EM, cioè applicando la formula:

$$\hat{x} = x\,\hat{B}\hat{B}^T$$

dove nel caso di LSI la matrice B corrisponde alla matrice V trovata con SVD e troncata al k-esimo livello, mentre in pLSA corrisponde alla matrice le cui colonne sono le distribuzioni di probabilità condizionate da una variabile aleatoria $P(\boldsymbol{term}|topic)$.

Come si è già visto in LSI questa operazione corrisponde a proiettare il vettore lungo le direzione della matrice $\hat{B}$ e a ricomporlo; siccome il livello di troncamento k è minore rispetto alla dimensione del vettore di input x questa operazione corrisponde a una perdita di informazione; tuttavia poiché le direzioni della matrice B sono ordinate per livello di sensibilità, l'informazione persa è quella riguardo alle relazioni documento-termine poco espresse nel dataset.

In pLSA, dove i vettori della matrice B ($\hat{B}$) non rappresentano delle direzioni e non sono ortogonali tra di loro, si può vedere che il modello non può essere applicato direttamente ma si è obbligati a ricorrere all'algoritmo EM. In generale:

$$x^T\hat{B}\hat{B}^T \neq p^T\hat{B}^T$$

dove p è la distribuzione di probabilità dal documento al *topic*, cioè il vettore $P(\boldsymbol{topic}\,|x)$, che minimizza la distanza di Kullback-Leibler:

$$KL(x^T \parallel p^T \hat{B}^T) = \sum_{term} x_{term} \log \frac{x_{term}}{\sum_{topic} p^T_{topic} \hat{B}^T_{term,topic}}. \quad \text{(Eq. 3)}$$

Infatti nel prodotto $x^T \hat{B} \hat{B}^T$ ogni elementox_i contribuisce sempre nello stesso modo alla determinazione di un valore $\hat{x}_j$, cioè un valore predetto $\hat{x}_j$ è la somma dei contributi dati da ogni valore in input x_i pesati con i valori della matrice B. Quindi il contributo dato da un elemento x_i non dipende dai valori degli altri elementi del x, infatti:

$$\hat{x}_j = \sum_{t=1}^{|x|} x_i * contributo(i,j).$$

Nel prodotto $p^T \hat{B}^T$, invece, il valore di un generico elemento p_i, essendo radice della derivata parziale della Eq.3 è funzione (non lineare) di tutti i valori del vettore x; in questo modo il contributo che un elemento x_i dà alla determinazione dei valori di $\hat{x}$ dipende anche dagli elementi $x_z, z \neq i$:

$$\hat{x}_j = \sum_{t=1}^{|topic|} p_t * contributo(t,j) = \sum_{t=1}^{|topic|} f_t(x_1, x_2, \dots, x_n) * contributo(t,j).$$

Questa differenza è alla base del maggiore potere espressivo di pLSA rispetto ad LSI, infatti permette di trattare con la polisemia perché l'interpretazione che si dà alla occorrenza di un termina all'interno di un documento dipende (anche) dal contesto in cui esso è inserito, cioè dagli altri termini del documento.

6.2 Schemi di peso

Fino ad ora abbiamo sempre usato una matrice ternaria (nel caso di LSI) o binaria (nel caso di pLSA). Questa rappresentazione però non permette di catturare al meglio le relazioni gerarchiche tra i termini delle ontologie. Precedenti studi nel campo della *Information Retrieval* hanno mostrato come l'uso di una rappresentazione a valori reali piuttosto che binaria o ternaria possa aumentare le prestazioni delle operazioni di estrazione di informazione dai dati.

Nell'ambito dell'analisi del linguaggio naturale, le intuizioni che portano all'uso di una rappresentazione più complessa sono:

- i termini che appaiono più spesso all'interno di un documento, sono candidati migliori a descrivere l'argomento del documento rispetto ai termini che appaiono raramente;
- i termini che appaiono poco frequentemente all'interno del corpus, cioè che appaiono in pochi documenti, sono migliori differenziatori tra i termini.

Ad esempio si pensi all'articolo *"the"*: esso apparirà nella maggior parte dei documenti in inglese, pertanto è un ottimo indicatore per la lingua del documento. Tuttavia, dato un corpus di documenti scritti in inglese, difficilmente aiuterà a distinguere l'argomento di uno specifico documento.
Simili relazioni potrebbero esistere tra i geni e le loro annotazioni: funzionalità che sono associate solo a pochi geni portano maggiori informazioni rispetto ai rispettivi geni e meglio li differenziano tra di loro; viceversa un termine annotato spesso ad un gene lo descrive meglio [10].
Quello che si vuole fare è creare delle funzioni con cui si pesa ogni associazione tra un gene e una funzionalità, in modo da dare maggiore rilevanza a quelle annotazioni che sono più importanti. Il nuovo valore di ogni elemento della matrice delle annotazioni sarà calcolato come:

$$gf_{i,j} * weight(\mathrm{C}, i, j).$$

Lo schema di peso è dunque un fattore moltiplicativo che dipende dal gene, dalla funzionalità ma anche dal corpus intero, ovvero dalle altre annotazioni.
Gli schemi proposti sono costruiti a partire da due grandezze: *g_freq*, "gene frequency" e *ia_freq*, "inverse annotation frequency".
La quantità *g_freq* è calcolata per ogni coppia gene-funzionalità e rappresenta il numero di volte che la funzionalità è annotata al gene. Può essere calcolata con il seguente algoritmo:

1) Si considera ogni vettore-gene g:
   a) Per ogni posizione $f = 1,2, \dots |feature|$ l'elemento $g\_freq_{g,f}$ viene inizializzato a 0;
   b) Per ogni funzionalità f tale per cui g è annotato ad f:
      i) Si genera il vettore-gene g_f in cui tutti gli elementi sono a zero, tranne lo f-esimo a cui viene assegnato il valore 1;
      ii) Si esegue l'operazione di *unfolding* partendo dal vettore g_f e si ottiene il nuovo vettore $g_{f,unfolded}$;
      iii) Per ogni posizione f tale per cui lo f-esimo elemento del gene $g_{f,unfolded}$ è positivo, l'elemento $g\_freq_{g,f}$ viene incrementato di 1.

Nella *Figura 14* un esempio dei applicazione dell'algoritmo; si può notare come i termini più

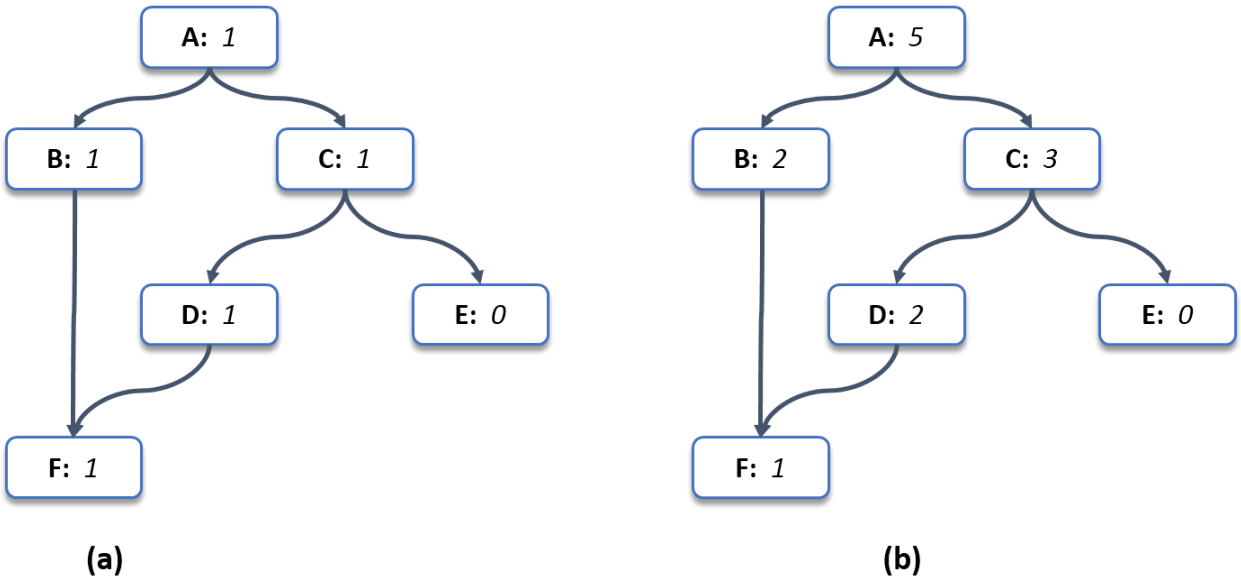


Figura 16: Esempio di calcolo dei valori di g_freq; (a) profilo di annotazioni di partenza, (b) valori calcolati di g_freq

generali abbiamo un valore maggiore di *g_freq* , questo cattura la natura gerarchica delle ontologie.

Per il fattore *ia_freq* si propongono due modi differenti di calcolo:

$$ia\_freq1\colon \forall f, \quad \text{ia\_freq1}_f = \frac{Numero\ totale\ di\ geni}{Numero\ di\ geni\ annotati\ ad\ f}$$

oppure:

$$ia\_freq2\colon \forall g, \quad \text{ia\_freq2}_f = \frac{Numero\ totale\ di\ annotazioni}{Numero\ di\ feature\ annotate\ a\ g}.$$

Il risultato dei due metodi è diverso; il primo tende a favorire le annotazioni che corrispondono a caratteristiche (feature) "rare" mentre il secondo le annotazioni che corrispondono a geni annotati a poche caratteristiche (feature). Verranno riportati i risultati di tutte e due le versioni.

Un schema di peso è creato a partire dalle quantità definite in precedenza, come composizione di tre contributi:

- Un peso locale, cioè che dipende solo dal gene considerato;
- Un peso globale, che dipende da tutta la matrice delle associazioni;

- Una normalizzazione, i valori della matrice delle annotazioni dopo la pesatura potrebbero non essere compresi tra 0 ed 1.

Gli schemi che verranno implementati sono derivati da quelli proposti da Draghici et al. nel 2006 [10].

6.2.1 Pesi locali

I pesi locali di basano sul valore di *g_freq*; ne vengono proposti due:

Maximum: il peso locale (lw) è la *g_freq* normalizzata rispetto al massimo delle *g_freq* nel vettore (gene) considerato:

$$lw = \frac{\text{g\_freq}}{\max\{\text{g\_freq nel gene}\}}.$$

Augmented: il peso locale (lw) è calcolato il funzione della *g_freq* come segue:

$$lw = 0.5 + 0.5 * \frac{\text{g\_freq}}{\max\{\text{g\_freq nel gene}\}}.$$

6.2.2 Pesi globali

I pesi globali forniscono informazione sulla rilevanza della annotazione all'interno del corpus. Un solo schema viene proposto, il peso globale (gl) corrisponde al valore di *ia_freq* relativo alla associazione.

6.2.3 Normalizzazione

Il prodotto dei pesi locale e globale viene normalizzato per riportare i valori all'interno dell'intervallo [0,1]. Due diversi metodi di normalizzazione sono proposti:

Maximum: i valori della funzione di peso vengono normalizzati rispetto al massimo del vettore-gene:

$$weight_{normalized} = \frac{weight}{\max\{weight\ nel\ gene\}}$$

Cosine: i valori sono normalizzati come segue:

$$weight_{normalized} = \frac{weight}{\sqrt{sommatoria\ del\ quadrato\ dei\ pesi\ relativi\ al\ gene}}$$

6.2.4 Composizione degli schemi

Ogni schema di peso è rappresentato da tre lettere: la prima indica il tipo di peso locale scelto, la seconda il peso globale mentre la terza la normalizzazione. Le corrispondenze tra lettere e tipo di peso sono riportate nella Tabella:

Pesi locali	
N	*Nessuno schema*
M	*Maximum*
A	*Augmented*
Pesi Globali	
T	*Frequenza inversa*
Normalizzazione	
N	*Nessuno schema*
M	*Maximum*
C	*Cosine*

Si possono creare quindi un totale di 9 schemi: *ntn*, *ntm*, *ntc*, *mtn*, *mtm*, *mtc*, *atn*, *atm* e *atc*.
Si noti che tuttavia questi corrispondono a solo 7 schemi differenti perché gli schemi *ntm* e *ntc* sono identici rispettivamente agli schemi *mtm* e *mtc*.

7 Validazione

In questo capitolo descriverò le diverse tecniche di validazione che è possibile utilizzare per valutare la bontà degli algoritmi predittivi proposti.

7.1 Categorie di annotazioni

Poiché le banche dati di annotazioni non possono mai essere considerate complete e definitive non possiamo usare categorizzare le annotazioni predette come veri o falsi positivi (TP o FP) oppure come veri o falsi negativi (TN o FN). Ad esempio, di una predizione che non trova conferma nel database di riferimento, non possiamo affermare con certezza che si tratti di un falso positivo, infatti potrebbe essere valida biologicamente ma non essere ancora stata inserita nella banca dati. Per questo motivo nel seguito useremo quattro altre categorie, simili a quelle sopra citate ma con un significato leggermente diverso:

- AP, *Annotation Predicted*, anziché FP per le annotazioni non presenti in input ma predette in output;
- NAC, *No Annotation Confirmed*, anziché TN, non presenti in input e non presenti in output;
- AC, *Annotation Confirmed*, anziché TP, per le annotazioni presenti in input e predette;
- AR, *Annotation to be Reviewed*, anziché FN, per le annotazioni presenti in input ma non in output.

7.2 Validazione

La validazione è una tecnica statistica utilizzata nell'ambito dei metodi predittivi per valutare l'accuratezza delle predizioni di un modello. Generalmente una procedura di validazione si compone di due passi:

1. Addestramento del modello: a partire da un insieme di dati conosciuti si crea il modello (ovvero si stimano i valori delle variabili);
2. Utilizzo del modello; si usa il modello generato al passo precedente per predire il valore di un insieme di dati diversi da quelli usati per l'addestramento.

