POLITECNICO DI TORINO

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Natural Language Processing per l'estrazione di informazioni mediche dai testi

Caso di studio ASL Modena

Relatore:
prof. Paolo Garza

Candidato:
Marco Balestri

Supervisore aziendale
Engineering Ingegneria Informatica S.p.A.
dott. Isabella Iennaco

Anno accademico 2018-2019

Sommario

Il Natural Language Processing è il processo di trattamento automatico mediante un calcolatore delle informazioni scritte in una lingua naturale. La tesi consiste nell'applicazione di tecniche di Natural Language Processing al fine di estrarre informazioni da testi liberi di provenienza medica.
La prima parte presenta una serie di tecniche ed algoritmi per il preprocessing e l'elaborazione dei testi e introduce il problema dell'applicazione del machine learning a dati non strutturati di tipo testuale. Viene inoltre descritto lo stato dell'arte per quanto riguarda alcuni dei principali algoritmi di machine learning.
La seconda parte descrive come tali tecniche sono state combinate ed utilizzate per trattare e risolvere il problema in oggetto. Vengono infine spiegate ed argomentate le scelte implementative ed analizzati i risultati ottenuti.

Ringraziamenti

Un sincero ringraziamento alla dott. Cazzin per l'opportunità concessami ed a tutto il gruppo di ricerca di Engineering Ingegneria Informatica S.p.A. per i mezzi messi a disposizione e per le informazioni riservate gentilmente fornite, ed in particolare alla dott. Iennaco per le utili discussioni che hanno permesso di svolgere al meglio il mio lavoro.

Indice

Elenco delle tabelle

Elenco delle figure

Parte I

Prima Parte

Capitolo 1

Introduzione

1.1 Principi generali

Si può affermare con certezza che tra le grandi rivoluzioni tecnologiche della storia recente vi sia l'invenzione dei computer.
La diffusione delle reti di telecomunicazioni e lo sviluppo di calcolatori via via più potenti hanno totalmente stravolto il nostro modo di vivere e di lavorare. Questi strumenti ci forniscono mezzi molto potenti per svolgere compiti estremamente sofisticati in tempi estremamente brevi; per queste ragioni le macchine sono entrate velocemente a far parte della vita comune delle persone, diventando sempre più indispensabili per gli usi più disparati.

Oltre alla potenza di elaborazione i computer iniziarono a fornire anche un modo per memorizzare ed organizzare i dati in maniera efficace. L'evoluzione della ricerca tecnologica ha quindi portato alla consapevolezza che la conoscenza (il possesso dei dati) potesse aprire le porte a nuovi scenari se unita efficacemente alla potenza di calcolo. [9]
Sulla base di questa idea nacque quindi il concetto di **machine learning**, ossia lo sviluppo di algoritmi che permettano ai calcolatori di sfruttare le grandi quantità di dati in loro possesso per estrarne conoscenza utile ad applicazioni di vario tipo.
E' quindi sull'onda di queste innovazioni che al giorno d'oggi acquisiscono sempre più importanza i dati, i quali rappresentano una condizione necessaria e fondamentale per l'implementazione di *"intelligenze artificiali"*.

Sebbene tradizionalmente gli algoritmi fossero sviluppati attorno a dati di tipo numerico e strutturato, ben presto si iniziò a comprendere che la maggior parte dell'informazione generata nel mondo reale fosse contenuta all'interno di documenti *non strutturati*.

Fotografie, disegni, testi scritti, registrazioni audio e video sono esempi di contenitori di dati di tipo eterogeneo e non strutturato che potrebbero contenere al loro interno grandi quantità di informazioni. Proprio per sfruttare queste informazioni sono stati sviluppati negli ultimi anni nuovi algoritmi e tecniche capaci di estendere il più possibile le applicazioni del machine learning anche a dati di questo tipo.

1.2 L'intelligenza artificiale

L'obiettivo di questa tesi è lo sviluppo di tecniche ed algoritmi capaci di combinare diversi approcci (statistici e rule-based) al fine di estrarre informazioni da dati non strutturati di tipo testuale.
In particolare i testi in questione sono referti redatti manualmente dai medici al termine di operazioni di tipo ospedaliero. L'obiettivo finale è quello di identificare all'interno di essi la presenza di informazioni utili per fini statistici e per l'analisi del comportamento e delle abitudini dei vari specialisti, e cercare di estrarle in maniera efficace.
Il comportamento da riprodurre deve simulare quanto più fedelmente possibile quello di un'intelligenza "umana", capace di leggere i testi, comprenderne il contenuto ed estrapolare le informazioni richieste.

Le tecniche sviluppate cercano quindi di combinare approcci di tipo statistico e modelli di apprendimento automatico con i più tradizionali approcci che prevedono l'uso di dizionari, al fine di costruire un procedimento in grado di svolgere il compito richiesto in maniera efficace in termini di tempo e prestazioni.
Più in dettaglio il progetto si basa sulla creazione di un modello che sia in grado, osservando i testi, di identificare autonomamente al loro interno un insieme di *"topic"* (o tematiche), di comprendere quale sia l'argomento trattato da ciascuno di essi e successivamente di riconoscere all'interno dei vari testi la presenza dei *topic* sopra identificati.

In seguito verranno anche implementate delle metodologie atte ad estrarre informazioni utili legate alla presenza di determinati farmaci, come ad esempio l'associazione ad ognuno di essi della quantità prescritta e/o della durata della terapia.

In letteratura sono presenti diversi algoritmi riguardanti la **classificazione** ed il **topic modeling** di interi documenti (ad esempio recensioni o articoli di giornale), i quali spesso necessitano di un insieme di dati già etichettati o evidenziati manualmente da poter usare in fase di addestramento.

Lo scopo di questa tesi è riuscire a sviluppare tecniche ed algoritmi in grado di ottenere risultati simili su singole frasi o pezzi di testo anziché sugli interi documenti, e senza la necessità di possedere dati già etichettati da usare come base di conoscenza.

