UNIVERSITÀ
DEGLI STUDI
DI PALERMO

Gene-Disease association analyzing the scientific literature

Salvatore Calderaro

Indice

1 Introduzione

In questo progetto è stato implementato un sistema che dato in input un gene (il suo ID), controlla se l'ID inserito appartiene a un gene realmente esistente, se il controllo va a buon fine vengono memorizzate all'interno di un dataframe una serie di informazioni inerenti il gene, in particolare: la tassonomia, l'ID, il simbolo e il nome ufficiale completo. Fatto ciò si procede con l'estrazione dalla piattaforma *PubMed* - mediante *web scraping* - dei duecento articoli più rilevanti (se disponibili) in cui il gene è stato studiato. Di quest'ultimi vengono estratti e caricati all'interno di un dataframe titolo ed abstract. Per ogni articolo presente nel dataframe viene effettuato un pre-processing: eliminazione delle stop words, della punteggiatura e altre tecniche di natural language processing che verranno descritte più approfonditamente nella prossima sezione. Fatto ciò si procede con la *Named Entity Recognition* (NER), per stabilire all'interno di un testo quali parole o insiemi parole possono essere etichettate come malattie. La lista di malattie cosi ottenuta viene filtrata in modo tale da eliminare eventuali duplicati o parole che si hanno a che fare con l'ambito biomedico, ma che in realtà non sono nomi riconducibili a malattie. Infine per valutare la bontà dei risultati ottenuti, la lista di malattie viene confrontata, mediante *fuzzy string matching* con la lista di malattie che sono associate al gene che si sta studiando estratta dal database *DisGenNet*. In output verrà restituita: la percentuale di malattie esatte trovate, la lista della malattie e una wordcloud per rappresentare graficamente i risultati. Il codice sorgente del software è reperibile alla seguente repository GitHub: https://github.com/Calder10/Gene-Disease-Association.

2 Descrizione del software

Il linguaggio di programmazione utilizzato per l'implementazione del software è *Python*. Come IDE per effettuare lo sviluppo è stato scelto *Atom*. Le librerie utilizzate per l'implementazione del software sono le seguenti:

- *pyspark*:
- *nltk*;
- *biopython*;
- *scispacy*;
- *spacy*;
- *textblob*;
- *fuzzywuzzy*;
- *wordcloud*;
- *matplotlib*.

I dati sono stati reperiti mediante *Entrez*, un sistema di ricerca integrato tra banche dati biomediche contenenti informazioni di tipo differente coordinato dal *NCBI*. Per la valutazione dei risultati ottenuti è stato utilizzato il database *DisGenNet*.

Di seguito verranno descritte in dettaglio le varie fasi della progettazione del software.

2.1 Estrazione delle informazioni inerenti il gene

Il sistema, una volta che l'utente ha inserito in input l'ID del gene di cui vuole ricavare le malattie associate, verifica se a quell'ID è associato effettivamente un gene mediante la funzione *check_gene(gene_id)*. Tale funzione esegue una query tramite *Entrez* sul database associato *Gene*, che raccoglie informazioni di sequenza centrate sui singoli geni. Qualora la query avesse esito positivo viene restituito un file XML, dalla quale vengono estratte ed inserite all'interno di un dataframe le seguenti informazioni:

- *TaxonomyName*;
- *ID*;
- *OfficialSymbol*;
- *OfficiallFullName*.

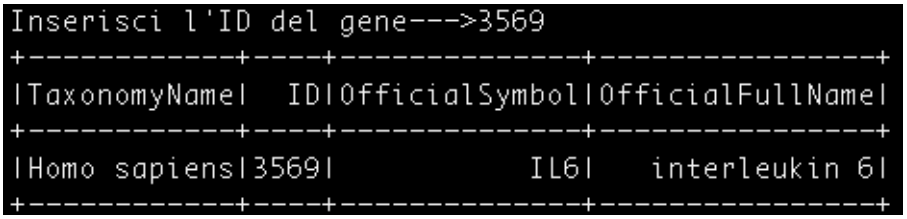


Figura 1: Dataframe contenente le informazioni sul gene

Qualora la query dovesse avere esito negativo viene visualizzato un messaggio di errore e viene richiesto l'inserimento di nuovo ID.

2.2 Estrazione degli articoli scientifici

Una volta identificato il gene di cui si vogliono ricavare le malattie associate, si procede - mediante *web scraping* - all'estrazione della letteratura scientifica inerente il gene in questione tramite la funzione *find_papers(gene_id)*. Quest'ultima prende in input l'ID del gene e sempre mediante l'utilizzo di *Entrez* prima effettua una query per verificare l'esistenza di riferimenti (link) agli articoli più rilevanti correlati all'ID cercato (l'ID del gene). Se non si dovessero trovare riferimenti viene mostrato all'utente un messaggio di errore. Successivamente per ogni link trovato si effettua una query sul database *PubMed* il quale raccoglie i riferimenti agli articoli scientifici apparsi su un numero elevato di riviste scientifiche, principalmente di tipo biomedico. Il risultato di tale query è sempre organizzato in formato XML, e contiene tutta una serie di informazioni inerenti l'articolo: l'anno di pubblicazione, la rivista, il titolo, l'abstract etc.

Degli articoli trovati ne vengono considerati solamente duecento - se disponibili. Di quest'ultimi le informazioni che vengono estratte sono il titolo e l'abstract se è disponibile. Infine questi duecento articoli con relativi titoli ed abstract vengono memorizzati all'interno di un dataframe.