Nel seguito propongo tre diverse tipologie di validazione che corrispondono a tre diverse tipologie del secondo passo.

7.2.1 Validazione tramite curve ROC

Questo tipo di validazione è il più semplice da applicare perché richiede di avere una sola fonte di dati.

Come detto in precedenza non è ancora conosciuto nessun modo per calcolare a priori il valore di alcuni ottimale parametri come, ad esempio, il numero di *topics* in pLSA. Questa metodologia di validazione può essere utilizzata per cercare di stimare un buon valore per questi parametri.

La tecnica scelta è la *K-Fold Cross-Validation* in cui l'insieme dei geni viene diviso in K sottoinsiemi ortogonali e, ad ogni iterazione, si addestra il modello su $K-1$ di essi e lo si testa sul rimanente.

Nel seguito i dettagli della procedura implementata. Partendo dalla matrice delle annotazioni *unfoldata* A:

1. Si considerano solo le colonne di A con almeno M (3) elementi pari a 1 e, dopo l'eliminazione delle colonne con meno di M elementi, solo le righe di A con almeno L (1) elementi pari a 1, si prende un insieme di K (10) sottoinsiemi random (ovvero ove ogni riga allocata ad ogni sottoinsieme è selezionata in modo random tra le righe della matrice A, evitando così di prendere set di righe tra loro consecutive) di uguale dimensione di righe (geni) di matrice A, di cui di volta in volta $K-1$ (9) insiemi verranno usati come *training set*, mentre 1 insieme verrà usato come *validation set*;
2. Viene addestrato il modello sul *training set*;
3. Per quanto concerne il k-esimo sottoinsieme, il *validation set*, si procede come segue:
    a. Viene eseguito il metodo predittivo su ogni riga dell'intero *validation set* per predire il valore degli elementi $x_{i,j}$ del *validation set* che hanno valore 0 in matrice A;
    b. Si predice separatamente come segue il valore di ognuno degli elementi $x_{i,j}$ del *validation set* che hanno valore 1 in matrice A (sostituendo il loro valore predetto al punto precedente). Per ogni elemento $x_{i,j}$ dell'insieme, corrispondente

all'associazione tra riga (gene) i e colonna (termine) j, che contiene un 1 (associazione presente):

i. Viene messo a 0 il valore dell'elemento $x_{i,j}$ (associazione cancellata) e di tutti gli elementi $x_{i,t}$ della stessa riga (gene) i corrispondenti ad associazioni tra riga (gene) i e colonne (termini) t parenti (antenati e discendenti) della colonna (termine) j a cui appartiene l'elemento $x_{i,j}$ in esame;

ii. Si (ri)mettono a 1 gli elementi $x_{i,t}$ relativi a colonne (termini) k antenati di colonne (termini) che hanno elementi in riga i contenenti uno 0 dopo cancellazione al punto precedente;

iii. Se dopo queste modifiche ai due punti precedenti la riga i contiene almeno 1 elemento a 1, viene eseguito il metodo predittivo sulla riga i ;

iv. Viene fatta la correzione (*from leaves*) delle anomalie per la riga i;

v. Si considera il valore predetto per elemento $x_{i,t}$ risultante dopo correzione anomalie e lo si prende come valore predetto finale per l'elemento $x_{i,t}$sostituendolo al valore di tale elemento risultante al punto *(a);*

4. Si ripetono le operazioni ai passi *(1)* e *(2)* usando ogni volta uno dei diversi *K* sottoinsiemi come *validation set*;
5. Viene creata la matrice totale delle predizioni $\hat{A}$ come unione delle predizioni sui *K validation set*;
6. Su ogni riga di $\hat{A}$ viene eseguita la correzione delle anomalie (*from leaves*);
7. Per ogni valore di τ compreso tra 0 ed 1, con risoluzione 10^{-4}:
    a. Viene binarizzata la matrice delle predizioni considerando pari ad 1 tutti i valori maggiori della soglia e pari a 0 gli altri;
    b. Confrontando la matrice $\hat{A}$ binarizzata con la matrice *A* ci calcolano AP, NAC, AC, AR;
    c. Si calcolano i punti della curva *ROC (Receiver Operating Characteristic)* del rapporto tra frequenza di AP (*APrate*) e frequenza di AR (*ARrate*), dove le due grandezze sono calcolate come segue:

$$\begin{cases} APrate = \dfrac{AP}{AP + NAC} \\ ARrate = \dfrac{AR}{AR + AC} \end{cases}$$

d. Si disegna la curva per valori di *APRate* compresi nell'intervallo $[0;\ 0.01]$ e si calcola il valore dell'area sottesa.

Il confronto tra prestazioni dei modelli è fatto sulla base del valore dell'area; di considerano migliori modelli con un valore di area minore. Infatti un valore piccolo di area sottesa alla curva ROC significa che abbiamo valori alti di *NAC* e *AC*, cioè troviamo tante conferme ai dati predetti.

7.2.2 Validazione su più database

Questo secondo tipo di validazione richiede due differenti versioni del database delle annotazioni, aggiornate a due istanti di tempo distinti; è possibile sfruttare le diverse versioni dei database memorizzate all'interno del data warehouse GPWD. La procedura è la seguente:

1. Dal database meno aggiornato si estrae la matrice delle annotazioni *unfoldata* A_1;
2. Partendo dalle informazioni contenute nella matrice A_1 si addestra il modello predittivo;
3. Si utilizza il modello predittivo per calcolare la matrice $\hat{A}_1$, cioè la matrice delle predizioni delle nuove annotazioni relative ai geni e alle funzionalità contenute nel database non aggiornato;
4. Si interroga il database aggiornato per ottenere la matrice A_2 delle annotazioni *unfoldata* rappresentate all'interno del database; questa nuova matrice dovrebbe contenere le relazioni tra gli stessi geni e le stesse funzionalità rappresentate nella matrice A_1, ma questo è raramente possibile perché sia la struttura della terminologia che l'insieme dei geni sono soggetti a cambiamenti nel corso del tempo;
5. La matrice A_1 viene binarizzata scegliendo un opportuno valore di soglia;
6. Ogni valore predetto dall'algoritmo, cioè ogni coppia gene-feature con valore sopra la soglia nella matrice $\hat{A}_1$ ma nullo in A_1, viene confrontato con le annotazioni presenti nella matrice aggiornata A_2; possiamo distinguere tra quattro casi:

- Confronto non possibile: significa che non è stato possibile trovare una corrispondenza tra il gene o la caratteristica dell'annotazione predetta nel database aggiornato; infatti i termini presenti in una terminologia sono soggetti a cambiamenti e le entità biomolecolari possono essere dichiarate obsolete e tolte dal database;
- Predizione non confermata: nel database aggiornato sono rappresentate sia l'entità biomolecolare che la funzionalità della annotazione predetta, ma non sono annotate tra di loro;
- Predizione confermata con evidenza IEA: sul database aggiornato è presenta la annotazione ma con grado di evidenza IEA;
- Predizione confermata: nel database aggiornato è presente la annotazione con un grado di evidenza migliore rispetto a IEA.

Questo secondo tipo di validazione è meno immediato rispetto al precedente e richiede la disponibilità di una quantità maggiore di informazioni; inoltre non permette di assegnare facilmente una valutazione numerica della qualità di un modello, come si poteva fare con l'area della curva ROC. Tuttavia i risultati ottenuti con questa seconda procedura sono più realistici; infatti una annotazione predetta non presente nel database iniziale ma biologicamente valida (cioè il risultato auspicabile di un metodo predittivo) nel caso della prima validazione verrà sempre considerata un errore mentre con questa procedura potrebbe essere riconosciuta come corretta. Ovviamente il risultato di questa validazione dipende dalla qualità del database aggiornato e il fatto di non trovare la conferma di una predizione non implica automaticamente che essa sia sbagliata ma potrebbe semplicemente non essere ancora stata aggiunta.

7.2.3 Validazione nella letteratura

Questo terzo tipo di validazione prevede di generare una lista di predizioni partendo da un database il più aggiornato possibile e di controllare manualmente ciascuna di queste annotazioni per cercare se siano trattate in letteratura e se siano state confermate o smentite. Infatti è verosimile aspettarsi che la maggior parte delle informazioni biologiche disponibili non sia ancora stata annotata ad una ontologia.

Questa procedura è sicuramente la più affidabile e quella che permette di ottenere i risultati migliori per la valutazione di un metodo predittivo, presenta però delle problematiche; infatti non può essere facilmente automatizzata ed è molto dispendiosa in termini di tempo. Ancora una volta l'assenza di informazioni in letteratura riguardo ad una relazione tra un'entità biomolecolare e una funzionalità non indica automaticamente un errore di predizione.

8 Risultati

In questo capitolo riporto i risultati di alcuni test eseguiti sui metodi predittivi implementati usando come input le informazioni contenute all'interno del data warehouse GPDW.

8.1 Stima dei parametri del modello pLSA

Nelle due sezioni successive mostrerò come si possa usare il primo metodo di validazione per determinare in via empirica il valore di due parametri dell'algoritmo pLSA: la soglia di stop dell'algoritmo EM e il numero di *topics*.

8.1.1 Stima della soglia di stop per l'algoritmo EM

I due passi dell'algoritmo EM vengono ripetuti iterativamente fino a che non si raggiunge la convergenza, cioè fino a che due iterazioni successive dell'algoritmo non portano cambiamenti nella struttura dati. Tuttavia nella pratica questo può risultare infattibile perché la convergenza potrebbe essere raggiunta dopo un numero eccessivo di passi o addirittura non essere mai raggiunta, considerate le approssimazioni delle rappresentazioni dei numeri reali sul calcolatore. Pertanto si è scelto di implementare la seguente politica di stop dell'algoritmo EM: si considera che l'algoritmo EM ha raggiunto la convergenza quando, per due passi successivi k e $k+1$, si ha:

$$\frac{\log \mathcal{L}^{(k)}}{\log \mathcal{L}^{(k+1)}} < (1+\alpha)$$

in cui il simbolo $\mathcal{L}$ rappresenta la funzione di verosimiglianza definiti a pag. 40.

Bisogna notare come non è detto che questa condizione assicuri che si sia vicini alla convergenza e nelle iterazioni successive la soglia potrebbe essere superata nuovamente; comunque, come si vedrà nei risultati, questa politica è sensata e, per i dataset testati, ha offerto buoni risultati.

Il problema si sposta quindi su come determinare un valore di α che offra un buon compromesso tra il tempo di esecuzione dell'algoritmo (cioè sul numero di iterazioni necessarie a raggiungere la condizione di stop) e la qualità del modello.

8.1.1.1 Descrizione esperimento

1. Si sceglie un dataset ed un numero di *topics*;
2. Si inizializza il modello in modo casuale e si applica l'algoritmo EM;
3. Per ogni valore di α^{-i} con $i = 1,2,\dots,10,12,15,20$ la prima volta che si raggiunge la condizione di stop $\frac{\log \mathcal{L}^{(k)}}{\log \mathcal{L}^{(k+1)}} < (1+\alpha)$ si simula lo stop dell'algoritmo e si memorizzano:
   - Il numero $k+1$ di iterazioni fino a quel punto;
   - La log-verosimiglianza del modello calcolata dopo le prime $k+1$ iterazioni;
   - La distanza (norma) tra le matrici $P(feature|topic)$ calcolata tra due soglie di stop consecutive.

Di seguito si riportano i risultati dell'esperimento.

8.1.1.2 Risultati

I dati relativi ai dataset sono riportati in appendice.

Tabella 2: Risultati con tassonomia 9031 GallusGallus, ontologia 6 CellularComponent, 104 topics = 40% numero geni.

GallusGallus(9031) - Cellular Component - topics = 104					
α	iterazioni	$log(L)$	incremento iter.	incremento L	distanza $P(f\|t)$
10^{-1}	2	-13 688.9	–	–	–
10^{-2}	3	-13 613.6	50.000%	0.550%	0.029
10^{-3}	17	-12 275.5	466.667%	9.829%	12.467
10^{-4}	33	-12 215.6	94.118%	0.487%	1.452
10^{-5}	60	-12 206.7	81.818%	0.073%	0.539
10^{-6}	111	-12 204.2	85.000%	0.021%	0.226
10^{-7}	231	-12 202.7	108.108%	0.012%	0.471
10^{-8}	268	-12 202.4	16.017%	0.002%	0.077
10^{-9}	279	-12 202.4	4.104%	>0.001%	>0.001
10^{-10}	301	-12 202.4	7.885%	>0.001%	>0.001
10^{-12}	309	-12 202.4	2.658%	>0.001%	>0.001
10^{-15}	313	-12 202.4	1.294%	>0.001%	>0.001
10^{-20}	476	-12 201.5	52.077%	0.007%	0.141

Tabella 3:: Risultati con tassonomia 9031 GallusGallus, ontologia 7 MolecularFunction, 124 topics = 40% numero geni.

GallusGallus(9031) - Molecular Function - topics = 124					
α	iterazioni	$log(L)$	incremento iter.	incremento L	distanza $P(f\|t)$
10^{-1}	2	-7 698.8	–	–	–
10^{-2}	9	-6 483.1	350.000%	15.782%	15.836
10^{-3}	17	-6 357.0	88.889%	1.956%	5.003
10^{-4}	29	-6 335.1	70.588%	0.344%	1.407
10^{-5}	52	-6 329.2	79.310%	0.094%	0.471
10^{-6}	138	-6 325.6	165.385%	0.057%	0.964
10^{-7}	160	-6 325.5	15.942%	0.001%	0.003
10^{-8}	214	-6 325.5	33.750%	>0.001%	0.004
10^{-9}	228	-6 325.5	6.542%	>0.001%	>0.001
10^{-10}	233	-6 325.5	2.192%	>0.001%	>0.001
10^{-12}	238	-6 325.5	2.146%	>0.001%	>0.001
10^{-15}	256	-6 325.5	7.563%	>0.001%	>0.001
10^{-20}	395	-6 325.2	54.297%	0.005%	0.384

Tabella 4: : Risultati con tassonomia 9913 BosTaurus, ontologia 6 CellularComponent, 99 topics = 40% numero geni.