1.3 Struttura della tesi

L'organizzazione del lavoro svolto ha la seguente struttura dei capitoli:

- Nel *secondo capitolo* si presenta il tema del NLP/NLU e si descrive un quadro generale sulle tecniche di Text Mining e i rapporti con il Data Mining.
- Il *terzo capitolo* espone le tecniche di text preprocessing spiegando quali siano i benefici ad esse dovuti.
- Nel *quarto capitolo* viene presentato lo stato dell'arte riguardante la classificazione ed il topic modeling di dati di tipo testuale.
- Nel *quinto capitolo* vengono illustrati gli strumenti hardware e software utilizzati durante lo svolgimento del lavoro.
- Il *sesto capitolo* presenta la spiegazione di come i vari algoritmi e le tecniche citate in precedenza siano stati utilizzati e combinati tra loro per riuscire a creare un modello da applicare al caso di studio in oggetto.
- Il *settimo capitolo* riporta le riflessioni conclusive sul lavoro svolto e fornisce spunti per eventuali miglioramenti, sviluppi ed estensioni future.

Capitolo 2

NLP e NLU

Il Natural Language Processing è una disciplina che si pone a metà tra l'informatica, l'intelligenza artificiale e il moderno concetto di "data mining". Si focalizza sul capire come gli esseri umani possano programmare dei computer per processare grandi quantità di dati rappresentati sotto forma di linguaggio naturale (come ad esempio un testo scritto o una conversazione orale) in una maniera che sia produttiva ed efficiente, con l'obiettivo di spostare alcune attività dalle mani degli esseri umani, delegandole alle macchine.

Spesso considerato come un sotto-argomento del Natural Language Processing, il Natural Language Understanding (NLU) è una parte vitale del successo dell'NLP.
Lo scopo dell'NLU è più specifico, e riguarda in particolare la capacità delle macchine di comprendere il contenuto e il vero significato di un testo o di una conversazione. Esso può essere applicato ad una serie di processi, come la classificazione dei testi, la raccolta di informazioni e l'analisi dei contenuti.
Il NLU può essere visto come una base di partenza per tutte quelle attività che si basano sul linguaggio naturale: per essere in grado di utilizzare delle informazioni esse devono prima di tutto essere estratte e comprese dalla persona o dalla macchina incaricata di elaborarle.

L'aspirazione finale di queste discipline è riuscire a realizzare un sistema che combini la potenza di elaborazione delle macchine (velocità di elaborazione, capacità di processare grandi quantità di dati) con un'intelligenza di tipo human-like. [13]

2.1 Storia

La storia del natural-language processing inizia negli anni '50, nonostante si possano trovare fonti antecedenti a questa data. Nel 1950 Alan Turing pubblicò un articolo intitolato "Intelligence" che proponeva come criterio di intelligenza quello che ora è conosciuto come Turing test.

Nel 1954 venne condotto l'esperimento di Georgetown, il quale richiedeva una traduzione automatica completa di più di sessanta frasi dal russo all'inglese. Gli autori sostenevano che nel giro di cinque anni il problema della traduzione automatica di testi sarebbe stato risolto.
Ciononostante i progressi reali furono decisamente più lenti e nel 1966, dopo che una ricerca di più di dieci anni non riuscì a mantenere le aspettative, i fondi per la traduzione automatica furono drasticamente ridotti e la ricerca venne sostanzialmente interrotta fino agli anni '80, quando vennero sviluppati i primi sistemi statistici di traduzione automatica.

Fino ad allora infatti la maggior parte dei sistemi di natural-language processing erano basati su insiemi complessi di regole scritte a mano; a cominciare dalla fine degli anni '80 però ci fu una rivoluzione nell'ambito del Natural Language Processing grazie all'introduzione degli algoritmi di machine learning: ciò fu possibile grazie alla crescita delle capacità computazionali a disposizione (vedi Legge di Moore).
Alcuni dei primi algoritmi di machine learning usati, per esempio gli alberi di decisione, producevano sistemi di rigidi costrutti if-then simili alle già esistenti regole scritte manualmente dai programmatori.
Tuttavia col passare del tempo i ricercatori iniziarono a focalizzarsi maggiormente su modelli statistici in grado di prendere decisioni probabilistiche basate su valori di pesi attribuiti a un set di dati reali forniti in input al modello.
Molti dei sistemi di riconoscimento vocale ancora oggi si basano su questi concetti approfittando del fatto che essi garantiscono una forte robustezza al rumore e un risultato più affidabile in presenza di input non familiari o errati.

Grazie al lavoro svolto dai ricercatori IBM un grande successo venne raggiunto nell'ambito della traduzione automatica, dove vennero sviluppati modelli statistici sempre più accurati e capaci di utilizzare decine di documenti multilingue già esistenti come base di conoscenza per imparare a crearne di nuovi.
Tuttavia la grande quantità di dati necessari in input rappresenta un limite per questo tipo di modelli, per questa ragione gran parte degli sforzi dei ricercatori negli ultimi anni si sono concentrati sulla creazione di modelli capaci di essere addestrati efficacemente anche partendo da quantità limitate di dati.

Un altro campo che ha acquisito sempre più importanza col passare del tempo è quello dell'apprendimento non supervisionato. Gli algoritmi unsupervised sono in grado di apprendere da dati non etichettati manualmente. Generalmente questa attività è molto più complicata rispetto all'apprendimento supervisionato e tipicamente produce risultati meno accurati in termini di precisione.
Tuttavia l'enorme quantità di dati non etichettati presenti nel mondo reale conferisce sempre più importanza a questo argomento.
Negli ultimi dieci anni con l'arrivo del deep learning e delle reti neurali nuovi scenari e nuove frontiere si sono aperte anche nell'ambito del Natural Language Processing. [24]

2.2 Cos'è il text mining?

Il text mining è una branca del Natural Language Processing che si occupa in particolare dell'estrazione di informazioni da input di tipo **testuale**.
Esso è definito come il processo di esplorazione e analisi di grandi quantità di dati testuali non strutturati grazie all'aiuto di software specifici che riescano ad identificare concetti, patterns, topics, parole chiave e altre peculiarità dei dati.
L'obiettivo della ricerca riguardante il text mining è quindi studiare nuovi metodi e algoritmi per **estrarre automaticamente conoscenza dal testo**, al fine ad esempio di classificare o raggruppare documenti in base ai contenuti.