Title	Abstract
Interleukin-6 in ...	The role of inter...
Role of Interleuk...	COVID-19 is viral...
EBV Rta-induced I...	Rta, a transactiv...
Elevated levels o...	Coronavirus disea...
Prognostic value ...	The inflammatory ...
IL-6 produced by ...	Trichomonas vagin...
Association betwe...	We aimed to compa...
Association of IL...	The -174G>C (rs18...
Interleukin-6 gen...	Several studies h...
IL-6 is present i...	IL-6 is a pro-inf...
IL-6 mediated JAK...	We investigated t...
Does serum interl...	The diagnosis of ...
Association of -1...	Interleukin-6 (IL...
Baseline Interleu...	The objective of ...
Interleukin-6 med...	The phosphoinosit...
Association betwe...	The association b...
Association of <i...	Autoimmune thyroi...
The Role of Inter...	Studies have show...
Association betwe...	
Association of Va...	Lung cancer is kn...

only showing top 20 rows

Figura 2: Dataframe articoli scientifici

2.3 Pulizia dei dati

Dopo aver memorizzato all'interno di un dataframe gli articoli scientifici si procede con la fase di pulizia dei dati tramite la funzione *clean_data(paper_df)*. Il primo step della pulizia dei dati consiste nell'eliminazione - sia nel titolo che nell'abstract - dei simboli di punteggiatura (punti, virgole, punti esclamativi) e di eventuali spazi. Fatto ciò si effettua la *tokenizzazione* del testo, ovvero quel processo che ci permette di suddividere un testo nei suoi componenti principali i cosiddetti *token*. Il tipo di tokenizzazione più semplice è quello di suddividere un dato testo in base agli spazi, ma così facendo potrebbero essere token vicini che rappresentano una singola entità che verrebbero visti come due token diversi. Quello che infine si fa è utilizzare un tokenizzatore che per effettuare lo split segue un set prefissato di regole. In questo software per effettuare la tokenizzazione del testo si è utilizzato *RegexpTokenizer* di *nltk*. Quest'ultimo effettua la tokenizzazione del testo in base all'espressione regolare che viene fornita in input che sarà il delimitatore tra una parola ed un'altra.

```
TESTO:
A unique subset of low-risk Wilms tumors is characterized by loss of function of TRIM28 (KAP1), a gene crit
ical in early renal development: A Children's Oncology Group study


TESTO DOPO LA RIMOZIONE DELLA PUNTEGGIATURA E TOKENIZZATO:
['A', 'unique', 'subset', 'of', 'low', 'risk', 'Wilms', 'tumors', 'is', 'characterized', 'by', 'loss', 'of'
, 'function', 'of', 'TRIM28', 'KAP1', 'a', 'gene', 'critical', 'in', 'early', 'renal', 'development', 'A',
'Children', 's', 'Oncology', 'Group', 'study']
```

Figura 3: Esempio di rimozione della punteggiatura e tokenizzazione

Dopo aver effettuato la tokenizzazione del testo si procede con la rimozione delle *stopwords* ovvero quelle parole comunemente usate che non portano nessuna informazione utile al testo. Esempi tipici sono le congiunzioni, gli avverbi, le preposizioni, i pronomi e i verbi di uso comune come ad esempio i verbi essere ed avere. Eliminando le stopwords dal nostro testo riusciamo ad diminuire in un qualche modo la quantità di dati che successivamente dovrà essere processata. Per eliminare le stopwords si è fatto sempre uso della libreria *nltk* la quale mette a disposizione tutte le *stopwords* più comuni della lingua inglese.

```
TESTO DOPO LA RIMOZIONE DELLE STOPWORDS:
['unique', 'subset', 'low', 'risk', 'Wilms', 'tumors', 'characterized', 'loss', 'function', '
TRIM28', 'KAP1', 'gene', 'critical', 'early', 'renal', 'development', 'Children', 'Oncology',
 'Group', 'study']
```

Figura 4: Esempio di rimozione delle stopwords

L'ultimo step della fase di pulitura dei dati consiste nell'effettuare la *lemmatizzazione* dei token. Questo processo permette di ridurre le parole dalla loro forma flessa alla loro forma canonica, che viene detta giustappunto *lemma*. La lemmatizzazione non segue un insieme prefissato di regole, il lemma di una parola potrebbe variare da frase a frase in base al significato della parola. Lo scopo della lemmatizzazione è quello di ridurre il numero di parole diverse all'interno del nostro corpus di testo. Si noti che si sarebbe potuto scegliere anche la tecnica dello *stemming*, ovvero ridurre una parola alla sua forma base: lo *stem*. *Stemming* e *lemmatizzazione* hanno lo stesso scopo, ma essendo la *lemmatizzazione* una tecnica più sofisticata e che porta a risultati migliori nell'implementare questo software si è scelta la seconda. Per effettuare la *lemmatizzazione* si è utilizzato la classe *WordNetLemmatizer* di *nltk*. Tale classe non esegue una corretta lemmatizzazione dei verbi in quanto necessita di sapere a che parte del discorso appartiene una determinata parola. Si è scelto di utilizzarla ugualmente in quanto i sostantivi

che rappresentano verbi non sono utili agli scopi finali e verranno eliminati in seguito quando verrà applicato il *Part of speech tagging*.

```
TOKEN DOPO LA LEMMATIZZAZIONE:
['unique', 'subset', 'low', 'risk', 'wilms', 'tumor', 'characterized', 'loss', '
function', 'trim28', 'kap1', 'gene', 'critical', 'early', 'renal', 'development'
, 'child', 'oncology', 'group', 'study']
```

Figura 5: Esempio di lemmatizzazione

Riassumendo la funzione *clean_data(paper_df)* - per ogni articolo presente nel dataframe - esegue i seguenti passi:

1. rimozione dei segni di punteggiatura e di eventuali spazi;
2. tokenizzazione del testo;
3. rimozione delle stopwords;
4. lemmatizzazione.

Infine gli articoli cosi processati vengono memorizzati all'interno di un nuovo dataframe il quale conterrà sia per il titolo che per l'abstract una lista di token.

```
+--------------------+--------------------+
|               Title|            Abstract|
+--------------------+--------------------+
|[interleukin, 6, ...|[role, interleuki...|
|[role, interleuki...|[covid, 19, viral...|
|[ebv, rta, induce...|[rta, transactiva...|
|[elevated, level,...|[coronavirus, dis...|
|[prognostic, valu...|[inflammatory, re...|
|[il, 6, produced,...|[trichomonas, vag...|
|[association, ser...|[aimed, compare, ...|
|[association, il,...|[174g, c, rs18007...|
|[interleukin, 6, ...|[several, study, ...|
|[il, 6, present, ...|[il, 6, pro, infl...|
|[il, 6, mediated,...|[investigated, ch...|
|[serum, interleuk...|[diagnosis, persi...|
|[association, 174...|[interleukin, 6, ...|
|[baseline, interl...|[objective, study...|
|[interleukin, 6, ...|[phosphoinositol,...|
|[association, int...|[association, pla...|
|[association, il6...|[autoimmune, thyr...|
|[role, interleuki...|[study, shown, si...|
|[association, il,...|                  []|
|[association, var...|[lung, cancer, kn...|
+--------------------+--------------------+
only showing top 20 rows
```

Figura 6: Dataframe dopo la pulitura dei dati

2.4 Part of speech tagging

Dopo aver effettuato la pulitura dei dati, per rimuovere le parti di testo non essenziali per lo scopo finale del software, viene utilizzato il *Part of speech tagging* mediante la funzione *posTagging(clean_paper_df)*. Il *part of speech* tagging è una tecnica che permette di identificare la parte del discorso (*part of speech*) di una determinata parola all'interno di un testo. Esempi di tag che possono essere identificati sono:

- aggettivi;
- preposizioni;
- avverbi;
- congiunzioni;
- articoli;
- nomi;
- numeri;
- etc.