GallusGallus(9031) - Biological Process - topics = 110					
α	iterazioni	$log(L)$	incremento iter.	incremento L	distanza $P(f\|t)$
10^{-1}	1	-41 434.6	–	–	–
10^{-2}	2	-41 301.7	100.000%	0.321%	0.005
10^{-3}	19	-35 139.9	850.000%	14.919%	6.620
10^{-4}	41	-34 910.4	115.789%	0.653%	1.047
10^{-5}	118	-34 817.1	187.805%	0.267%	0.828
10^{-6}	379	-34 779.5	221.186%	0.108%	0.615
10^{-7}	797	-34 772.5	110.29%	0.020%	0.186
10^{-8}	913	-34 772.0	14.554%	0.001%	0.018
10^{-9}	1253	-34 770.8	37.240%	0.003%	0.027
10^{-10}	1255	-34 770.8	0.160%	>0.001%	>0.001
10^{-12}	1270	-34 770.8	1.195%	>0.001%	>0.001
10^{-15}	1378	-34 769.6	8.503%	0.004%	0.007
10^{-20}	1382	-34 769.6	0.290%	>0.001%	>0.001

Tabella 5: Risultati con tassonomia 9913 BosTaurus, ontologia 6 CellularComponent, 99 topics = 40% numero geni.

BosTaurus(9913) - Cellular Component - topics = 99					
α	iterazioni	$log(L)$	incremento iter.	incremento L	distanza $P(f\|t)$
10^{-1}	2	-33 726.9	–	–	–
10^{-2}	3	-33 616.8	50.000%	0.327%	0.218
10^{-3}	16	-29 848.4	433.333%	11.201%	135.371
10^{-4}	26	-29 767.5	62.500%	0.271%	6.445
10^{-5}	45	-29 750.7	73.077%	0.056%	2.183
10^{-6}	79	-29 747.2	75.555%	0.012%	1.011
10^{-7}	130	-29 746.1	64.557%	0.004%	0.301
10^{-8}	144	-29 746.0	10.769%	>0.001%	0.005
10^{-9}	148	-29 746.0	2.778%	>0.001%	>0.001
10^{-10}	157	-29 746.0	6.081%	>0.001%	0.001
10^{-12}	164	-29 746.0	4.459%	>0.001%	0.001
10^{-15}	168	-29 746.0	2.439%	>0.001%	>0.001
10^{-20}	176	-29 746.0	4.7619%	>0.001%	0.003

Tabella 6: Risultati con tassonomia 9913 BosTaurus, ontologia 7 MolecularFunction, 207 topics = 40% numero geni.

BosTaurus(9913) - Molecular Function - topics = 207					
α	iterazioni	$log(L)$	incremento iter.	incremento L	distanza $P(f\|t)$
10^{-1}	2	-15 777.6	–	–	–
10^{-2}	10	-12 939.1	400.000%	17.990%	36.350
10^{-3}	17	-12 752.9	70.000%	1.439%	6.947
10^{-4}	29	-12 709.3	70.588%	0.341%	2.277
10^{-5}	71	-12 686.2	144.828%	0.181%	2.698
10^{-6}	149	-12 681.9	109.859%	0.034%	1.204
10^{-7}	331	-12 674.2	122.148%	0.061%	1.421
10^{-8}	345	-12 674.1	4.229%	>0.001%	0.001
10^{-9}	352	-12 674.1	2.029%	>0.001%	>0.001
10^{-10}	361	-12 674.1	2.557%	>0.001%	>0.001
10^{-12}	363	-12 674.1	0.554%	>0.001%	>0.001
10^{-15}	368	-12 674.1	1.377%	>0.001%	>0.001
10^{-20}	375	-12 674.1	1.902%	>0.001%	>0.001

Tabella 7: Risultati con tassonomia 9913 GallusGallus, ontologia 8 BiologicalProcess, 205 topics = 40% numero geni.

BosTaurus(9913) - Biological Process - topics = 205					
α	iterazioni	$log(L)$	incremento iter.	incremento L	distanza $P(f\|t)$
10^{-1}	2	-93 773.8	–	–	–
10^{-2}	3	-93 338.7	50.000%	0.464%	0.011
10^{-3}	20	-78 932.8	566.667%	15.434%	14.349
10^{-4}	45	-78 400.6	125.000%	0.674%	2.691
10^{-5}	118	-78 204.0	162.222%	0.251%	1.392
10^{-6}	321	-78 127.4	172.034%	0.098%	0.973
10^{-7}	803	-78 086.8	150.156%	0.052%	0.690
10^{-8}	1012	-78 078.4	26.027%	0.011%	0.115
10^{-9}	1121	-78 076.0	10.771%	0.003%	0.034
10^{-10}	1140	-78 075.9	1.695%	>0.001%	>0.001
10^{-12}	1202	-78 074.6	5.439%	0.002%	0.012
10^{-15}	1207	-78 074.6	0.416%	>0.001%	>0.001
10^{-20}	1212	-78 074.6	0.414%	>0.001%	>0.001

8.1.1.3 Conclusioni

Analizzando i risultati dei test si può vedere che in nessuno di essi una soglia con $\alpha = 10^{-15}$ è stata sufficiente a raggiungere la convergenza. Tuttavia il con una soglia $\alpha = 10^{-5}$ si ottengo dei risultati abbastanza stabili: diminuendo fino a 10^{-20} non si ottiene mai un miglioramento della log-verosimiglianza superiore allo 0.2%, mentre il numero di iterazioni continua a crescere; dunque usare una soglia $\alpha = 10^{-5}$ sembra un buon compromesso tra prestazioni e qualità del modello.

8.1.2 Numero di *topics*

La scelta del numero di *topics* è cruciale nell'applicazione dell'algoritmo pLSA. L'obiettivo è quello di usare un numero di *topics* il più piccolo possibile (per aumentare le performance dell'algoritmo e per ridurre l'*overfitting*) ma sufficientemente grande da dare al modello un buon potere predittivo.

Iniziamo facendo una considerazione preliminare: come mostrato in precedenza, un *topic* corrisponde ad un vettore dello spazio F-dimensionale (dove F è il numero di funzionalità del dataset). Il vettore delle annotazioni predette è anche esso un vettore F-dimensionale ed è combinazione lineare dei vettori corrispondenti ai *topics*; cioè ogni vettore predetto è contenuto in un cono di massimo F dimensioni. In Figura 15 un esempio nel caso di due sole funzionalità.

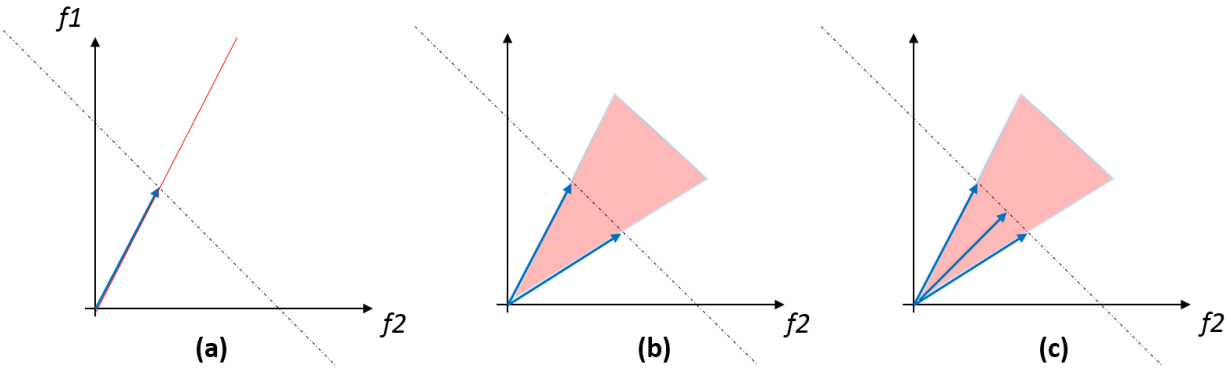


Figura 17: Caso con due funzionalità: in azzurro i vettori-topic, in rosso lo spazio delle predizioni; (a) numero di topics minore del numero di funzionalità; (b) numero di topics pari al numero di funzionalità; (c) numero di topics maggiore del numero di funzionalità

Nel primo caso, utilizzando un solo *topic*, lo spazio delle predizioni è un retta; se invece usiamo un numero di *topics* pari al numero di caratteristiche (*feature*) otteniamo uno spazio uguale a un cono di F dimensioni; aumentando ulteriormente il numero di *topics* non aumentiamo la dimensione dello spazio delle predizioni perché uno dei tre *topics* è combinazione lineare a coefficienti positivi degli altri due.

In base a queste osservazioni sembra sensato scegliere un numero di *topics* minore o uguale al numero di funzionalità; andando oltre non si guadagna nulla in termini di espressività del modello ma si introducono parametri in più e questo aumenta gli effetti dell'*overfitting* durante la selezione dei *topics*.

Nel seguente esperimento faremo variare il numero di *topics* sull'intervallo $[1;\ F]$ per cercare di determinare una euristica per stabilire il migliore valore di questo parametro. Andremo anche oltre per mostrare come non si abbiano più grandi miglioramenti; le piccole variazioni possono essere giustificate oltre che dalle fluttuazioni statistiche del modello dall'*overfitting* generato dal numero eccessivo di parametri che si tenta di stimare durante la fase di validazione.

8.1.2.1 Descrizione dell'esperimento

L'esperimento è molto semplice e prevede di ripetere la procedura di validazione su una matrice delle annotazioni, scegliendo ad ogni iterazione un numero diverso di *topics* diverso. Nel seguito l'insieme di numeri di *topics* usato sarà:

$$T = \{t = k\%\ di\ F \mid k = 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1.0, 1.25\}$$

Per ogni dataset, oltre a tutti i parametri della interrogazione con cui lo si ottiene, relativamente ad ogni valore di *topics* nell'insieme T, si memorizzano:

- L'area sotto la curva ROC (AUC, Area Under Curve);
- Il tempo medio di addestramento del modello (cioè il tempo per creare il modello a partire da nove decimi del dataset);
- Il tempo medio di utilizzo del modello per la predizione delle annotazioni di un singolo gene (non incluso nel *training set*).

8.1.2.2 Risultati

Tabella 8: Confronti valore AUC ROC al variare del numero di topic con dataset Gallus Gallus – Cellular Component

Gallus Gallus (9031) – Cellular Component (6)				
%	**# *topics***	**AUC**	**Avg train time [ms]**	**Avg test time [ms]**
10	12	*0.00559114*	*142*	*1*
20	24	*0.00477404*	*233*	*1*
30	36	*0.00425451*	*276*	*2*
40	49	*0.00421041*	*327*	*3*
50	61	*0.00347376*	*344*	*4*
75	92	*0.00355824*	*501*	*5*
100	123	*0.00372164*	*554*	*7*
125	153	*0.00364020*	*658*	*8*

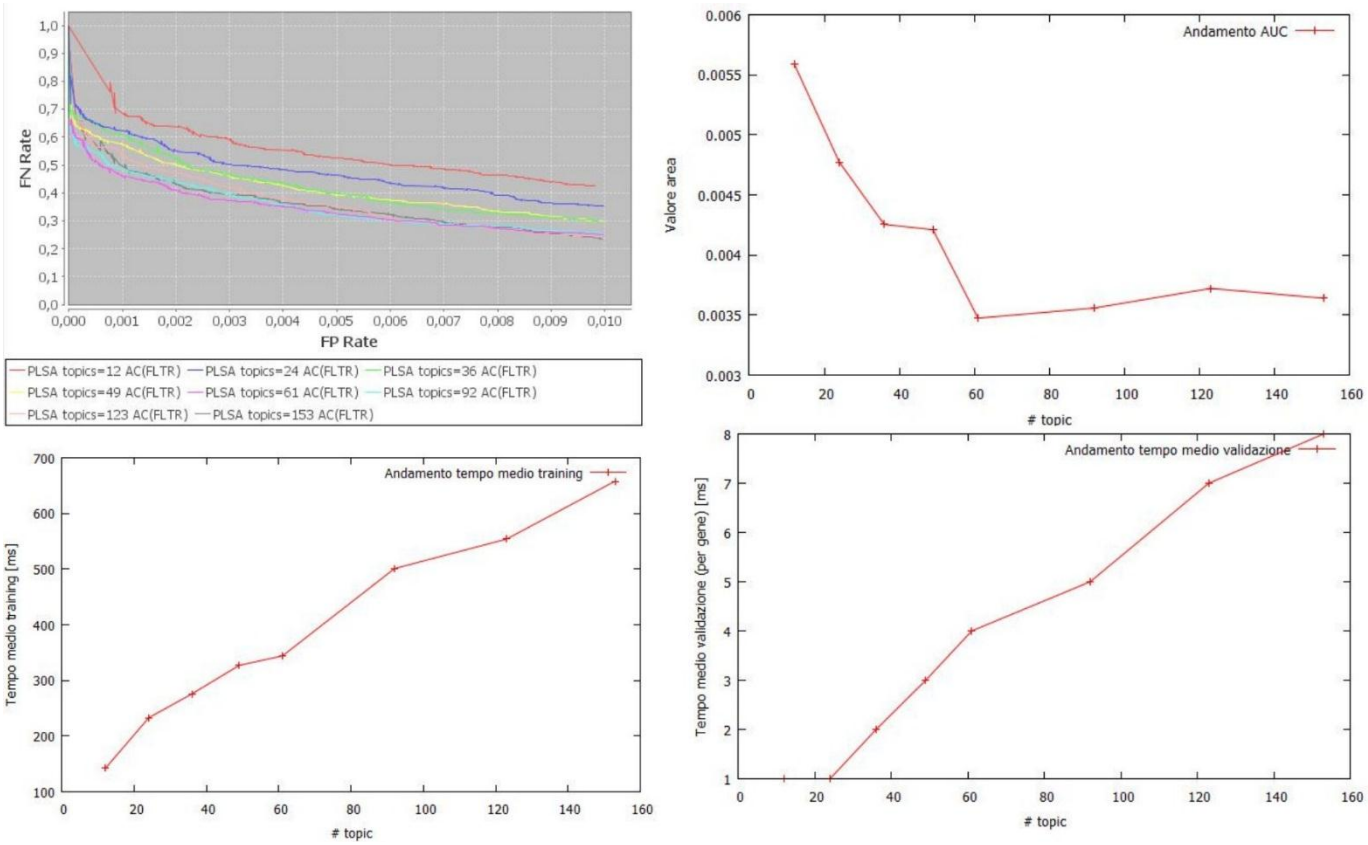


Figura 18: In alto a sinistra le curve ROC ottenute, in alto a destra l'andamento dell'area delle curve ROC in funzione del numero di topics, in basso a sinistra l'andamento del tempo medio necessario all'addestramento del modello in funzione del numero di topics, in basso a destra l'andamento medio del tempo necessario a produrre le predizioni su un gene. I tempi sono riportati in millisecondi.