«La scoperta da parte di un computer di nuovi, in precedenza sconosciute informazioni, attraverso l'estrazione automatica di differenti documenti scritti» [10]

Per natura il text mining è simile al data mining, anche se si concentra su testi che in genere non sono strutturati.

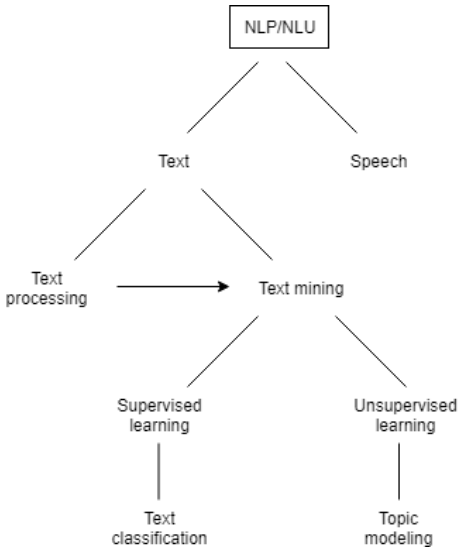

Figura 2.1. Rappresentazione schematica delle relazioni tra le varie branche del NLP/NLU

2.3 Differenza tra text mining e data mining

Il **data mining** può essere liberamente definito come la ricerca di pattern nei dati (tipicamente numerici), il **text mining** invece è la ricerca di pattern nel testo. La somiglianza tra i due tuttavia nasconde reali differenze. [6]

Il data mining può essere meglio descritto come l'estrazione dai dati di informazioni *precedentemente sconosciute* e potenzialmente utili. L'informazione contenuta nei dati è implicita, difficile se non impossibile da estrarre senza ricorrere a tecniche automatiche di data mining.

Nel text mining invece, le informazioni da estrarre sono *esplicitamente e chiaramente indicate nel testo* e pertanto facilmente utilizzabili da parte degli esseri umani.
Il problema, naturalmente, è che le informazioni non sono formulate in un modo strutturato e quindi non è possibile utilizzare tecniche di elaborazione automatica.

Per questa ragione grazie all'utilizzo di tecniche di text mining si tenta di estrarre il testo in formati adeguati per l'utilizzo di algoritmi di data mining. Alla luce di ciò risulta quindi che nel Data Mining le informazioni possedute in ingresso sono archivi di dati strutturati, mentre al contrario il text mining dispone di un insieme di documenti testuali liberi.

In conseguenza di quanto affermato finora nell'ambito della disciplina del text mining assumono quindi grande importanza operazioni preliminari alla vera e propria elaborazione dei testi: tutte le manipolazioni e le trasformazioni necessarie vengono dunque effettuate sui documenti nella cosiddetta fase di *pre-processing*.

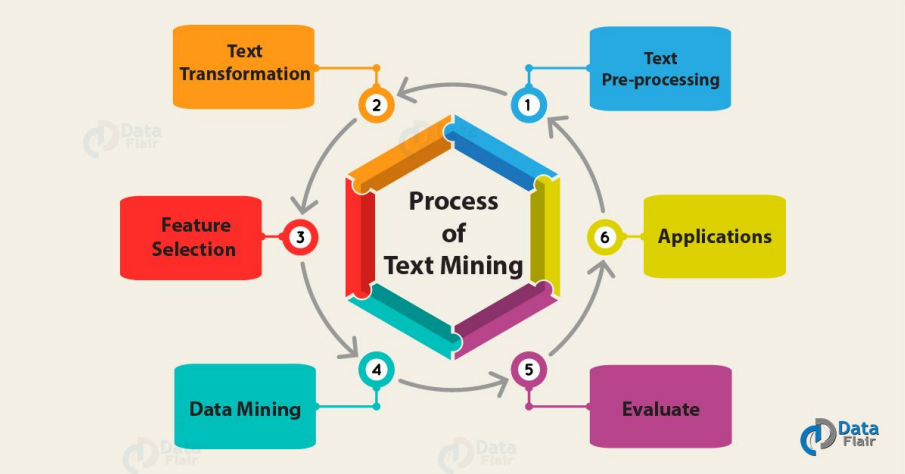

Figura 2.2. Il processo del text mining.

2.4 A cosa serve il text mining?

L'esponenziale crescita della capacità di elaborare dati ed estrarne conoscenza ha reso il possesso e la disponibilità di dati una fonte di ricchezza primaria. A questo punto quindi gioca un ruolo fondamentale la capacità di saper gestire questi dati e soprattutto di poterli elaborare in ogni loro forma.

In questo contesto viene dunque inserito il text mining, dal momento che si crede che il suo potenziale commerciale sia superiore a quello dell'estrazione di dati strutturati.
Basta infatti pensare che la maggior parte della produzione d'informazioni di una società/azienda proviene da chat, e-mail, blog e forum e quindi testo scritto in linguaggio naturale. Al contrario solamente una piccola percentuale di informazione è espressa in dati numerici.

Con tali prospettive di sviluppo solo grazie al text mining si riesce a dare maggiore efficacia e robustezza all'analisi automatica dei documenti poiché essa è l'unica tecnologia che esegue uno studio approfondito del "significato" del testo.

2.5 Esempi di applicazione del text mining

Il text mining proprio grazie alla sua duttilità può essere utilizzato in una vasta gamma di contesti diversi, dal classificare brevi recensioni o pezzi di testo (ad esempio tweets o titoli di giornale) all'analizzare documenti ben più corposi (ad esempio interi articoli di giornale o contratti ufficiali).
Tra le più note applicazioni del text mining vi sono ad esempio sentiment analysis, topic labeling, language detection e intent detection.