Esistono diverse tipologie di *part of speech tagging* che utilizzano metodi differenti. Le tipologie di metodi più utilizzati sono:

- *metodi lessicali*: utilizzano un corpus di testo già etichettato ovvero un corpus di testo in cui ogni parola è stata già associata ad un tag. Utilizzando questo corpus di testo già etichettato si effettua il *part of speech tagging* su di un nuovo documento assegnando per ogni parola del documento la *part of speech* più frequente all'interno del corpus di testo per la parola presa in considerazione;
- *metodi basati su regole*: si stabiliscono una serie di regole che ci permettono di stabilire quale è la *part of speech* di una determinata parola. Di solito tali metodi vanno utilizzati insieme ai metodi lessicali nel senso che se viene incontrata una parola che non è presente all'interno del corpus già etichettato verrà utilizzata una regola per etichettarla;
- *metodi probabilistici*: esempi sono *Conditional random fields* e *Hidden Markov Models*.

I metodi metodi probabilistici erano quelli più frequentemente utilizzati fino a qualche anno fa. Negli ultimi anni tali tecniche sono state implementate mediante l'utilizzo di *machine learning* e *deep learning* in particolare vengono utilizzate le *Recurrent neural networks*.

Nell'implmentare questo software - per effettuare il *part of speech tagging* - è stata utilizzato1 il modello di machine learning *Averaged Perceptron Tagger* unitamente alla funzione *pos_tag* di *nltk*. Per la lingua inglese il tagger che viene utilizzato sfrutta il tagset *Penn Treebank*.

La funzione *pos_tag* prende in input il testo già tokenizzato e restituisce una lista di tuple del tipo *(token,tag)*:

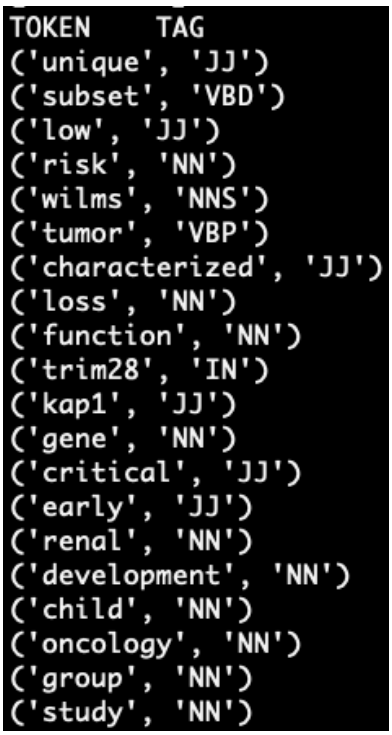


Figura 7: Esempio di Part of speech tagging

Dopo aver effettuato tale operazione, si procede con un'ulteriore filtraggio dei dati. Più dettagliatamente vengono eliminati dalla lista quelle parole che sono state etichettate come *part of speech*, che per i fini del programma non risultano necessarie. In questo modo è possibile ridurre ancora di più la quantità di dati che devono essere manipolati.

In particolare le parole che non verranno scartate sono quelle che avranno i seguenti tag:

- *NNS*: sostantivo comune plurale;
- *NN*: sostantivo comune singolare;
- *FW*: parola straniera:
- *SYM*: simbolo;
- *CD*: numerico, cardinale.

Per avere una descrizione completa di ogni tag può essere utilizzato il seguente comando: *nltk.help.upenn_tagset()*.

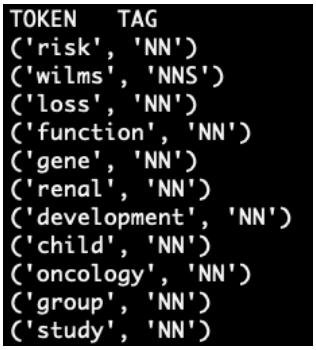
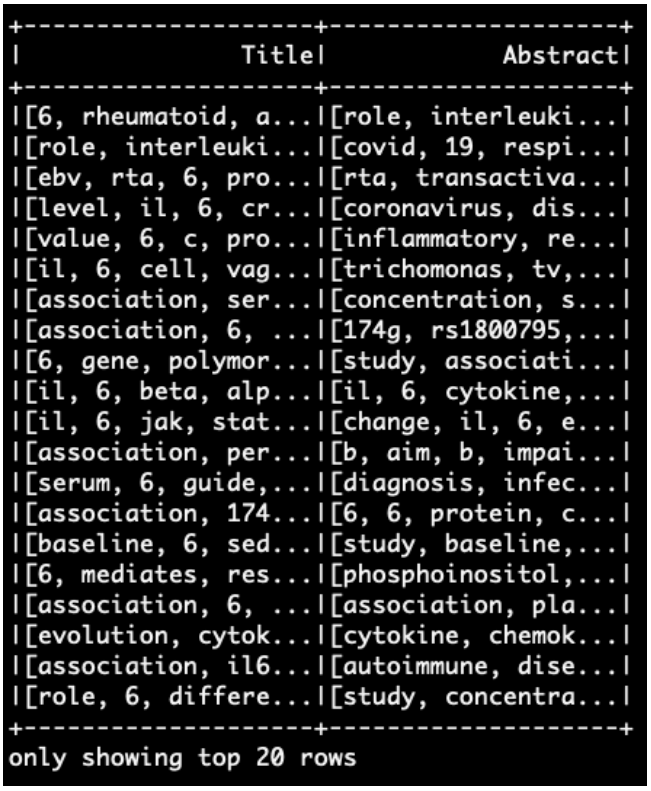


Figura 8: Esempio di part of speech tagging dopo il filtraggio

Riassumendo la funzione *posTagging(clean_paper_df)* - per ogni articolo presente nel dataframe - esegue i seguenti passi:

1. effettua il *part of speech tagging*;
2. rimuove i token che sono stati etichettati con tag non utili ai fini ultimi del programma;

Infine gli articoli cosi processati vengono memorizzati all'interno di un dataframe il quale conterrà sia per il titolo che per l'abstract una lista di token.