Tabella 9: Confronti valore AUC ROC al variare del numero di topic con dataset Gallus Gallus – Molecular Function

Gallus Gallus (9031) – Molecular Function (7)				
%	**# *topics***	**AUC**	**Avg train time [ms]**	**Avg test time [ms]**
10	13	*0.00583859*	*148*	*1*
20	26	*0.00501575*	*230*	*2*
30	40	*0.00430892*	*326*	*3*
40	53	*0.00424737*	*328*	*3*
50	67	*0.00407294*	*384*	*4*
75	100	*0.00363574*	*483*	*5*
100	134	*0.00348746*	*595*	*6*
125	167	*0.00341166*	*639*	*9*

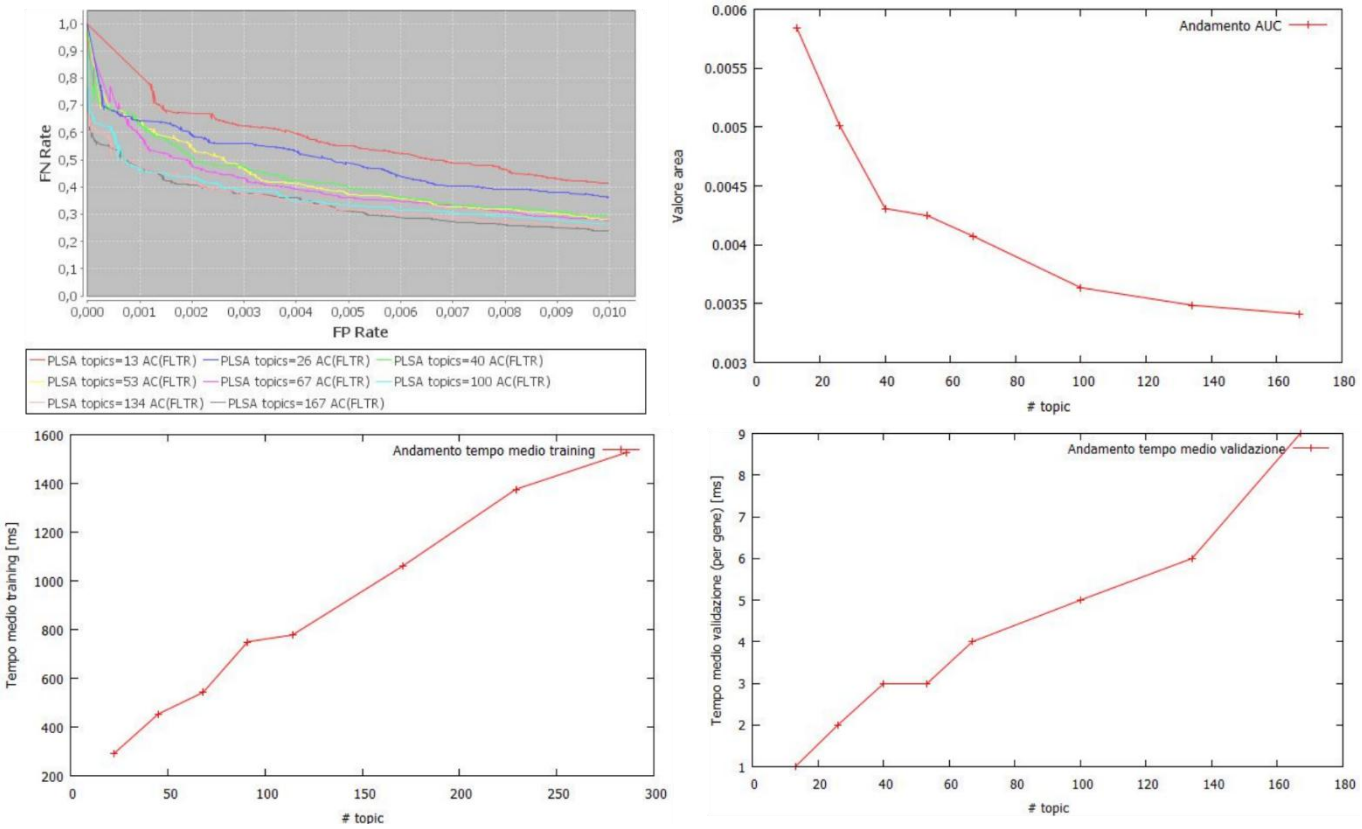


Figura 19: In alto a sinistra le curve ROC ottenute, in alto a destra l'andamento dell'area delle curve ROC in funzione del numero di topics, in basso a sinistra l'andamento del tempo medio necessario all'addestramento del modello in funzione del numero di topics, in basso a destra l'andamento medio del tempo necessario a produrre le predizioni su un gene. I tempi sono riportati in millisecondi.

Tabella 10: Confronti valore AUC ROC al variare del numero di topic con Gallus Gallus - Biological Process

Gallus Gallus (9031) – Biological Process (8)				
%	**# *topics***	**AUC**	**Avg train time [ms]**	**Avg test time [ms]**
10	61	*0.00541549*	*1 097*	*59*
20	122	*0.00427957*	*2 129*	*89*
30	183	*0.00409875*	*2 749*	*142*
40	244	*0.00397668*	*3 680*	*142*
50	305	*0.00389732*	*4 619*	*169*
75	457	*0.00368590*	*6 535*	*276*
100	610	*0.00379770*	*8 076*	*484*
125	762	*0.00356514*	*9 248*	*969*

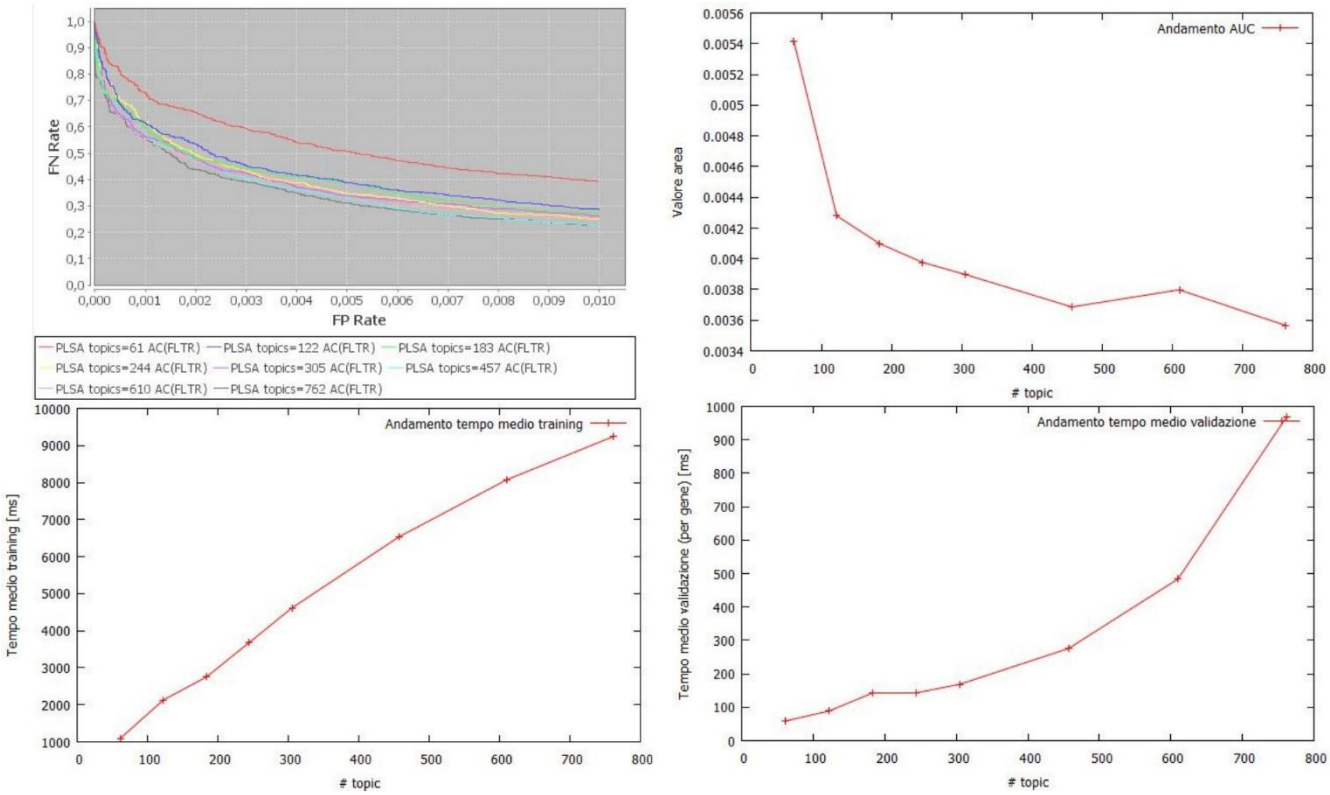


Figura 20: In alto a sinistra le curve ROC ottenute, in alto a destra l'andamento dell'area delle curve ROC in funzione del numero di topics, in basso a sinistra l'andamento del tempo medio necessario all'addestramento del modello in funzione del numero di topics, in basso a destra l'andamento medio del tempo necessario a produrre le predizioni su un gene. I tempi sono riportati in millisecondi.

Tabella 11: Confronti valore AUC ROC al variare del numero di topic con dataset BosTaurus - Cellular Component

Bos Taurus (9913) – Cellular Component (6)				
%	**# *topics***	**AUC**	**Avg train time [ms]**	**Avg test time [ms]**
10	20	*0.00428172*	*334*	*4*
20	41	*0.00373556*	*614*	*5*
30	62	*0.00333649*	*876*	*7*
40	82	*0.00314547*	*987*	*8*
50	103	*0.00310969*	*1 127*	*13*
75	155	*0.00293790*	*1 270*	*14*
100	207	*0.00279465*	*1 784*	*21*
125	258	*0.00289490*	*2 120*	*25*

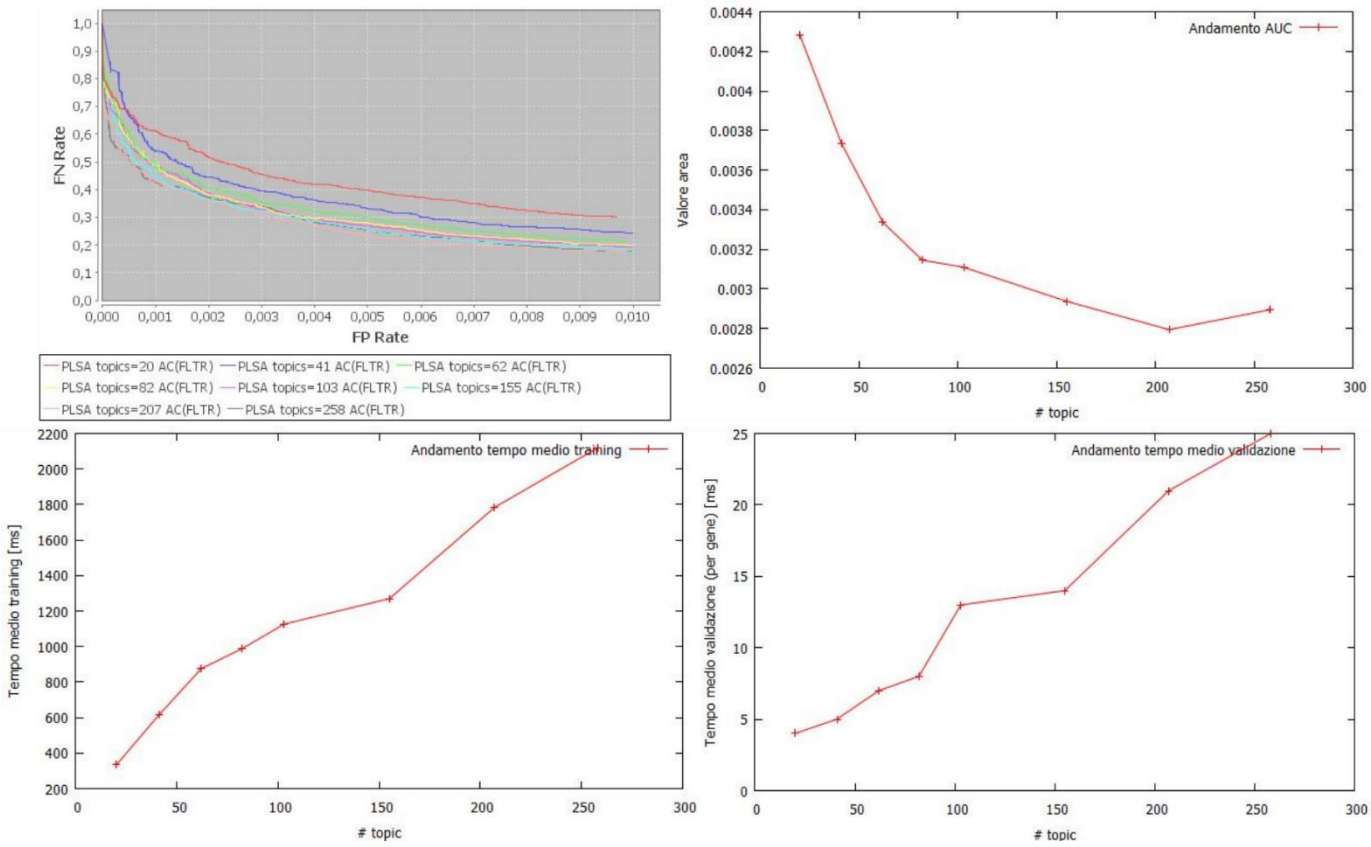


Figura 21: In alto a sinistra le curve ROC ottenute, in alto a destra l'andamento dell'area delle curve ROC in funzione del numero di topics, in basso a sinistra l'andamento del tempo medio necessario all'addestramento del modello in funzione del numero di topics, in basso a destra l'andamento medio del tempo necessario a produrre le predizioni su un gene. I tempi sono riportati in millisecondi.