- *Sentiment analysis* è probabilmente la più nota applicazione del text mining e mira ad identificare il sentimento di un cliente nei confronti di un'azienda. L'analisi del sentimento attinge dati da recensioni online, social networks, email, interazioni coi call center ed altre fonti e cerca di identificare argomentazioni comuni che testimoniano soddisfazione o insoddisfazione da parte dei clienti. Grazie a queste informazioni è possibile tra le altre cose sistemare eventuali errori nel prodotto, migliorare il servizio clienti e pianificare nuove campagne di marketing.

- Un altro esempio di utilizzo comune è il *topic labeling* che consiste nel capire quale argomento sia trattato all'interno di un dato testo. Viene spesso usato per organizzare dati come recensioni e articoli di giornale in base alla materia in questione.

- Il *language detection* consiste nel determinare quale lingua sia usata in un determinato testo. Ciò può essere utile ad esempio al fine di capire quale correttore automatico utilizzare, o a quale servizio clienti affidare un determinato ticket.

- L'*intent detection* viene usata da molte aziende per capire quale sia la reale intenzione dei propri clienti (ad esempio per capire se una persona è intenzionata ad un prodotto oppure se un cliente intende passare alla concorrenza), in maniera da sviluppare strategie di marketing mirate e specifiche.

2.6 Supervised vs unsupervised learning

All'interno del campo del machine learning, ci sono due principali tipologie di attività: *supervisionata* e *non supervisionata*.

La principale differenza tra le due sta nel fatto che l'**apprendimento supervisionato** è fatto usando una ground truth, o in altre parole una *conoscenza a priori* di quali debbano essere i valori forniti in output dal modello. Per questa ragione l'obiettivo del supervised learning è apprendere una funzione che, dato un campione

di dati e degli output desiderati, approssima al meglio la relazione osservabile nei dati tra input e output.
Esso è tipicamente utilizzato nei problemi di classificazione, quando vogliamo mappare valori in input su rispettive label in output, o di regressione, quando vogliamo mappare un input su un output continuo.
Sia nella classificazione che nella regressione l'obiettivo è trovare una specifica relazione o struttura nei dati in input che permetta di produrre efficacemente l'etichetta corretta. La correttezza dell'output si basa interamente sulla struttura dei dati di training, che rappresentano la cosiddetta *ground truth* o conoscenza a priori.
Pertanto avere un training set contenente dati scorretti o rumorosi comprometterà l'efficacia del modello, dal momento che esso assumerà i dati forniti come verità assoluta.
Tra i più diffusi algoritmi di apprendimento supervisionato troviamo Logistic Regression, Naive-Bayes, SVM, Random Forest e le reti neurali.

L'**unsupervised learning** al contrario non possiede dati etichettati e pertanto non ha nessuna conoscenza a priori di quale debba essere la struttura dell'output.
Per questa ragione l'obiettivo dell'apprendimento non supervisionato è proprio inferire quale sia la struttura naturale che accomuna e meglio rappresenta un insieme di dati.
Nell'unsupervised learning non viene fornita in input alcuna etichetta. I due principali obiettivi degli unsupervised learning algorithms sono il clustering e la dimensionality reduction.
Il clustering consiste nel dividere i dati in gruppi (cluster) in modo che oggetti dello stesso gruppo siano più simili tra loro rispetto a quelli degli altri cluster. Esempi di algoritmi di clustering sono DBSCAN , k-means clustering e clustering gerarchico.
La dimensionality reduction consiste nel ridurre drasticamente la quantità di dati in input in modo da rendere affrontabile la complessità computazionale e il tempo di processing. L'obiettivo è di conservare quanta più informazione possibile, eliminando invece dati inutili o informazioni ridondanti. Alcuni esempi di algoritmi sono PCA, SVD, t-SNE.

Esistono infine dei casi reali in cui i due approcci vengono combinati, dando così vita al cosiddetto apprendimento **semi-supervisionato**.
Tipicamente nei problemi reali la quantità di dati etichettati a disposizione è estremamente esigua (poiché le etichette devono essere prodotte manualmente da esseri umani) mentre al contrario esistono grandi quantità di dati non etichettati.
Per questa ragione molti ricercatori hanno capito che i dati non etichettati, quando usati in abbinamento a dati etichettati, possono fornire considerevoli miglioramenti nella precisione dei modelli rispetto all'uso esclusivo di una delle due tecniche. [3]

2.7 Cos'è la text classification?

Nell'ambito del supervised learning la classificazione testuale è il processo di assegnare *tag* o *etichette* al testo libero a seconda del suo contenuto.
E' uno dei task fondamentali all'interno del NLP con varie applicazioni come ad esempio sentiment analysis, topic labeling, spam detection.
I classificatori testuali possono essere usati per organizzare, strutturare e classificare articoli, chat, recensioni e molti altri tipi di testi.

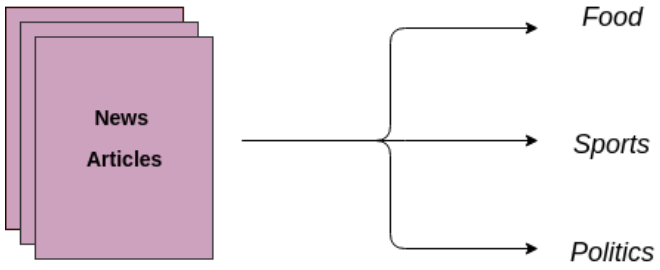

Figura 2.3. Classificazione di testi.

2.8 Cos'è il topic modeling?

Mentre la text classification è un algoritmo di apprendimento automatico supervisionato, nel quale ogni documento viene classificato entro un insieme di classi predefinite, il topic modeling è il processo di scoprire topic astratti ricorrenti in una collezione di documenti.
Dal momento che l'insieme dei vari topic non è conosciuto a priori esso è un modello di apprendimento **non supervisionato**.