Title	Abstract
[6, rheumatoid, a...	[role, interleuki...
[role, interleuki...	[covid, 19, respi...
[ebv, rta, 6, pro...	[rta, transactiva...
[level, il, 6, cr...	[coronavirus, dis...
[value, 6, c, pro...	[inflammatory, re...
[il, 6, cell, vag...	[trichomonas, tv,...
[association, ser...	[concentration, s...
[association, 6, ...	[174g, rs1800795,...
[6, gene, polymor...	[study, associati...
[il, 6, beta, alp...	[il, 6, cytokine,...
[il, 6, jak, stat...	[change, il, 6, e...
[association, per...	[b, aim, b, impai...
[serum, 6, guide,...	[diagnosis, infec...
[association, 174...	[6, 6, protein, c...
[baseline, 6, sed...	[study, baseline,...
[6, mediates, res...	[phosphoinositol,...
[association, 6, ...	[association, pla...
[evolution, cytok...	[cytokine, chemok...
[association, il6...	[autoimmune, dise...
[role, 6, differe...	[study, concentra...

only showing top 20 rows

Figura 9: Esempio di dataframe dopo aver efettuato il part of speech tagging

2.5 Analisi della letteratura scientifica

Dopo aver effettuato il pre-processing di tutti gli articoli presenti nel dataframe, come spiegato nelle precedenti sezioni, si procede con la procedura di analisi per ricavare le malattie associate al gene d cui l'utente ha inserito in input l'ID. Tutto ciò viene fatto tramite la funzione *analyze_papers(clean_papers_df)*.

Per ricavare le malattie associate al gene viene utilizzata la *Named Entity Recognition* un processo utilizzato per identificare la classe di appartenenza di una parola all'interno di un certo documento. Per classe di appartenenza si intende cosa sta indicare quella parola, cioè la sua macrocategoria ad esempio persone, organizzazioni, luoghi, quantità, quantità di denaro.

Per applicare la *Named Entity Recogniton* è stata utilizzata la libreria *spacy*. Dato che il modello utilizzato da *spacy* non permetteva - dato un testo - di identificare le parole o le coppie di parole riconducibili a nomi di malattie in quanto quest'ultime venivano viste come semplici sostantivi. Per tale motivo si è utilizzato la libreria *scispacy* che contiene dei modelli *spacy* per il processing di testi biomedici, scientifici o clinici [1].

In particolare il modello scelto è stato il *en_ner_bc5cdr_md* un modello di *Named Entity Recognition* di *Spacy* addestrato sul corpus *BC5CDR*. È stato scelto tale modello in quanto le entità che riesce a riconoscere sono le malattie ed i composti chimici.

Una volta caricato il modello sopra menzionato, partendo dalla lista dei token inerenti il titolo e quella dei token inerenti l'abstract vengono create due stringhe in cui ogni singolo token sarà separato dal successivo tramite uno spazio. Infine le stringhe cosi ottenute vengono concatenate interponendo tra titolo ed abstract un carattere di newline. La stringa cosi formata sarà il documento sulla quale verrà applicata la *Named Entity Recognition*.

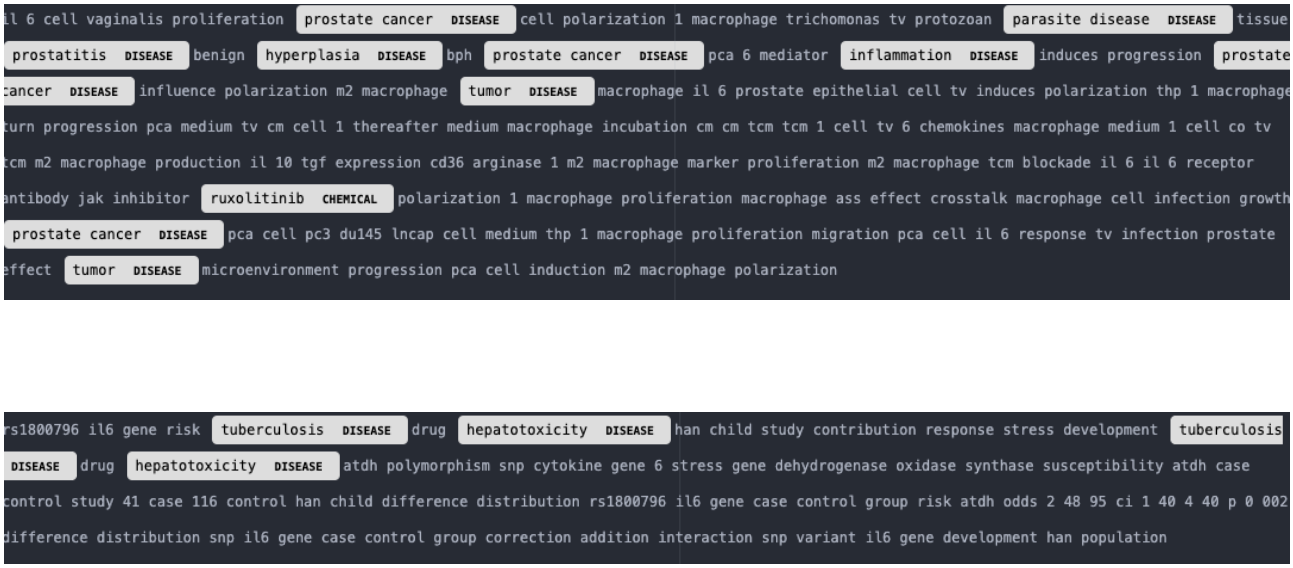


Figura 10: Esempio di Named Entity Recognition su di un articolo

[1] La libreria può essere reperita al seguente link: https://allenai.github.io/scispacy/

Dopo aver applicato la *Named Entity Recognitizion*, si vanno ad estrarre e ad inserire all'interno di una lista solo quelle parole che sono state identificate con l'entità *DISEASE*.

Figura 11: Esempio di lista di malattie ottenute dopo aver effettuato la Named Entity Recognition

Come si evince chiaramente dalla figura 11, la lista ottenuta ha bisogno di essere manipolata tramite delle opportune operazioni di filtraggio. Queste operazioni consistono nel:

- rimuovere i duplicati dalla lista;
- analizzare ogni singola stringa della lista ed eliminare eventuali parole che occorrono più di una volta all'interno della stessa stringa (ad esempio *breast cancer cancer* $\rightarrow$ *breast cancer*);
- eliminare dalla lista quelle parole che sono state identificate come malattie o patologie, ma che in realtà non lo sono. Questi termini possono essere termini biomedici, nomi di organi del corpo etc.