Tabella 12: Confronti valore AUC ROC al variare del numero di topic con dataset BosTaurus - Molecular Function

Bos Taurus (9913) – Molecular Function (7)				
%	**# *topics***	**AUC**	**Avg train time [ms]**	**Avg test time [ms]**
10	22	*0.00490516*	*290*	*1*
20	45	*0.00388924*	*454*	*1*
30	68	*0.00375202*	*542*	*2*
40	91	*0.00345991*	*750*	*2*
50	114	*0.00339220*	*778*	*3*
75	171	*0.00297161*	*1 062*	*4*
100	229	*0.00297161*	*1 376*	*6*
125	286	*0.00294190*	*1 527*	*11*

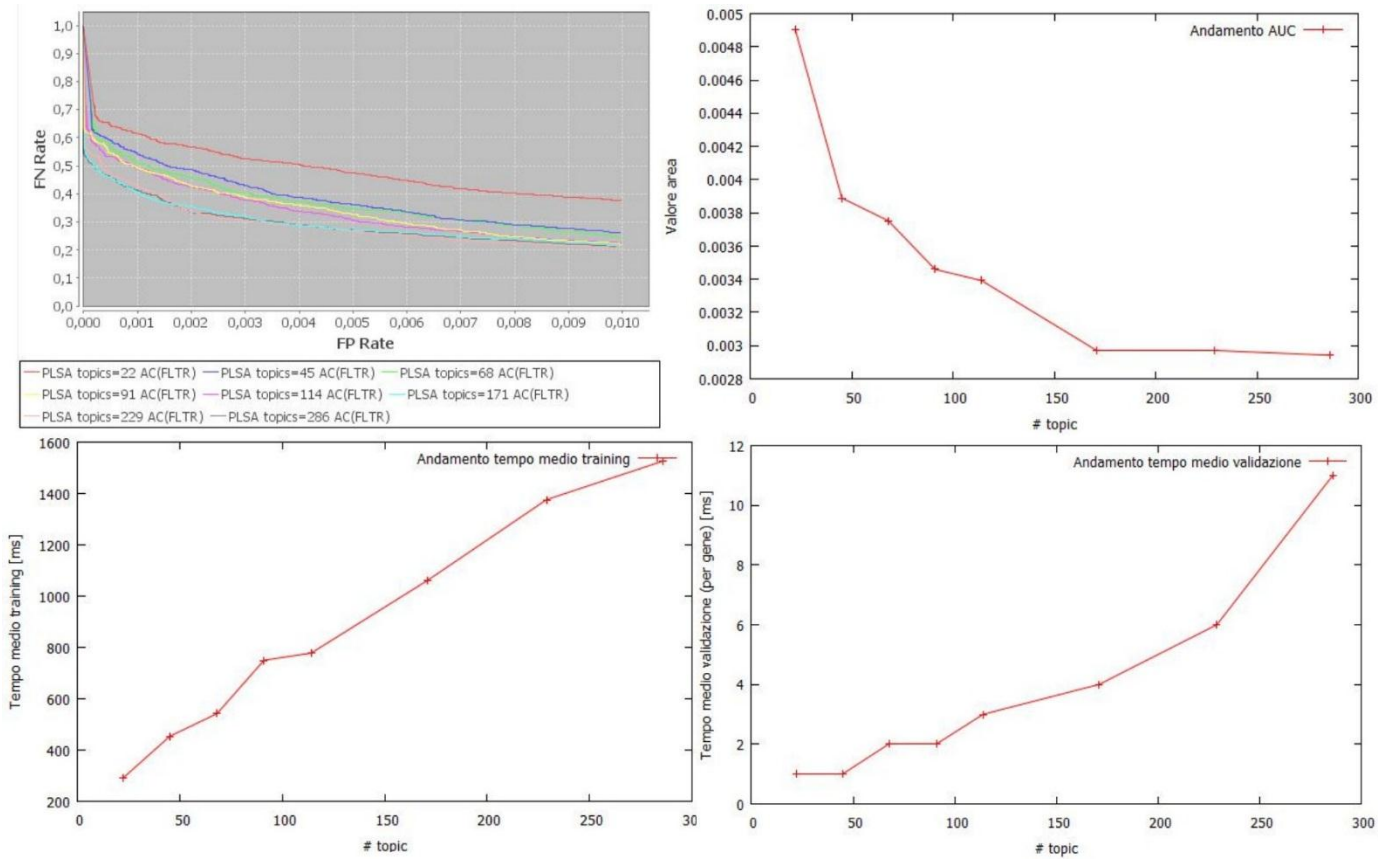


Figura 22: In alto a sinistra le curve ROC ottenute, in alto a destra l'andamento dell'area delle curve ROC in funzione del numero di topics, in basso a sinistra l'andamento del tempo medio necessario all'addestramento del modello in funzione del numero di topics, in basso a destra l'andamento medio del tempo necessario a produrre le predizioni su un gene. I tempi sono riportati in millisecondi.

Tabella 13: Confronti valore AUC ROC al variare del numero di topic con dataset BosTaurus - Biological Process

Bos Taurus (9913) – Biological Process (8)				
%	**# *topics***	**AUC**	**Avg train time [ms]**	**Avg test time [ms]**
10	102	*0.00440266*	*3 586*	*121*
20	204	*0.00344742*	*8 579*	*205*
30	306	*0.00314327*	*13 364*	*240*
40	409	*0.00304376*	*18 302*	*533*
50	511	*0.00291648*	*20 951*	*496*
75	767	*0.00291697*	*28 963*	*1 202*
100	1 023	*0.00301554*	*32 820*	*1 627*
125	1 278	*0.00299310*	*40 516*	*1 889*

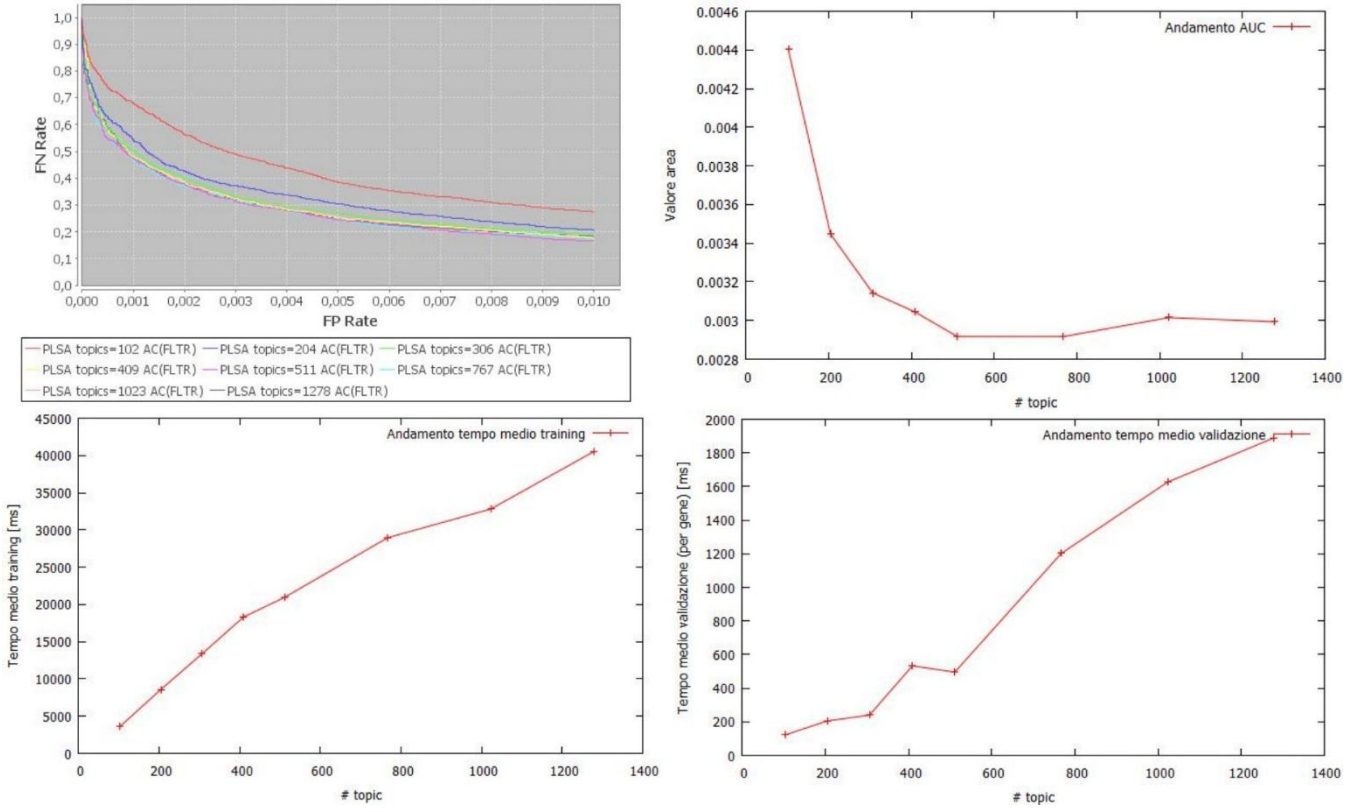


Figura 23: In alto a sinistra le curve ROC ottenute, in alto a destra l'andamento dell'area delle curve ROC in funzione del numero di topics, in basso a sinistra l'andamento del tempo medio necessario all'addestramento del modello in funzione del numero di topics, in basso a destra l'andamento medio del tempo necessario a produrre le predizioni su un gene. I tempi sono riportati in millisecondi.

8.1.2.3 Conclusioni

Dai risultati riportati nelle tabelle precedenti si può vedere come il tempo necessario al calcolo dell'algoritmo EM, sia nell'addestramento del modello che nel suo utilizzo, dipenda linearmente dal numero di *topics* scelto. La curva si discosta dalla retta ideale perché il tempo di esecuzione, così come il valore della soluzione trovata, dipende dalla inizializzazione scelta.

Per quanto riguarda l'andamento delle curve ROC possiamo vedere che inizialmente gli incrementi del numero di *topics* portano una sensibile diminuzione dell'area sottesa alla curva. Il livello dell'area tende a stabilizzarsi quando si raggiunge un numero di *topics* vicino al 75% del numero delle caratteristiche (*feature*). Le variazioni oltre a questa soglia sono ridotte mentre il tempo di esecuzione continua a crescere, quindi usare il 75% del numero delle caratteristiche (feature) sembra una buona euristica per la scelta della dimensione del modello.

8.1.3 Variazioni dovute alla inizializzazione del modello

Come già notato in precedenza la soluzione fornita dal metodo pLSA dipende fortemente dalla inizializzazione scelta. Con questo esperimento intendo valutare sperimentalmente quale sia l'incidenza dell'assegnamento iniziale del valore delle variabili sul risultato finale.

8.1.3.1 Descrizione dell'esperimento

Scelto un dataset, si ricava la matrice *A* delle annotazioni *unfoldata*; la matrice viene divisa in due parti: un *training set* contenente i primi 90% dei geni e un *validation set* contenente i rimanenti (la proporzione è scelta per coerenza con la procedura di validazione usata). Si procede come segue:

i. Si inizializza il training set con valori casuali;
ii. Si applica l'algoritmo EM;
iii. Se il valore finale della verosimiglianza del modello è migliore di tutti i precedenti lo si memorizza come *best_init* , se è peggiore come *worst_init* altrimenti lo si elimina scarta;

iv. Si calcola la distribuzione campionaria del valore della verosimiglianza del modello, basandosi sulle osservazioni fatte nelle iterazioni precedenti;

v. Se c'è evidenza statistica che la probabilità di ottenere una inizializzazione peggiore della *worst_init* sia minore dell'1% e che la probabilità di ottenerne una migliore della *best_init* sia anch'essa minore dell'1% si prosegue col passo (vi) altrimenti si torna al passo (i);

vi. Usando il modello calcolato con l'inizializzazione peggiore si eseguono 10 inizializzazione del *validation set* ; ugualmente si fa partendo dal modello calcolato con l'inizializzazione peggiore;

vii. Si confrontano le medie delle aree ottenute nei due casi.

8.1.3.2 Risultati e commenti

Di seguito riporto i risultati dell'esperimento appena descritto per alcuni dataset; il numero di *topics* scelto è pari al 75% del numero delle caratteristiche (feature), secondo la prassi definita nell'esperimento precedente.