Nell'ambito del Natural Language Processing il *topic modeling* viene effettuato grazie all'uso di modelli statistici basati sulla distribuzione e sulla co-occorrenza di parole, frasi e strutture semantiche all'interno dei documenti.
Intuitivamente dato il topic di un documento, ciò che ci si aspetta è che determinate parole siano presenti all'interno di questo più o meno frequentemente: ad esempio le parole "palla" e "giocatore" appariranno più di frequente in documenti sportivi e meno frequentemente in quelli di politica. Di conseguenza i topic prodotti dalle tecniche di topic modeling sono cluster di parole simili tra loro.
I modelli statistici raccolgono questi concetti all'interno di strutture matematiche che

permettono in seguito di analizzare insiemi di documenti e scoprire, in base alla distribuzione delle parole in essi contenute, a quale dei diversi topic essi appartengano. [1]

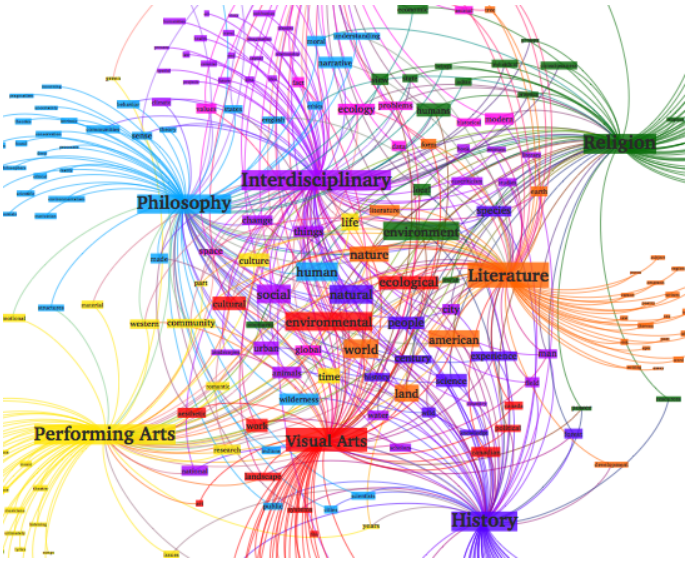

Figura 2.4. Topic modeling.

2.9 Rule-based vs statistical NLP

Nei primi anni di ricerca la maggior parte dei sistemi di language processing si basavano su regole progettate ed implementate manualmente dai ricercatori, seguendo regole grammaticali o ricavando regole euristiche.
Queste metodologie erano scarsamente robuste alle variazioni e alle numerose eccezioni del linguaggio naturale. [1]
Tuttavia dal momento della cosiddetta “rivoluzione statistica” avvenuta tra gli anni ’80 e ’90, il natural language processing ha iniziato a basarsi sempre più fortemente sul machine learning. L’apprendimento automatico infatti usa deduzioni statistiche per apprendere automaticamente le regole attraverso l’analisi di grandi corpora (insiemi di documenti) provenienti dal mondo reale.
I sistemi basati su algoritmi di apprendimento automatico presentano diversi vantaggi rispetto ai tradizionali sistemi basati su regole:

- Le procedure di machine learning si focalizzano automaticamente sui casi più comuni, mentre nel contesto delle regole manuali non è sempre chiara la direzione nel quale lo sforzo debba essere diretto.

- Le procedure di apprendimento automatico fanno uso di algoritmi di deduzione statistica al fine di produrre modelli robusti anche in presenza di input non familiari (frasi o parole mai viste in precedenza) o addirittura scorretti (errori grammaticali o di battitura, omissione o abbreviazione di parole).
  In genere la gestione manuale di questo tipo di casistiche è estremamente difficile, soggetta ad errori e dispendiosa in fatto di tempo.

- I sistemi basati sull'apprendimento automatico possono essere migliorati nell'accuratezza semplicemente fornendo quantità via via crescenti di dati in input man mano che essi vengono raccolti. Ciò non vale invece per i sistemi a regole, per i quali difficilmente la complessità delle regole può essere aumentata a dismisura: esiste infatti un limite oltre il quale la complessità delle regole diventerebbe ingestibile.

Capitolo 3

Tecniche di preprocessing

Come descritto in precedenza, i documenti testuali contengono la maggior parte delle informazioni generate manualmente dagli esseri umani, ma per rendere i testi processabili agevolmente dai vari algoritmi di apprendimento automatico è necessario trasformarli dal formato non strutturato in cui si trovano ad un formato più strutturato ed adatto ad essere processato da un calcolatore.
Per completare questa conversione bisogna quindi eseguire una serie di operazioni sui testi, in assenza delle quali si potrebbero ottenere risultati altamente insoddisfacenti.

3.1 Selezione dei dati

Come fase preliminare alla manipolazione e al processing dei dati è necessario svolgere uno studio approfondito e un'analisi dei dati in questione, al fine di recuperare importanti informazioni quali la provenienza, la struttura, il contesto e l'eterogeneità degli stessi.
In particolare nei vari casi in analisi è importante riconoscere alcune caratteristiche:

- Il *contesto* considerato e l'ambito trattato, a seconda del quale varierà il vocabolario utilizzato ad esempio per quanto riguarda l'uso di termini tecnici.

- La *lingua* utilizzata, poiché eventuali regole o vocabolari usati dovranno riferirsi ad una specifica lingua.

- La *struttura* e l'eterogeneità dei testi, i quali potrebbero ad esempio essere fortemente caratterizzati dalla presenza di abbreviazioni e/o errori di battitura o presentare una struttura comune nell'uso di delimitatori e pattern semantici.

- La presenza di *outliers* (testi vuoti, estremamente corti o estremamente lunghi) che devono essere filtrati e gestiti a parte per non "inquinare" il modello.

3.2 Suddivisione del testo

La prima vera fase di manipolazione dei testi consiste nello spezzare il documento in parti più corte e più semplici da processare.
La tecnica maggiormente utilizzata per svolgere questa operazione è la cosiddetta tokenization (tokenizzazione) il cui scopo è suddividere il testo contenuto in un documento, appunto, in token: a seconda dell'applicazione considerata essi possono essere singole parole, intere frasi o parti di esse.