Riassumendo, la funzione emphanalyze_papers(clean_papers_df) applica i seguenti step:

1. applicazione della *Named Entity Recognition* su ogni articolo presente all'interno del dataframe ed estrazione delle parole che sono state etichettate con l'entità *DISEASE*;
2. pulitura della lista con rimozione di duplicati, pulitura delle singole stringhe, rimozione di termini non riconducibili a malattie o patologie.

La lista delle malattie ottenute oltre ad essere stata normalmente stampata, è stata utilizzata per creare una *wordcloud* - mediante la libreria *wordcloud* - per rappresentare graficamente e visivamente i risultati.

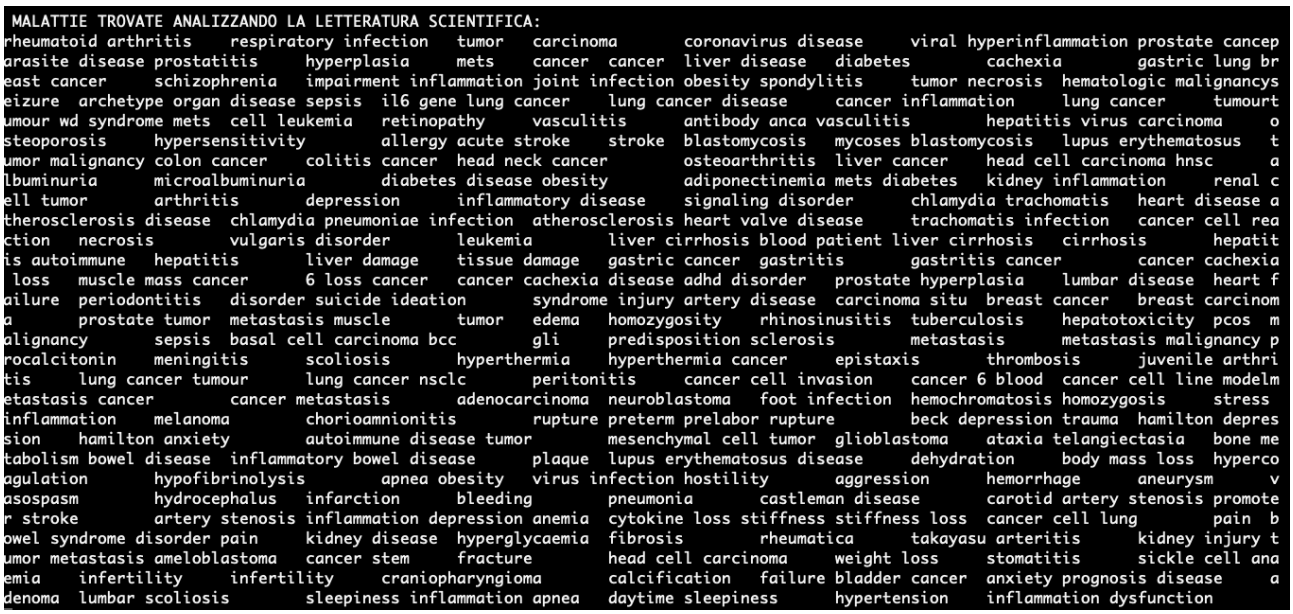


Figura 12: Esempio di lista delle malattie ottenute dopo la fase di pulitura dei dati

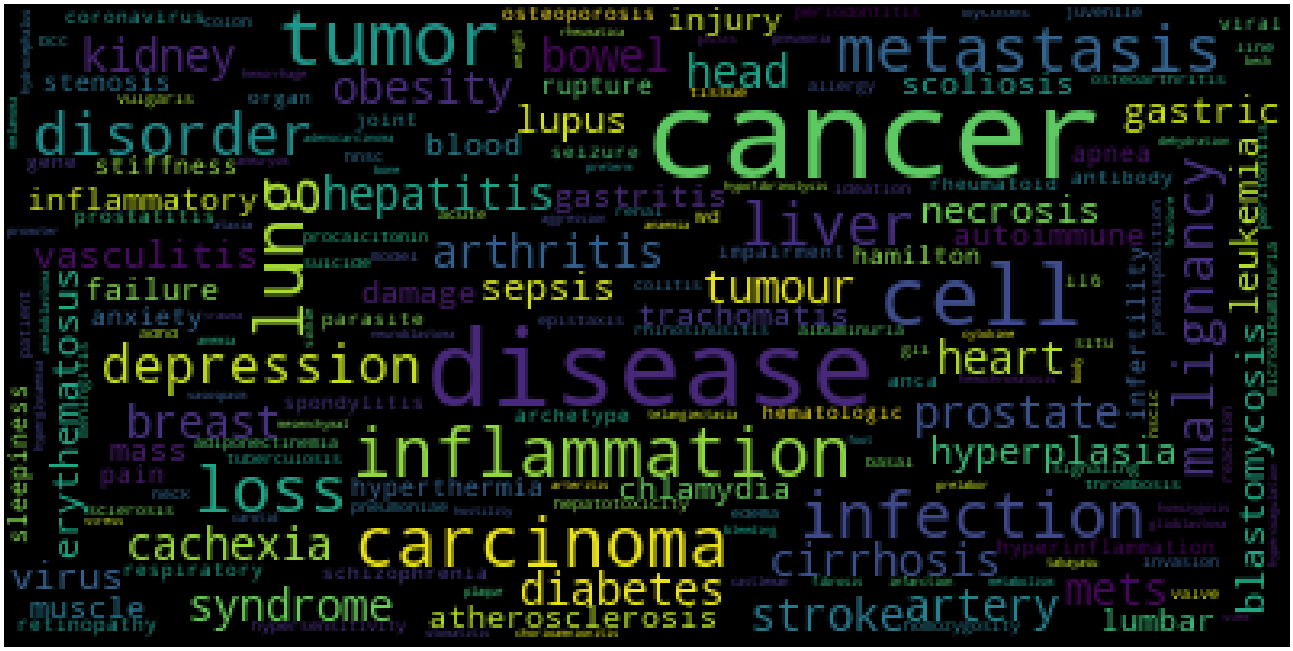
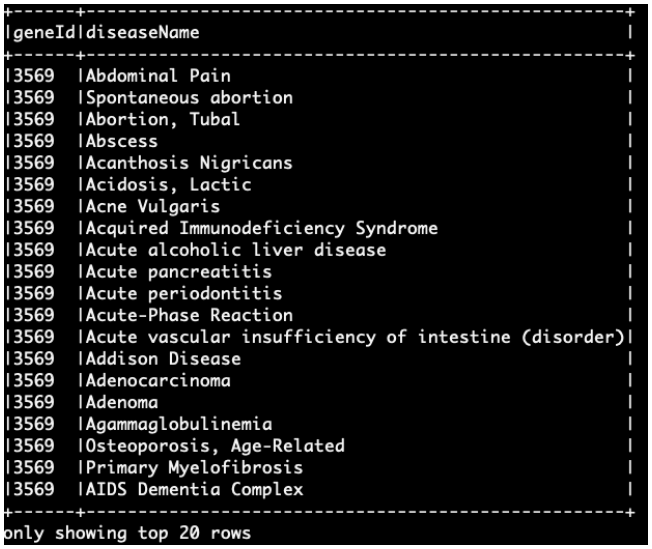