Tabella 14: Variazioni dell'area sottesa alla curva ROC al variare delle inizializzazioni del modello

Dataset	# *topics*	Media migliore	Media peggiore	Differenza
BosTaurus-CC	155	0.003166	0.003288	3.71%
BosTaurus-MF	171	0.003654	0.003925	6.90%
BosTaurus-BP	767	0.004360	0.004808	9.31%
Aspergillus-MF	255	0.000529	0.000587	10.90%
Aspertillus-BP	421	0.000556	0.000598	4.13%

Possiamo vedere come effettivamente l'insieme di assegnamenti iniziali che si fanno alle variabili di pLSA possa influire sensibilmente sui risultati raggiunti. Tuttavia possiamo notare come la verosimiglianza del modello in fase di addestramento sia un buon indicatore delle qualità delle annotazioni predette; è quindi possibile eseguire più addestramenti dello stesso modello partendo ogni volta da un diverso assegnamento casuale alle variabili e scegliere quello migliore, cioè quello con valore di verosimiglianza maggiore, magari implementando la stessa politica di scelta

proposta in questo esperimento. In questo modo è lecito aspettarsi di avere un modello che si comporta meglio.

8.2 Confronto tra pLSA e LSI

In questa sezione eseguo un confronto tra i modelli LSI e pLSA relativamente alla procedure di validazione basate sulle curve ROC o su versioni successive del database.

8.2.1 Confronto basato su curve ROC

Nel seguito vengono riportati dei confronti tra le validazioni sullo stesso database dei modelli pLSA e LSI.

Nella tabella il significato delle colonne è il seguente:

- Dim. (dimension): nel caso di LSI rappresenta il livello di troncamento mentre nel caso di pLSA il numero di *topics*;
- AUC: valore dell'area sottesa alla curva ROC;
- T. train: tempo medio necessario per addestrare il modello in millisecondi;
- T. uso: tempo medio necessario per usare il modello per creare le predizioni su un singolo vettore-gene il millisecondi;
- Mem. : picco di memoria consumato dalla applicazione in MegaBytes;

Tabella 15: Risultati del confronto delle curve ROC dei modelli LSI e pLSA

Dataset	Metodo	Dim.	AUC	T. train [ms]	T. uso [ms]	Mem. [Mb]
GallusGallus-CC	**PLSA**	92	0.003768	450	6	81
	LSI	11	0.004174	1	1	90
GallusGallus-MF	**PLSA**	100	0.002890	371	3	84
	LSI	10	0.004099	1	1	98
GallusGallus-BP	**PLSA**	457	0.003619	6 338	280	135
	LSI	10	0.005592	4	15	108
BosTaurus-CC	**PLSA**	155	0.002905	1 330	15	90
	LSI	16	0.002932	3	2	84
BosTaurus-MF	**PLSA**	171	0.002995	1 068	4	107
	LSI	9	0.004623	5	1	91
BosTaurus-BP	**PLSA**	767	0.002930	1 281	27 718	188
	LSI	24	0.004320	10	36	1 263