Esistono diversi modi per eseguire la tokenizzazione, la tecnica più semplice ed intuitiva ma spesso anche quella più efficace consiste nello spezzare il testo in base ai cosiddetti delimitatori. I delimitatori solitamente considerati sono quelli appartenenti alla cosiddetta punteggiatura "forte", ad esempio terminatori di riga *(\n)*, di frase *(.)* o di periodo *(, ; :)*.
Ovviamente è necessario tener conto e gestire eventuali eccezioni: ad esempio il carattere *" ."* usato come separatore di cifre decimali non è da considerarsi allo stesso modo di quando delimita la fine di una frase.
Per alcune specifiche funzionalità può anche essere utile spezzare il testo in singole parole; questo task è tuttavia più semplice poiché basta considerare alcuni caratteri speciali (apostrofo, spazio) che delimitano lo stacco tra due parole. [13]

3.3 Stopword filtering e lowercasing

Nel linguaggio umano esistono una serie di termini che non trasportano al proprio interno alcun significato semantico, ma servono solo come intercalare tra periodi.
Di questo insieme fanno parte ad esempio articoli, congiunzioni, preposizioni e tutti quei vocaboli che, dopo una scrupolosa analisi delle frequenze, risultano essere comuni.
Pertanto questi sono solitamente rimossi dal testo durante la fase di preprocessing in modo da diminuire la quantità di parole e di risorse computazionali richieste e al contempo aumentare la qualità dei dati, ponendo in maggior risalto parole che trasportano una maggiore quantità di informazione.
In aggiunta a questa operazione viene effettuata una conversione in minuscolo di tutti i caratteri che compongono il testo, per semplificare gli algoritmi e fare in modo che parole uguali vengano riconosciute come tali indipendentemente dai caratteri utilizzati per rappresentarle. [19]

3.4 Stemming

Lo stemming è il processo di riduzione della forma flessa di una parola alla sua forma radice, detta tema. Il tema non corrisponde necessariamente alla radice morfologica della parola: di solito è sufficiente che parole correlate siano mappate sullo stesso tema (ad esempio mang- per mangiare, mangiai, mangiò), indipendentemente dal fatto che esso corrisponda o meno ad una radice valida della parola in questione.

Lo stemming permette di impedire che declinazioni diverse dello stesso verbo o sostantivo vengano interpretate come parole distinte dal modello, evitando quindi di perdere l'associazione semantica tra singolari-plurali, maschili-femminili e diversi tempi verbali. Inoltre garantisce un'ulteriore riduzione della cardinalità del numero di parole distinte presenti nel testo, agevolando la computabilità e l'interpretabilità del modello.

La creazione di un algoritmo di stemming è stato per anni uno dei problemi più complicati dell'informatica, in quanto intrinsecamente dipendente dal contesto di analisi e da elementi come ad esempio la lingua utilizzata poiché ognuno di questi casi possiede le proprie peculiarità, eccezioni ed irregolarità specifiche.
Al momento in letteratura esistono diversi algoritmi di stemming, che differiscono tra loro per complessità ed accuratezza. Alcuni tra i più importanti esempi di algoritmi di stemming sono:

- *Lookup table*: la più semplice implementazione per uno stemmer, consiste nel mantenere una tabella di associazioni parola-radice ed eseguire di volta in volta una ricerca della parola considerata.
  Questa è la tecnica più semplice e veloce da implementare e permette di gestire qualsiasi tipo di eccezione, ma presenta lo svantaggio di dover mantenere in memoria una tabella molto grande con un'entry per ogni forma flessa di ogni singola parola. Non permette inoltre di gestire il caso di parole sconosciute (non contenute nella tabella).

- *Suffix stripping algorithm*: questa implementazione non si basa su una tabella ma piuttosto su una lista di regole che forniscono un determinato path da seguire a seconda della parola in input per ottenere il corretto risultato in output, ad esempio rimuovere i suffissi -ato/-ito/-uto e così via.
  Il vantaggio di questa tecnica è la semplicità della struttura dati impiegata, mentre lo svantaggio è la poca flessibilità nel gestire casi particolari ed eccezioni.

- *Lemmatizzazione*: questa tecnica richiede di determinare in quale parte del

testo appare una determinata parola, ed applicare diverse regole di normalizzazione a seconda del contesto.
Il contesto deve essere identificato prima di applicare le tradizionali regole di normalizzazione poiché in alcuni linguaggi esse variano a seconda della parte del discorso nella quale è inserita la parola.
Il vantaggio è la possibilità di poter applicare regole di normalizzazione più accurate grazie alla conoscenza del contesto, lo svantaggio è la difficoltà nel riuscire a generare una procedura in grado di identificare correttamente le varie parti del testo.

- Gli *approcci ibridi* combinano due o più degli approcci citati. Un semplice esempio di approccio ibrido è un algoritmo che combina suffix stripping e lookup table. La tabella è mantenuta piccola e serve a gestire i casi particolari, mentre per tutti gli altri casi viene usata una strategia basata su regole.

3.5 Modello Bag of Words

Il modello Bag of Words è una rappresentazione di dati di tipo testuale in forma numerica, in modo che essi possano essere forniti a modelli di apprendimento automatico i quali notoriamente funzionano quando applicati a dati numerici (in particolare vettori di numeri).
In questo senso il modello BoW è di facile comprensione e fornisce ottimi risultati negli ambiti del language modeling e document classification.

«Nel language processing, i vettori X sono derivati da dati di tipo testuale in una maniera che rifletta varie proprietà linguistiche del testo» [8]

Bag of Words è una rappresentazione che descrive le occorrenze delle parole all'interno di un documento e coinvolge due elementi:

1. Un dizionario di parole conosciute
2. Una misura della presenza di ognuna delle varie parole conosciute

Il modello è definito come "borsa di parole" proprio perché in esso non viene conservata alcuna informazione circa l'ordine o la struttura delle varie parole all'interno del documento.

«Una procedura molto comune per l'estrazione delle feature da frasi e documenti testuali è il modello Bag of Words (BoW). Tramite questo approccio si guarda all'istogramma delle parole all'interno del testo, ad esempio considerando la somma delle occorrenze di ogni parola come feature» [8]

Nella sua forma più basilare quindi, il modello BoW non fa altro che associare ad ogni parola il numero di occorrenze nel documento.