Figura 13: Esempio di wordcloud generata a partire dalla lista delle malattie

2.6 Valutazione dei risultati ottenuti

Per valutare l'attendibilità dei risultati ottenuti si è utilizzato il database *DisGenNet* [2] il quale contiene 1.134942 associazioni fra geni e malattie. In particolare contiene associazioni tra 21.671 geni e 30.170 malattie.

Il file contenente talli associazioni è in formato *TSV*. Per potere essere utilizzato è stato dunque importato all'interno di un dataframe sul quale sono state eseguite tutte le operazioni opportune di casting. Fatto ciò si è proceduto con un'ulteriore scrematura, filtrando il dataframe ed ottenendo solo i dati inerenti il gene che si sta analizzando. Da quest'ultimo data frame ottenuto è stato possibile ottenere la lista completa di tutte le malattie che sono associate al gene che si sta analizzando.



geneId	diseaseName
3569	Abdominal Pain
3569	Spontaneous abortion
3569	Abortion, Tubal
3569	Abscess
3569	Acanthosis Nigricans
3569	Acidosis, Lactic
3569	Acne Vulgaris
3569	Acquired Immunodeficiency Syndrome
3569	Acute alcoholic liver disease
3569	Acute pancreatitis
3569	Acute periodontitis
3569	Acute-Phase Reaction
3569	Acute vascular insufficiency of intestine (disorder)
3569	Addison Disease
3569	Adenocarcinoma
3569	Adenoma
3569	Agammaglobulinemia
3569	Osteoporosis, Age-Related
3569	Primary Myelofibrosis
3569	AIDS Dementia Complex

only showing top 20 rows

Figura 14: Esempio di dataframe con associazioni fra ID del gene e malattia

LISTA DELLE MALATTIE ASSOCIATE AL GENE con ID: 3569

Abdominal Pain Spontaneous abortion Abortion, Tubal Abscess Acanthosis Nigricans Acidosis, Lactic Acne Vulgaris Acquired Immunodeficiency Syndrome Acute alcoholic liver disease Acute pancreatitis Acute periodontitis Acute-Phase Reaction Acute vascular insufficiency of intestine (disorder) Addison Disease Adenocarcinoma Adenoma Agammaglobulinemia Osteoporosis, Age-Related Primary Myelofibrosis AIDS Dementia Complex AIDS related complex Albuminuria Alcohol Use Disorder Alcoholic Intoxication Alcoholic Intoxication, Chronic Alexithymia Alloxan Diabetes Alopecia Alveolar Bone Loss Extrinsic allergic alveolitis Alzheimer's Disease Ameloblastoma Amnesia Amyloidosis Amyotrophic Lateral Sclerosis anaphylaxis Anaplasia Anemia Anemia of chronic disease Aplastic Anemia Autoimmune hemolytic anemia Refractory anemias Anemia, Sickle Cell Aneurysm Angina Pectoris Angina, Unstable Fabry Disease Anorexia Anorexia Nervosa AnoxiaAnthracosilicosis Anthracosis Anuria Anxiety Anxiety Disorders Aortic Aneurysm Aortic Valve Insufficiency Aortic Valve Stenosis Apnea Appendicitis Cardiac Arrhythmia Arteriosclerosis Arteriovenous fistula Congenital arteriovenous malformation Arthralgia Arthritis Arthritis, Adjuvant-Induced Arthritis, Gouty Arthritis, Infectious Arthritis, Psoriatic Rheumatoid Arthritis Ascites Ascorbic Acid Deficiency Aspergillosis Asphyxia Neonatorum Asthma Astrocytoma Atherosclerosis Atrial Fibrillation Autistic Disorder Autoimmune Diseases Autoimmune state Melanoma, B16 Back Pain Nonproliferative diabetic retinopathy Bacteremia Bacterial Infections Pneumonia, Bacterial Barrett Esophagus Mental disorders Behavioral Symptoms Behcet Syndrome Berylliosis Cholestasis, Extrahepatic Biliary AtresiBipolar Disorder Malignant neoplasm of urinary bladder Bladder Neoplasm Blast Phase Blastomycosis Blood Coagulation Disorders Bloom Syndrome Bone Diseases Bone Diseases, Developmental Metabolic Bone Disorder Bone Marrow Diseases Bone neoplasmsBrain Diseases Brain Neoplasms Malignant neoplasm of breast Bronchiectasis Bronchiolitis Bronchiolitis Obliterans Bronchopulmonary Dysplasia Brucellosis Burkitt Lymphoma Burning Mouth Syndrome Bursitis Cachexia Cadmium poisoning Malignant Neoplasms Oral candidiasis Carcinoma Malignant tumor of colon Malignant neoplasm of endometrium Malignant neoplasm of larynx Adenocarcinoma of prostate Rectal Carcinoma Malignant neoplasm of skin Malignant neoplasm of thyroid Basal cell carcinoma Bronchioloalveolar Adenocarcinoma Bronchogenic Carcinoma Noninfiltrating Intraductal Carcinoma Non-Small Cell Lung Carcinoma Carcinoma, Papillary Renal Cell Carcinoma Squamous cell carcinoma Carcinoma, Transitional Cell Cardiomyopathy, Dilated Cardiovascular Diseases Carotid Artery Diseases Carotid Stenosis Celiac Disease Neoplastic Cell Transformation CNS disorder Central Nervous System Infection Intracranial Aneurysm Intracranial Arteriovenous Malformation Cerebral Infarction Brain Ischemia Transient Ischemic Attack Cerebral Palsy Cerebrovascular Disorders Malignant tumor of cervix Uterine Cervical Neoplasm Chagas Cardiomyopathy Chest Pain Chikungunya Fever Developmental Disabilities Primary biliary cirrhosis Cholelithiasis Cholera Cholestasis Chondrosarcoma Chorioamnionitis Choriocarcinoma Chorioretinitis Congenital chromosomal disease Chronic osteomyelitis Cochlear Diseases Cognition Disorders Colitis Ulcerative Colitis Collagen Diseases Colonic Neoplasms Colorectal Carcinoma Colorectal Neoplasms Common Cold Common Variable Immunodeficiency Condylomata Acuminata Conn Adenoma Connective Tissue Diseases Constipation Constitutional Symptom Febrile Convulsions Corneal Ulcer Coronary Aneurysm Coronary Arteriosclerosis Coronary heart disease Coronary Artery Vasospasm Coxsackievirus Infections Craniopharyngioma Craniosynostosis Crohn Disease Infection by Cryptococcus neoformans Cushing Syndrome Cystic Fibrosis Cystitis Cytomegalovirus Infections Pressur