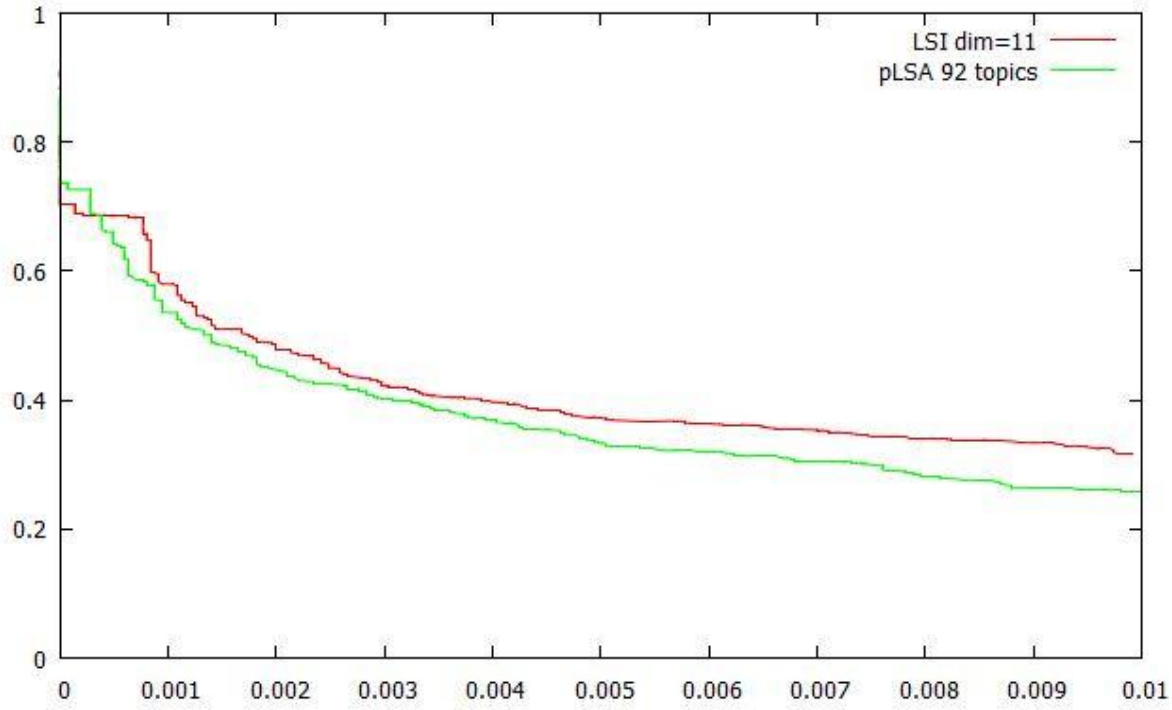


Figura 24: Curve ROC Dataset GallusGallus - Cellular Component

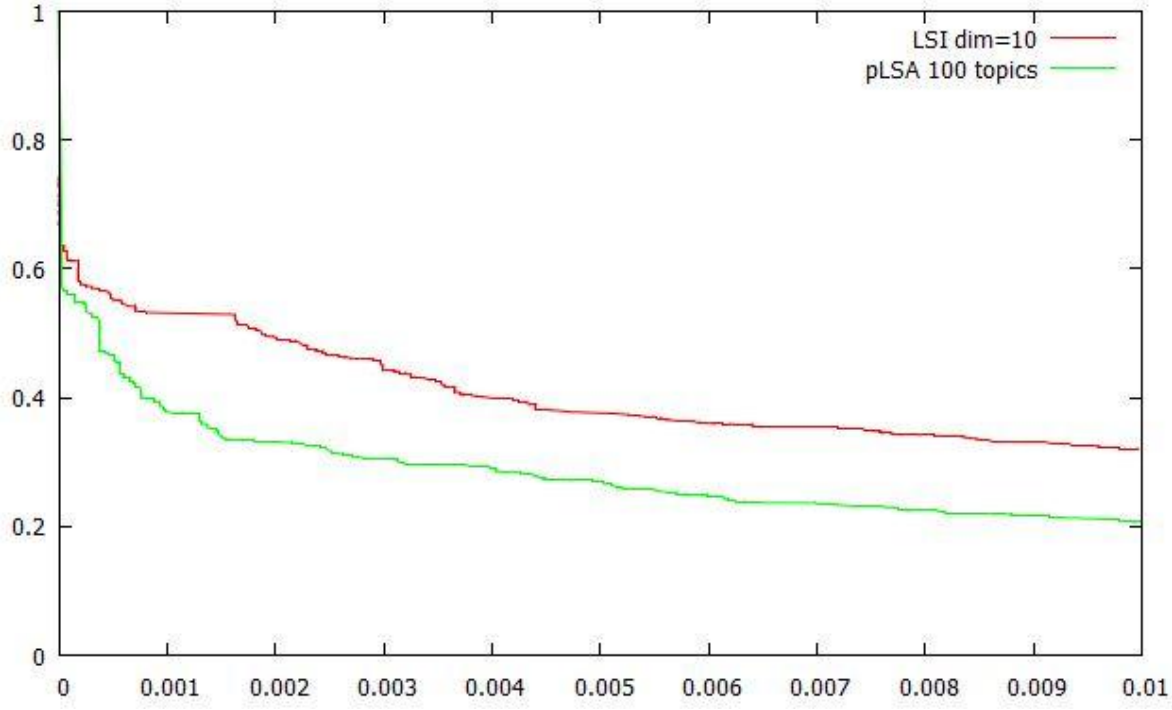


Figura 25: Curve ROC Dataset GallusGallus - Molecular Function

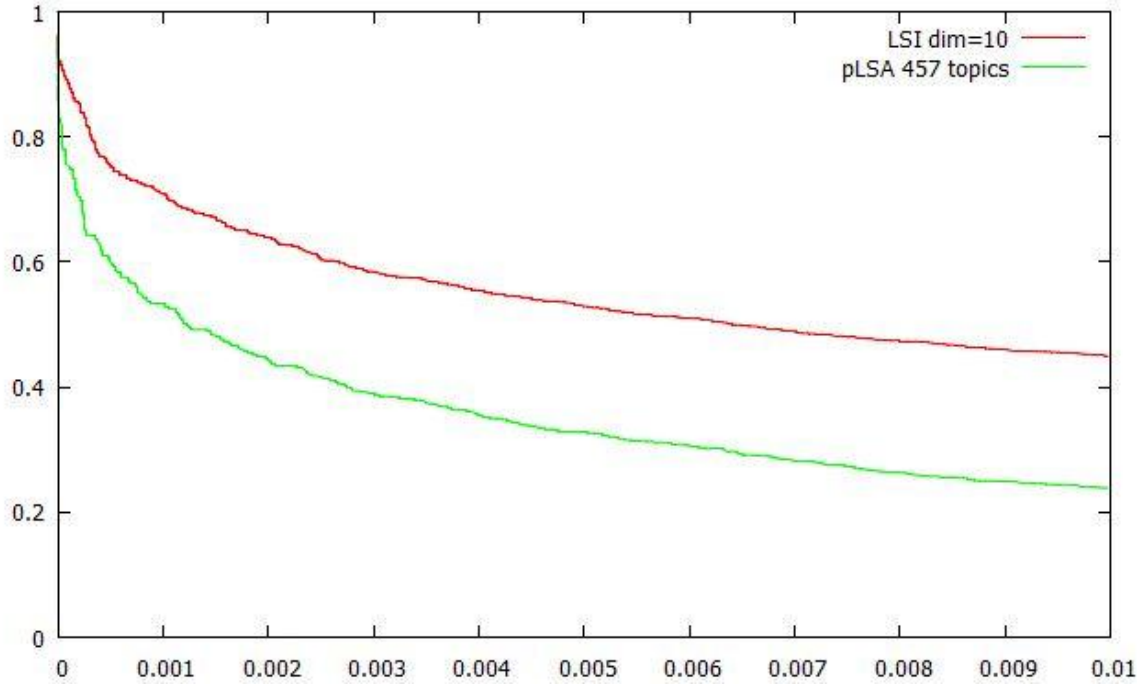


Figura 26: Curve ROC Dataset GallusGallus - Biological Process

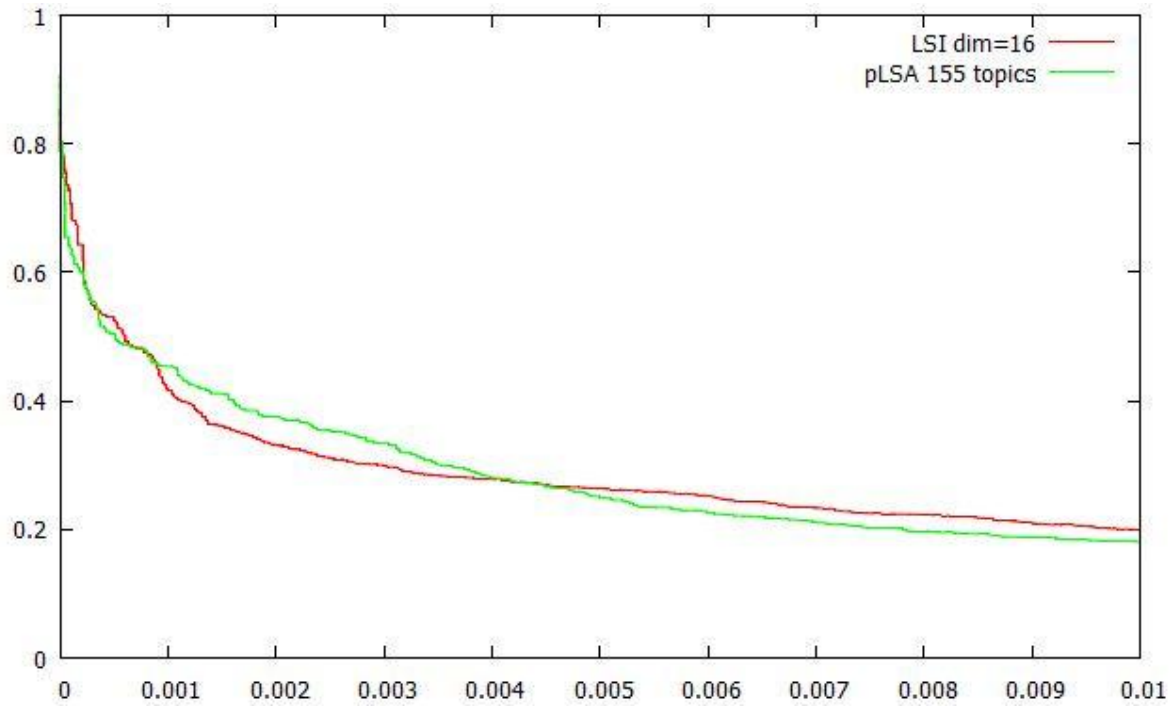


Figura 28: Curve ROC BosTaurus - Cellular Component

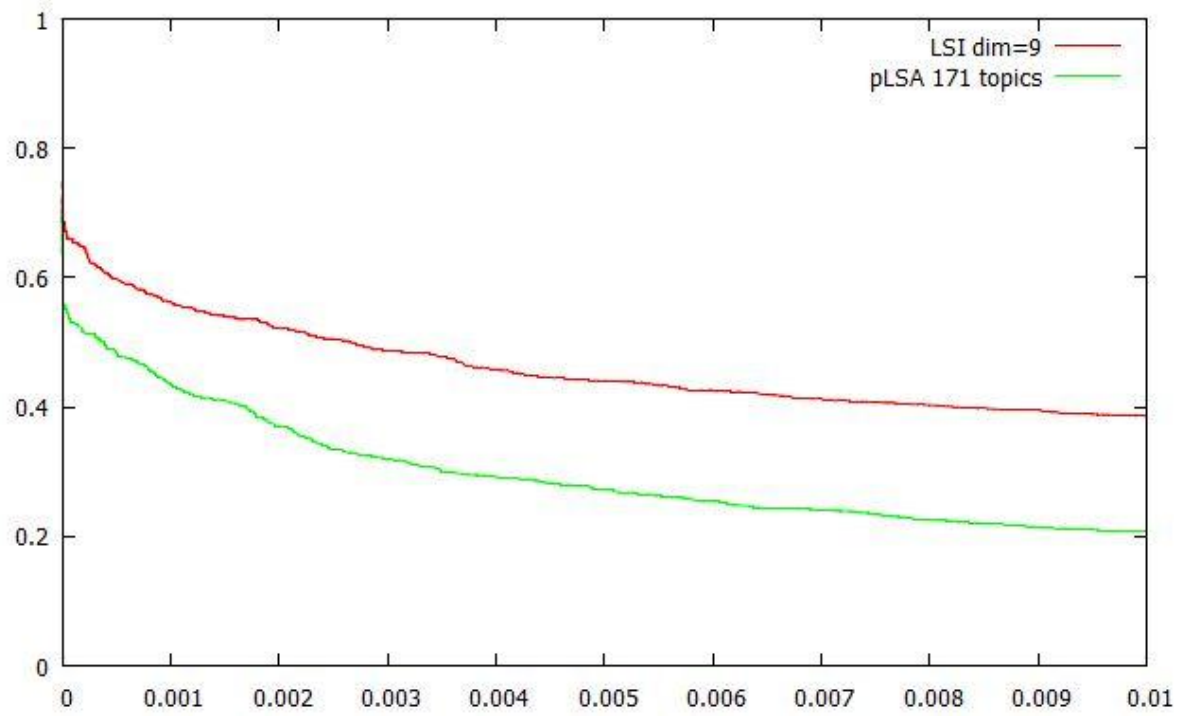


Figura 29: Curve ROC Dataset BosTaurus - Molecular Function

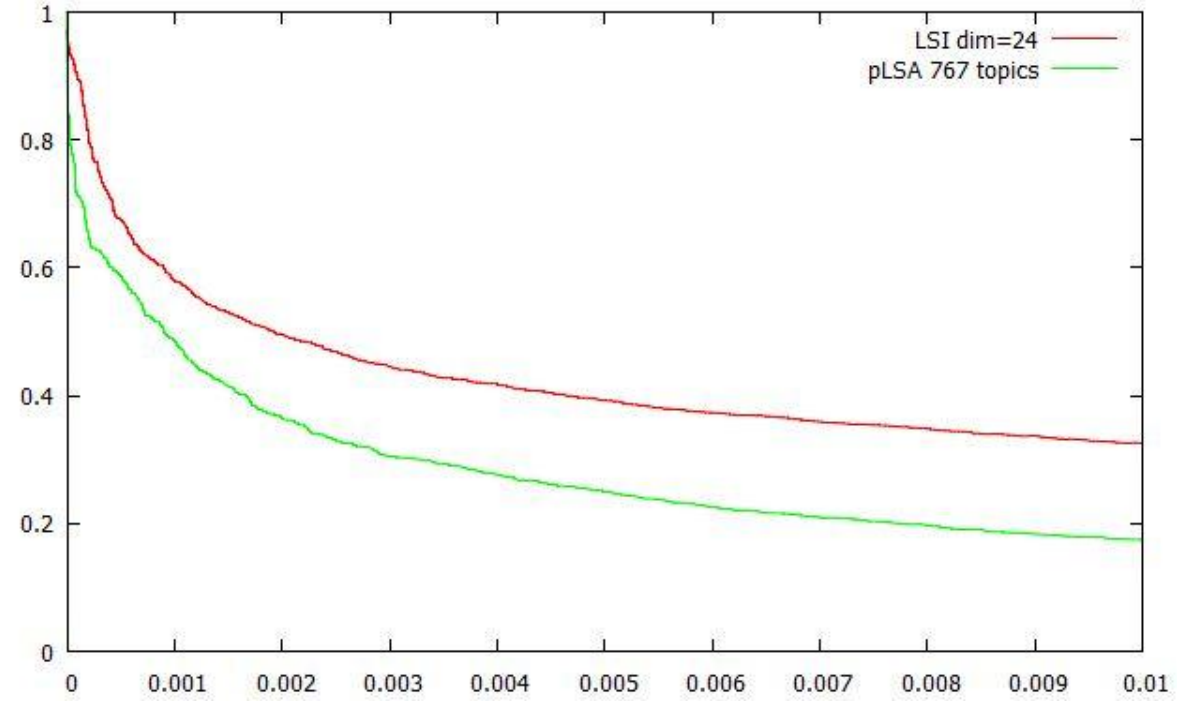


Figura 30: Curve ROC Dataset BosTaurus - Molecular Function

Dall'analisi dei risultati possiamo vedere come l'algoritmo pLSA permetta di raggiungere risultati migliore in termini di area delle curve ROC. Tuttavia i risultati evidenziano anche grandi problemi relativi al tempo di esecuzione che sembrano rendere inutilizzabile l'algoritmo pLSA.

8.2.2 Confronto basato sulla percentuale di predizioni confermate

Questa sezione si basa sulla validazione tra versioni diverse del database; di seguito mostro i risultati delle predizioni ottenute coi due metodi partendo dal database *giorgio2* e confermate sul database *arif5*. In tutti i test di seguito si sono usate 20 variabili nascoste.

Tabella 16: Validazione delle annotazioni predette sulla versione aggiornata del database

Dataset	Metodo	No Confr.	#IEA	%IEA	#NotIEA	%NotIEA
9031_6	PLSA	185	91	11.17%	61	7.48%
	LSI	216	78	9.84%	52	6.55%
9031_7	PLSA	280	50	6.94%	25	3.47%
	LSI	203	72	9.03%	26	3.26%
9031_8	PLSA	278	140	19.39%	87	12.05%
	LSI	260	137	18.51%	111	15.00%
9913_6	PLSA	49	46	4.84%	210	22.08%
	LSI	66	39	4.17%	152	16.27%
9913_7	PLSA	56	24	2.54%	92	9.75%
	LSI	67	23	2.46%	58	6.21%
9913_8	PLSA	92	89	89.98%	157	17.29%
	LSI	92	61	6.71%	136	14.98%

I risultati giustificano la scelta di pLSA rispetto a LSI infatti possiamo vedere come in tutti i dataset, tranne nel terzo, l'algoritmo pLSA superi LSI in quanto a numero di annotazioni predette e confermate.

8.3 Schemi di peso

In questo test, scelto un dataset e un numero di *topics* si applicano gli schemi di peso e si confrontano i risultati che si ottengono al variare dello schema scelto; in particolare si riportano le curve ROC della validazione sullo stesso database e un conteggio delle predizioni confermate nella versione più aggiornata del database.
Il confronto tra versioni successive del database è fatto sulle prime 1000 annotazioni predette ordinate per valore decrescente di predizione. Le colonne delle tabelle in cui sono riportate hanno il seguente significato:

- "Non Confr.": annotazioni per cui non è stato possibile eseguire un confronto con la versione aggiornata del database, questo perché nel nuovo database non sono riportate la sequenza oppure la funzionalità;
- "#IEA" e "%IEA": rispettivamente numero e percentuale delle predizioni confermate nel database aggiornato con grado di evidenza pari a IEA;
- "#notIEA" e "%notIEA": rispettivamente numero e precentuale di annotazioni confermate nel database aggiornato con grado di evidenza migliore rispetto a IEA.

8.3.1 BosTaurus (9913) – Cellular Component (6)

Tabella 17: Confronto curver ROC tra schemi dataset BosTaurus – Cellular Component metodo pLSA

AUC: BosTaurus – CellularComponent – 155 *topics*				
Schema	**AUC**	**Tempo applicazione schema [ms]**	**Tempo training [ms]**	**Tempo predizione [ms]**
None	0.00296323	---	1 366	17
NTN	0.00377768	16	1 285	16
NTM-MTM	0.00297267	21	1 283	18
NTC-MTC	0.00306400	22	1 368	23
MTN	0.00299863	23	1 241	16
ATN	0.00263796	17	1 389	16
ATM	0.00306899	22	1 229	18
ATC	0.00282071	24	1 497	18

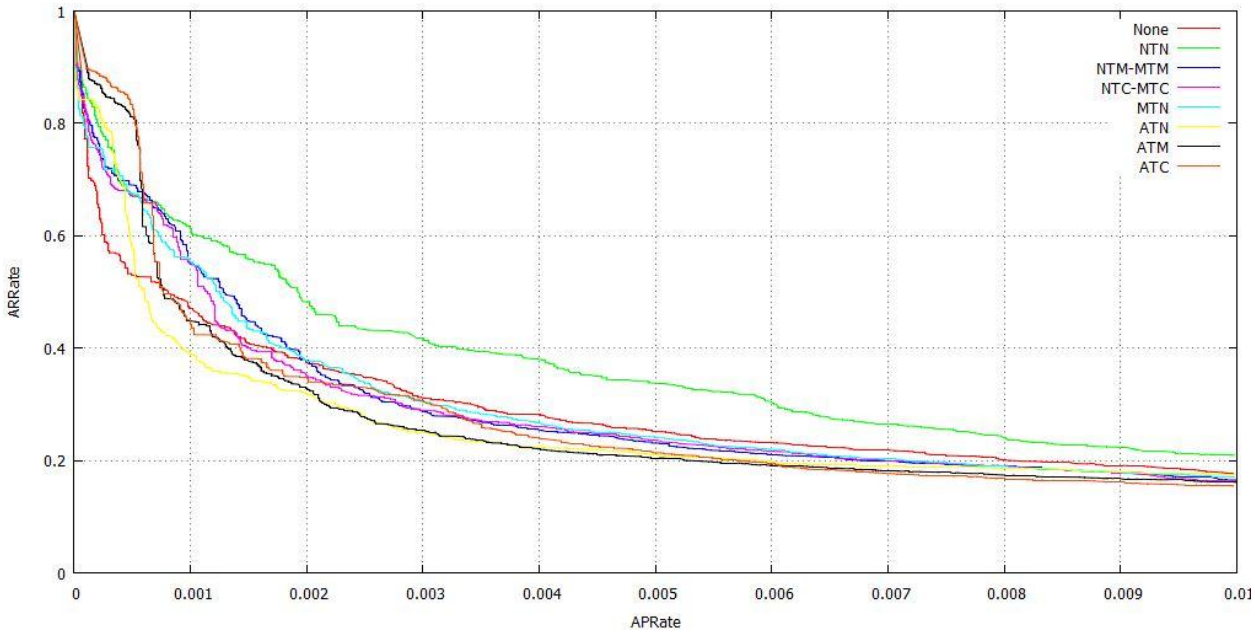


Figura 30: Curve ROC schemi di peso, dataset BosTaurus - Cellular Componen, metodo pLSA

Tabella 18: Risultati predizione con gli schemi di peso per il dataset BosTaurus – CellularComponent, metodo pLSA

Predizioni Confermate: BosTaurus – CellularComponent – 155 *topics*					
Schema	**Non Confr.**	**#IEA**	**%IEA**	**#NotIEA**	**%NotIEA**
None	71	155	16.7%	47	5.1%
NTN	42	177	18.5%	51	5.3%
NTM-MTM	46	192	20.1%	33	3.5%
NTC-MTC	55	126	13.3%	34	3.6%
MTN	56	160	16.9%	34	3.6%
ATN	63	181	19.3%	40	4.3%
ATM	39	206	21.4%	45	4.7%
ATC	50	182	19.2%	38	4.0%

8.3.2 BosTaurus (9913) – Molecular Function

Tabella 19: Confronto curver ROC tra schemi dataset BosTaurus - Molecular Function, metodo pLSA

AUC: BosTaurus – MolecularFunction– 171 *topics*				
Schema	**AUC**	**Tempo applicazione schema [ms]**	**Tempo training [ms]**	**Tempo predizione [ms]**
None	0.00319536	---	1 148	5
NTN	0.00342725	8	987	3
NTM-MTM	0.00315745	12	1 084	4
NTC-MTC	0.00369677	13	1 176	7
MTN	0.0034504	9	1 082	4
ATN	0.00342254	9	1 124	5
ATM	0.00327224	13	965	5
ATC	0.00326355	14	1 081	4