3.6 TF-IDF

Come sostenuto in precedenza il modello BoW ci permette di mappare testi su vettori numerici, con l'obiettivo di fornire questi ultimi ai vari algoritmi di apprendimento automatico.
Tuttavia se il nostro obiettivo è quello di enfatizzare parole chiave o particolarmente significative, esiste un'ulteriore trasformazione che possiamo applicare.
TF-IDF è una sigla che sta per "term frequency – inverse document frequency", ovvero il rapporto tra la frequenza locale di una parola rispetto alla frequenza globale della stessa.
Tramite il TF-IDF alle varie parole viene assegnato un peso, così TF-IDF diventa un modo per misurare la rilevanza delle parole piuttosto che la loro semplice frequenza.

- La *prima componente* è la "term frequency" (TF), cioè sostanzialmente l'output del modello BoW. Per uno specifico documento essa misura il numero di occorrenze dei vari termini. Più volte una parola appare all'interno di un documento più importante essa sarà in relazione al documento considerato.
- La *seconda componente* è la "inverse document frequency" (IDF). Essa misura quanto diffuso sia l'uso di un termine nell'arco di tutti i vari documenti. Infatti una parola per essere considerata distintiva oltre che essere frequente nel documento considerato deve apparire di rado in tutti gli altri.

La componente IDF è solitamente calcolata come:

$$IDF(W) = \log \frac{\#documents}{\#documents\ containing\ word\ W} \tag{3.1}$$

La TF-IDF non è altro che il prodotto di queste due quantità. Una parola per avere un alto valore di TF-IDF in un determinato documento deve apparire molte volte nello stesso e raramente negli altri. Deve quindi essere un *elemento distintivo* di quel documento.
Nonostante sia semplice ed intuitivo, TF-IDF è uno strumento estremamente potente ed è utilizzato per sviluppare funzionalità diffuse su scala mondiale come ad esempio Google Search.

Capitolo 4

Stato dell'arte

Vengono di seguito riportati i principali algoritmi presenti in letteratura come soluzione ai problemi di **classificazione** e **topic modeling**.
I seguenti algoritmi sono stati presi in considerazione come soluzione al problema affrontato in questa tesi. Alcuni di questi necessitano di ulteriori dati per poter essere utilizzati e vengono quindi contestualizzati come possibili estensioni future, altri invece sono stati fonte di spunto per lo sviluppo delle tecniche utili alla risoluzione del caso di studio, le quali verranno spiegate nel *sesto capitolo*.

4.1 Classificazione

4.1.1 Naive-Bayes

Il classificatore Bayesiano deriva direttamente dal celebre teorema di Bayes in ambito probabilistico, e modella la distribuzione dei documenti in ogni classe usando un modello probabilistico basato su assunzioni di indipendenza tra le distribuzioni delle varie parole.
L'idea generale del classificatore Bayesiano consiste nel basarsi su una rappresentazione numerica dei documenti (tipicamente BoW), e di calcolare per ogni classe di riferimento la probabilità a posteriori $P(X|Y = y)$ basata sulla distribuzione delle parole nel documento. In fase di classificazione questo modello verrà sfruttato per assegnare in output l'etichetta con il più alto valore di probabilità. [1]

$$\hat{y} = \arg\max_{y} P(X|y) * P(y) \tag{4.1}$$

Il modello viene costruito assumendo che l'ordine delle parole non sia importante ai fini del significato della frase, ma solamente il numero di occorrenze lo sia, e che le parole non abbiano relazioni di dipendenza tra loro.
Malgrado le assunzioni non siano totalmente verificate il modello riesce comunque ad ottenere ottime prestazioni nonostante la sua semplicità.

4.1.2 SVM

L'algoritmo SVM nasce come soluzione al problema della classificazione binaria, e mira a trovare un iperpiano che divida i dati in due gruppi.
L'iperpiano è un separatore lineare per qualsiasi dimensione, esso può essere una linea (2D), un piano (3D) o un iperpiano (4D+).
I dati possono essere separati da un numero infinito di iperpiani, per questo motivo l'obiettivo è scegliere quello che massimizzi il margine.
Il margine è la distanza tra l'iperpiano e i punti più vicini ad esso: questi punti sono chiamati support vectors poiché controllano l'iperpiano. Questo è il cosiddetto *Large Margin Classifier*.

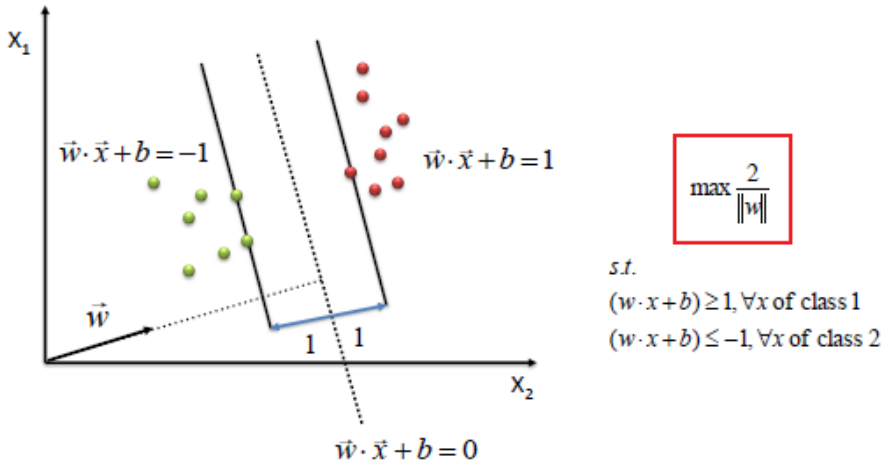

Figura 4.1. Spiegazione grafica di SVM

Nei problemi reali però non sempre i dati sono linearmente separabili, per questo motivo il *Large Margin Classifier* viene spesso sostituito dal più moderno *Soft Margin Classifier*: l'obiettivo è sempre quello di massimizzare il margine, ma in questo caso viene aggiunta la possibilità di tollerare un'errata classificazione di alcuni punti. La soglia di tolleranza è pilotata da un parametro chiamato C, più grande scegliamo C più alta sarà la tolleranza nei confronti degli errori di classificazione. Il problema da risolvere per trovare il classificatore è il seguente.