Figura 15: Esempio di lista di malattie corrette associate al gene

[2] Il database è reperibile al seguente link: https://www.disgenet.org/downloads

Per stabilire se una certa malattia che è stata identificata analizzando la letteratura scientifica può essere etichettata come corretta viene utilizzato il *fuzzy string matching* tra due stringhe. In particolare viene utilizzata la libreria *Python fuzzywuzzy* che utilizza la distanza di *Levenshtein* per calcolare la sequenza tra due stringhe. Date due stringhe x e y, la distanza di *Levenshtein* è il numero minimo di modifiche elementari che ci permettono di trasformare x in y. Per modifiche elementari si intendono:

- cancellazione di un carattere;
- sostituzione di un carattere con un altro;
- inserimento di un carattere.

La funzione utilizzata per paragonare le due liste di malattie - quella ottenuta utilizzando l'analisi scientifica e quella delle malattie corrette ottenuta dal dataframe *DisGenNet* - è *evaluate_result(result, correct_result).*

Tale funzione prese in input due liste:

1. *result*: lista delle malattie ottenute analizzando la letteratura scientifica;
2. *correct_result*: lista delle malattie associate al gene ottenuta dal dataframe *DisGenNet.*

Per calcolare quale malattia della prima lista può essere etichettata come corretta e calcolare dunque la percentuale di malattie esatte trovate vengono paragonate a coppe le stringhe della prima e della seconda lista utilizzando la funzione di *fuzzywuzzy emph.token_set_ratio(x,y)*, la quale restituisce in output uno score che ci indica il grado di similarità tra le due stringhe che la funzione prende input.

Verranno aggiunte alla lista finale delle malattie solo quelle stringhe che confrontate con le stringhe della seconda lista superano una certa soglia, in questo caso è stato scelto come valore di soglia 80. In sintesi, la funzione *evaluate_result(result, correct_result)* esegue i seguenti step:

```
('rheumatoid arthritis', 'Arthritis', 100)
('respiratory infection', 'Respiratory Tract Infections', 86)
('tumor', 'Malignant tumor of colon', 100)
('carcinoma', 'Carcinoma', 100)
('prostate cancer', 'Prostate cancer recurrent', 100)
('parasite disease', 'Parasitic Diseases', 88)
('prostatitis', 'prostatitis', 100)
('hyperplasia', 'Angiolymphoid hyperplasia', 100)
('cancer', 'Hypopharyngeal Cancer', 100)
('liver disease', 'Acute alcoholic liver disease', 100)
('diabetes', 'Alloxan Diabetes', 100)
('cachexia', 'Cachexia', 100)
('gastric lung breast cancer', 'Early gastric cancer', 82)
('schizophrenia', 'Schizophrenia', 100)
('impairment inflammation', 'Inflammation', 100)
('joint infection', 'Prosthetic joint infection', 100)
('obesity', 'Obesity', 100)
('spondylitis', 'Spondylitis', 100)
('tumor necrosis', 'Necrosis', 100)
('seizure', 'Jacksonian Seizure', 100)
('sepsis', 'Sepsis', 100)
('lung cancer disease', 'Chronic lung disease', 82)
('cancer inflammation', 'Inflammation', 100)
('lung cancer', 'Progression of non-small cell lung cancer', 100)
('tumour', 'Tumour inflammation', 100)
('tumour wd syndrome mets', 'Job Syndrome', 80)
('cell leukemia', 'leukemia', 100)
('retinopathy', 'Nonproliferative diabetic retinopathy', 100)
('vasculitis', 'Vasculitis', 100)
('antibody anca vasculitis', 'Vasculitis', 100)
('hepatitis virus carcinoma', 'Carcinoma', 100)
('osteoporosis', 'Osteoporosis, Age-Related', 100)
```

Figura 16: Esempio di fuzzy string matching

1. crea la lista finale delle malattie, effettuando il *fuzzy string matching* tra le due liste di malattie, quella ottenuta analizzando la letteratura scientifica e quella ottenuta dal database *DisGenNet*;

2. calcola la percentuale di quante malattie identificate sono corrette.