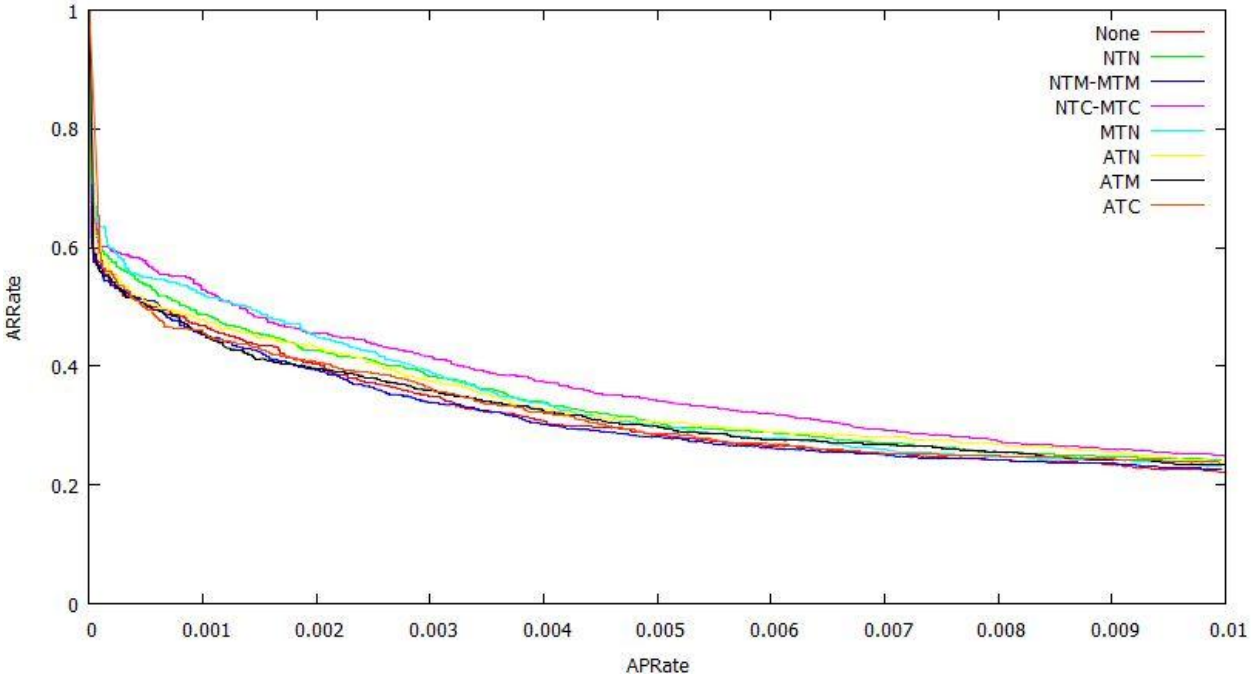


Figura 31: Curve ROC schemi di peso, dataset BosTaurus – Molecular Function, metodo pLSA

Tabella 20: Risultati predizione con gli schemi di peso per il dataset BosTaurus – Molecular Function, metodo pLSA

AUC: BosTaurus – MolecularFunction– 171 *topics*					
Schema	Non Confr.	#IEA	%IEA	#NotIEA	%IEA
None	37	81	8.4%	17	1.8%
NTN	56	135	14.0%	16	1.7%
NTM-MTM	41	99	10.3%	11	1.1%
NTC-MTC	46	49	5.1%	20	2.1%
MTN	58	143	15.2%	29	3.1%
ATN	77	143	15.5%	22	2.4%
ATM	61	68	7.2%	19	2.0%
ATC	40	60	6.3%	15	1.6%

9 Conclusioni

Un primo obiettivo che mi ero preposto all'inizio di questo progetto di tesi era la modifica architetturale del software *AnnotationPredictor* per permette l'implementazione di nuovi modelli predittivi. Le modifiche descritte nel Capitolo 5 hanno sono state fatte a questo scopo; l'implementazione dell'algoritmo pLSA e delle funzioni di pesatura sono due esempi operativi di come sia possibile usare la nuova architettura per estendere le funzionalità del software.

Un secondo obiettivo era lo studio dell'algoritmo pLSA per evidenziarne le potenzialità ed i difetti. I risultati riportati mostrano che l'algoritmo proposto, a fronte di un buon potere espressivo talvolta migliore di LSI, ha rilevanti problemi: innanzitutto i risultati forniti sono fortemente influenzati da una componente non deterministica, ovvero l'inizializzazione casuale dei parametri. In secondo luogo l'algoritmo EM si è rivelato significativamente più lento della decomposizione a valori singolari. Considerati questi problemi, l'applicazione dell'algoritmo pLSA in un caso pratico, dove, data la numerosità dei dati, il tempo di risposta è cruciale, sembra al momento non attuabile; occorre ancora studiare delle modifiche o delle euristiche per velocizzare il procedimento.

La reingegnerizzazione effettuata ha permesso anche di implementare delle funzioni di pesatura indipendenti dal modello predittivo utilizzato. Dai risultati si può vedere come queste tecniche possano efficacemente modificare il risultato del processo di predizione, migliorando o peggiorando la valutazione fornita dalle nostre metriche. Come abbiamo discusso in precedenza tuttavia i processi di validazione implementati possono fornire soltanto una stima della reale bontà del metodo statistico. Ulteriori studi, specialmente la validazione manuale delle annotazioni, potranno fornire migliori indicazioni sulla effettiva utilità degli schemi.

10 Sviluppi Futuri

Il progetto di tesi svolto può essere ancora esteso; possiamo dividere i possibili sviluppi futuri in due categorie: miglioramenti dell'algoritmo pLSA e miglioramenti nel processo di predizione di annotazioni.

10.1 Sviluppi pLSA

L'algoritmo pLSA ha ancora delle questioni aperte, illustrate nella sezione 6.1.5 di questa tesi. Possibili sviluppi futuri potrebbe cercare da dare una soluzioni a qui problemi: in particolare si potrebbero progettare e implementare delle tecniche per la determinazione di una buona inizializzazione o la scelta automatica di un numero un numero appropriato di *topics.*
Altre estensioni possono concernere lo sviluppo di algoritmi alternativi a EM per l'addestramento del modello probabilistico di pLSA, oppure l'integrazione di tecniche per velocizzare l'algoritmo come le tecniche per accelerare la convergenza di una serie (ad esempio l'accelerazione di Aitken).
Una modifica interessante che propongo di analizzare è quella di invertire i ruoli di entità biomolecolari e funzionalità all'interno dell'algoritmo; infatti noi abbiamo usato sempre "il punto di vista dei geni", cioè dato un vettore-gene incompleto si cerca di predire a quali altre funzionalità esso è annotato. Un approccio complementare è quello di considerare i vettori-funzionalità, cioè a partire da un vettore colonna della matrice delle annotazioni si potrebbe cercare di predire quali altri geni sono annotati ad esso; si tratta in pratica di usare un approccio duale a quello usato finora.

10.2 Sviluppi nella predizioni di annotazioni

In questa tesi si sono evidenziati i problemi nel processo automatizzato di predizione di annotazioni. L'algoritmo pLSA e le funzioni di pesatura proposte permettono di risolvere alcuni di essi. Altre soluzioni potrebbero essere proposte, avvalendosi di strumenti statistici alternativi a pLSA. Uno di questi è l'algoritmo LDA (*Latent Dirichlet Allocation*) [11], basato su un modello generativo simile a quello di pLSA ma più raffinato. In particolare LDA tenta di risolvere uno dei

problemi principali di pLSA, ovvero la determinazione delle probabilità $P(documento \mid topic)$; infatti, come fanno notare gli autori, l'uso dell'algoritmo EM per la determinazione di queste probabilità significa adattare il modello ai nuovi dati. Uno sviluppo futuro potrebbe essere sicuramente l'integrazione di questa tecnica all'interno del software *AnnotationPredictor*.

Inoltre gli schemi di peso proposti sono solo un sottoinsieme di quelli disponibili in letteratura. Un altro sviluppo futuro potrebbe essere la ricerca e l'implementazione di altri di questi schemi.

Infine si potrebbe sviluppare una procedura per la validazione automatica delle annotazioni predette in letteratura oppure integrare strumenti già disponibili [19].

Il software descritto in questa tesi potrebbero anche essere implementato come servizio web basato sul protocollo REST (*Representational Transfer State*) per poter essere integrati all'interno della piattaforma BioSeCo [17] di Search Computing [18].

12 Bibliografia

[1] Genomic Proteomic Data Warehouse, http://www.bioinformatics.dei.polimi.it/GPKB/ .

[2] O. King, R. Foulger, S. Dwight, J. White, F. Roth, "Predicting gene function from patterns of annotation", *Genome res.*, vol. 13, no. 5, pp. 896-904, 2003.

[3] Y. Tao, L. Sam, J. Li, C. Friedman, and Y. A. Lussier, "Information theory applied to the sparse gene ontology annotation network to predict novel gene function", *Bioinformatics*, vol. 23, no. 13, pp. 529–538, 2007.

[4] P. Khatri, B. Done, A. Rao, A. Done, and S. Draghici, "A semantic analysis of the annotations of the human genome", *Bioinformatics*, vol. 21, no. 16, pp. 3416–3421, 2005.

[5] D. Chicco, M. Tagliasacchi, M. Masseroli, "Biomolecular annotation prediction through information integration", *Proceedings of CIBB2011 -8th International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*, Gargnagno sul Garda, Italy, pp. 1-9, 2011.

[6] R. Sarati, "Metodi e architetture software per la predizione di annotazioni genomiche funzionali e la stima di similarità semantica di geni e proteine", 2009.

[7] T. Hofmann, "Probabilistic Latent Semantic Indexing", *Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99)*, pp. 50–57, 1999.

[8] M. Masseroli, D. Chicco, P. Pinoli, "Probabilistic Latent Semantic Analysis for predicting Gene Ontology annotations", *IEEE WCCI 2012 - the 2012 IEEE World Congress on Computational Intelligence proceedings*, pp. 1-8, 2012.

[9] C. Ding, L. Tao, and Wei Peng, "On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing", *Computational Statistics & Data Analysis*, 52.8, pp. 3913-3927, 2008.

[10] B. Done, P. Khatri, A. Done, S. Draghici, "Semantic analysis of genome annotations using weighting schemes", Proceedings of 2007 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, pp. 212-218, 2007.

[11] D. Blei, A. Ng, M. Jordan, "Latent Dirichlet Allocation", *Journal of Machine Learning Research*, pp. 993-1022, 2003.

[12] Cullum and Willoughby, "Lanczos Algorithms for Large Symmetric Eigenvalue Computations", Vol. 1, 2002.

[13] A. Vinokourov, M. Girolami, "A Probabilistic Framework for the Hierarchic Organisation and Classification of Document Colletions" , *Information Processing and Management*, 2002.

[14] R. Nallapati, W. Cohen, "Link-PLSA-LDA: a new unsupervised model for topics and influence of blogs", 2009.

[15] A. Bosch. A. Zisserman, X. Munoz, "Scene classification via pLSA", *ECCV Proceedings*, 2006.

[16] A. Dempster, N. Laird, D. Rubin, "Maximum likelihood form incomplete data via EM algorithm", *Journal of Royal Statistical Society*, 1977.

[17] M. Masseroli, G. Ghisalberti, S. Ceri, "Bio-Search Computing: Integration and global ranking of bioinformatics search results", *Journal of Integrative Bioinformatics*, pp. 1-9, 2011.

[18] Stefano Ceri, Marco Brambilla , "Search Computing challenges and directions", 2010.

[19] A. Nuzzo, F. Mulas , M. Gabetta , E. Arbustini , B. Zupan, C. Larizza, R. Bellazzi,, "Text Mining approaches for automated literature knoledge extraction and representation.", *Studies Health Technol Informatics*, 2010;

Appendice - Datasets

Aggiornamento Database:

- ***giorgio2*** Luglio 2009
- ***arif5*** Ottobre 2011

GallusGallus – CellularComponent – giorgio2

- Tassonomia: GallusGallus (9031)
- Ontologia: Cellular Component (6)
- Numero entità biomolecolari (geni / proteine): 260
- Numero caratteristiche (feature): 123
- Geni/proteine esclusi: obsoleti
- Caratteristiche escluse: obsolete
- Annotazioni escluse: IEA e ND
- Annotazioni dirette positive: 477
- Annotazioni dirette negative: 1
- Annotazioni matrice *unfoldata:* 3 450
- Grado di sparsità della matrice: 0.10788

GallusGallus – MolecularFunction – giorgio2

- Tassonomia: GallusGallus (9031)
- Ontologia: Molecular Function (7)
- Numero entità biomolecolari (geni / proteine): 309
- Numero caratteristiche (feature): 134
- Geni/proteine esclusi: obsoleti
- Caratteristiche escluse: obsolete
- Annotazioni escluse: IEA e ND
- Annotazioni dirette positive: 505

- Annotazioni dirette negative: 4
- Annotazioni matrice *unfoldata:* 1 944
- Grado di sparsità della matrice: 0.0469497

GallusGallus – Biological Process – giorgio2

- Tassonomia: GallusGallus (9031)
- Ontologia: Biologica Process (8)
- Numero entità biomolecolari (geni / proteine): 275
- Numero caratteristiche (feature): 610
- Geni/proteine esclusi: obsoleti
- Caratteristiche escluse: obsolete
- Annotazioni escluse: IEA e ND
- Annotazioni dirette positive: 735
- Annotazioni dirette negative: 3
- Annotazioni matrice *unfoldata:* 8 371
- Grado di sparsità della matrice: 0.0520477

BosTaurus – Cellular Component – giorgio2

- Tassonomia: BosTaurus (9913)
- Ontologia: Cellular Component (6)
- Numero entità biomolecolari (geni / proteine): 497
- Numero caratteristiche (feature): 207
- Geni/proteine esclusi: obsoleti
- Caratteristiche escluse: obsolete
- Annotazioni escluse: IEA e ND
- Annotazioni dirette positive: 921
- Annotazioni dirette negative: 0
- Annotazioni matrice *unfoldata:* 7 658

- Grado di sparsità della matrice: 0.074437

BosTaurus – Molecular Function – giorgio2

- Tassonomia: BosTaurus (9913)
- Ontologia: Molecular Function (7)
- Numero entità biomolecolari (geni / proteine): 543
- Numero caratteristiche (feature): 229
- Geni/proteine esclusi: obsoleti
- Caratteristiche escluse: obsolete
- Annotazioni escluse: IEA e ND
- Annotazioni dirette positive: 931
- Annotazioni dirette negative: 4
- Annotazioni matrice *unfoldata:* 3 574
- Grado di sparsità della matrice: 0.0287421

BosTaurus – Biological Process – giorgio2

- Tassonomia: BosTaurus (9913)
- Ontologia: Biological Process (8)
- Numero entità biomolecolari (geni / proteine): 512
- Numero caratteristiche (feature): 1 023
- Geni/proteine esclusi: obsoleti
- Caratteristiche escluse: obsolete
- Annotazioni escluse: IEA e ND
- Annotazioni dirette positive: 1 555
- Annotazioni dirette negative: 2
- Annotazioni matrice *unfoldata:* 18 167
- Grado di sparsità della matrice: 0.0346847