La lista delle malattie cosi ottenute, insieme alla percentuale calcolata oltre ad essere stata normalmente stampata, è stata utilizzata per creare una *wordcloud* per rappresentare graficamente e visivamente i risultati.

```
DELLE MALATTIE IDENTIFICATE SOLO IL 71.63 % SONO RISULTATE CORRETTE:
rheumatoid arthritis    respiratory infection   tumor   carcinoma       prostate cancer parasite disease        prostatitis     hyperpl
asia    cancer  liver disease   diabetes        cachexia        gastric lung breast cancer      schizophrenia   impairment inflammatioj
oint infection  obesity spondylitis     tumor necrosis  seizure sepsis  lung cancer disease     cancer inflammation     lung cancer   t
umour   tumour wd syndrome mets cell leukemia   retinopathy     vasculitis      antibody anca vasculitis        hepatitis virus carcino
ma      osteoporosis    hypersensitivity        allergy stroke  blastomycosis   mycoses blastomycosis   lupus erythematosus     colon c
ancer   colitis cancer  osteoarthritis  head cell carcinoma hnsc        albuminuria     microalbuminuria        diabetes disease obesit
y       mets diabetes   kidney inflammation     arthritis       depression      inflammatory disease    signaling disorder      chlamyd
ia trachomatis  heart disease   atherosclerosis disease chlamydia pneumoniae infection  atherosclerosis heart valve disease     trachom
atis infection  necrosis        vulgaris disorder       leukemia        liver cirrhosis blood patient liver cirrhosis   cirrhosis     h
epatitis autoimmune     hepatitis       gastric cancer  gastritis       gastritis cancer        cancer cachexia loss    cancer cachexia
 disease        adhd disorder   lumbar disease  heart failure   periodontitis   disorder suicide ideation       syndrome injury artery
disease carcinoma situ  breast cancer   breast carcinoma        prostate tumor  edema   tuberculosis    malignancy      basal cell carc
inoma bcc       metastasis      meningitis      scoliosis       thrombosis      juvenile arthritis      peritonitis     cancer cell inv
asion   adenocarcinoma  neuroblastoma   foot infection  hemochromatosis stress inflammation     melanoma        chorioamnionitis      r
upture  beck depression hamilton depression     hamilton anxiety        autoimmune disease tumor        glioblastoma    ataxia telangie
ctasia  inflammatory bowel disease      plaque  lupus erythematosus disease     dehydration     apnea obesity   virus infection hemorrh
age     aneurysm        vasospasm       hydrocephalus   infarction      pneumonia       castleman disease       carotid artery stenosia
rtery stenosis  inflammation depression anemia   cancer cell lung        pain    bowel syndrome disorder pain    kidney disease  hypergl
ycaemia fibrosis        rheumatica      takayasu arteritis      kidney injury   tumor metastasis        ameloblastoma   head cell carci
noma    stomatitis      sickle cell anaemia     infertility     craniopharyngioma       calcification   failure anxiety adenoma lumbar
scoliosis       sleepiness inflammation apnea   hypertension    inflammation dysfunction
```

Figura 17: Esempio di lista finale delle malattie ottenute

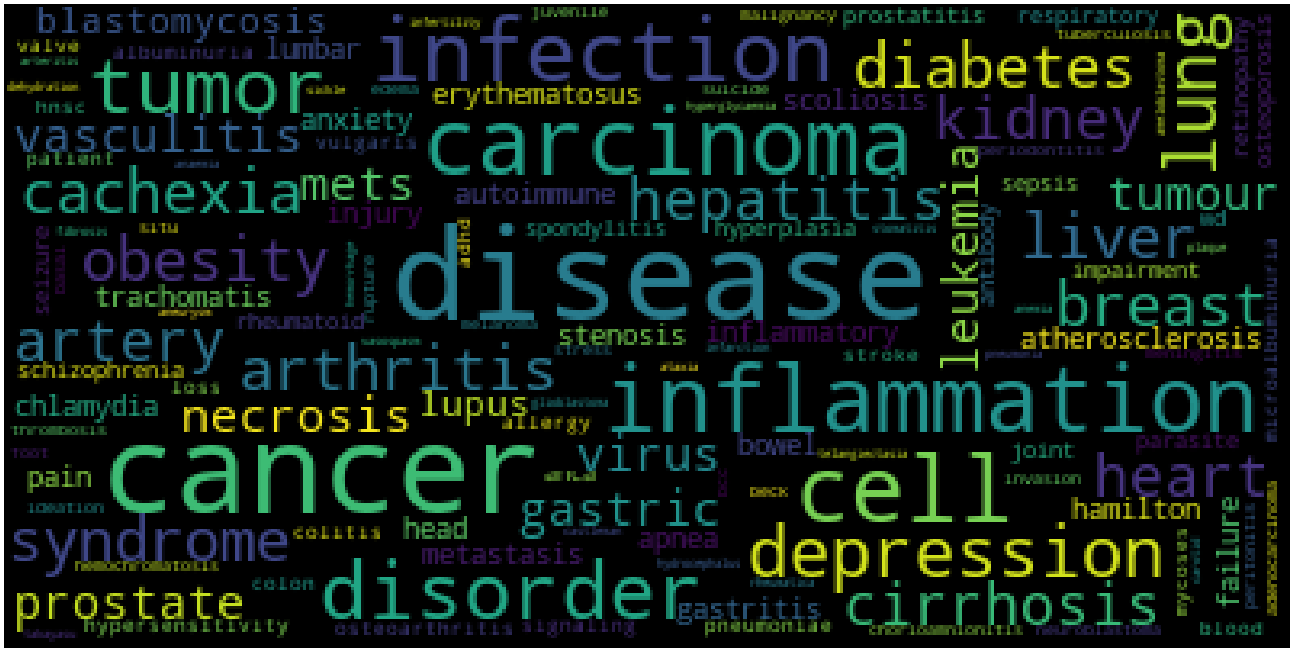


Figura 18: Esempio di wordcloud generata a partire dalla lista finale delle malattie

3 Conclusioni

Nella realizzazione di questo progetto è stato realizzato un software che preso un gene in input (il suo ID), una volta ricavate le informazioni inerenti tali gene ed estratta la letteratura scientifica ad esso annessa da *PubMed*, tramite tecniche di *Natural Language Processing*, *Text Mining* e *Sentiment Analisys* restituisce in output un insieme di malattie associate al gene.

In particolare le tecniche più importanti utilizzate sono state:

- *part of speech tagging*;
- *named enity recognition*.

Per valutare la bontà dei risultati ottenuti è stato utilizzato il database *DisGenNet*. Nella maggior parte dei casi più del 50% delle malattie ottenute analizzando la letteratura scientifica corrisponde con quelle ottenute dal database *DisGenNet*. Nella lista finale delle malattie potrebbe capitare di riscontrare nomi - che pur essendo riconducibili a malattie o patologie - sono stati erroneamente inseriti nella lista (ad esempio nomi di due malattie concatenate, oppure due malattie che non hanno nulla in comune identificate come un'unica malattia). Questo potrebbe derivare dal fatto che la maggior parte delle tecniche utilizzate per implementare questo software - ad esclusione di qualcuna come ad esempio la *Named Entity Recognition* - utilizzano modelli che non sono addestrati su corpus di testo contenente prettamente terminologia biomedica. Il software dunque, potrebbe essere migliorato utilizzando modelli che per applicare le tecniche utilizzate usino modelli addestrati prettamente per questo scopo.