Primal optimization problem:

$$minimize\ \frac{1}{2}||w||^2 + C\sum_i \xi_i \tag{4.2}$$

$$\text{subject to } y_i[\langle w, x_i\rangle + b] \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \tag{4.3}$$

Dual optimization problem:

$$maximize \ -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_i \alpha_i \tag{4.4}$$

$$\text{subject to } \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C] \tag{4.5}$$

Per superare il vincolo della separabilità lineare utilizziamo nel nostro modello un kernel di tipo RBF.

$$K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right) \tag{4.6}$$

Infine, per estendere la classificazione oltre il caso binario, viene usato un approccio del tipo One-vs-Rest.
Questa strategia consiste nel creare per ogni classe un singolo classificatore, che produrrà in output un valore di confidenza. La classe che avrà prodotto il più alto valore di confidenza verrà scelta come etichetta.

E' stato provato che il classificatore SVM si adatta molto bene alla classificazione testuale, grazie alla natura altamente multi-dimensionale e sparsa dei testi. Inoltre il classificatore SVM è semplice e facilmente interpretabile.
Esso inoltre ha dimostrato di essere fortemente efficace anche nei problemi di apprendimento semi-supervisionato, con grandi quantità di dati non etichettati e solamente una piccola quantità di dati etichettati.

4.1.3 KNN

L'algoritmo si basa sulla feature similarity, e per ogni punto da classificare valuta a quale delle N classi esso sia più "vicino": quando si vuole classificare un dato con il metodo del KNN, l'algoritmo va ad assegnargli l'etichetta più frequente tra i K training samples più vicini a quel dato punto.

Esso è un algoritmo "pigro" poiché tutto ciò che fa nella fase di training è memorizzare i vettori di input con la rispettiva classe di appartenenza, mentre la computazione è rimandata alla fase di classificazione.
Per questa ragione l'algoritmo è molto veloce nella fase di training ma relativamente lento nella classificazione.

Una metrica diffusa per calcolare la distanza tra i punti è la distanza euclidea.

$$d = \sqrt{\sum_{k=1}^{n}(p_k - q_k)^2} \tag{4.7}$$

Infine possono essere fatte delle modifiche nel conteggio dei record più vicini, al fine di migliorare le prestazioni. Ad esempio si può assegnare un peso al contributo di ogni campione, in maniera che i più vicini contribuiscano in maniera maggiore al risultato finale (ad esempio un peso di 1/d, dove d è la distanza dal punto).
Un parametro di configurazione importante è dunque il valore di K, poiché esso influenza sia la precisione dell'algoritmo che la sua velocità: un valore basso di K renderà l'algoritmo veloce ma sensibile alla presenza di outliers, mentre al contrario un K maggiore renderà l'algoritmo più robusto ma estremamente lento.

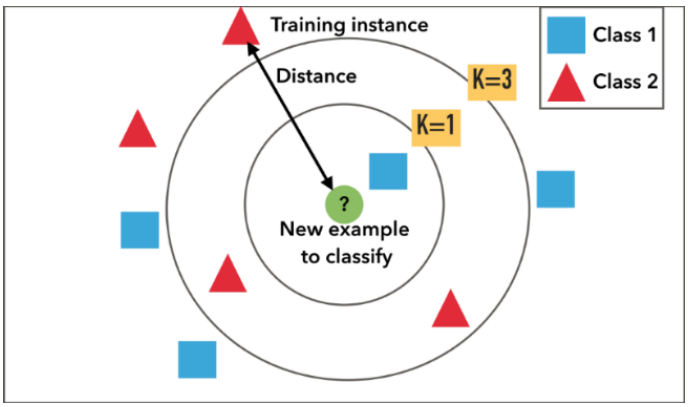

Figura 4.2. Spiegazione grafica di KNN

4.1.4 Decision tree

Un albero di decisione è un modello predittivo dove ogni nodo interno rappresenta una variabile, un arco verso un nodo figlio rappresenta un possibile valore per quella proprietà e una foglia il valore predetto partendo dai valori delle altre proprietà. Quindi classificare un record significa effettuare una serie di test, iniziando dal nodo radice e propagando via via il campione fino a terminare con un nodo foglia: questa sarà l'etichetta assegnata a quello specifico record.
Durante la creazione del modello, per decidere di volta in volta quale variabile considerare come criterio di split, l'algoritmo calcola quella che se usata come split massimizza la purezza dei dati e la sceglie come split.
I parametri più largamente usati per le condizioni di split sono:

- **Gini index**: raggiunge il suo minimo (zero) quando il nodo appartiene ad una singola categoria.
$$G = 1 - \sum_{j=1}^{m} f(i,j)^2 \tag{4.8}$$

- **Variazione di entropia**: basata sul concetto di entropia.
$$E = -\sum_{j=1}^{m} f(i,j) \log f(i,j) \tag{4.9}$$

In molte situazioni è utile definire un criterio di arresto (halting), o anche criterio di potatura (pruning) al fine di determinarne la profondità massima. Questo perché il crescere della profondità di un albero (ovvero della sua dimensione) non influisce direttamente sulla bontà del modello. Infatti, una crescita eccessiva della dimensione dell'albero potrebbe portare solo ad aumento sproporzionato della complessità computazionale rispetto ai benefici riguardanti l'accuratezza delle classificazioni ed indurre ad overfitting.

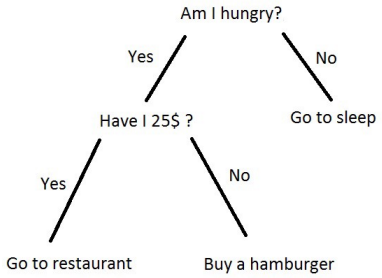

Figura 4.3. Semplice esempio di decision tree

4.1.5 Random forest

La foresta casuale è un metodo di classificazione che opera costituendo una moltitudine di alberi di decisione durante il training e producendo come output in fase di test l'etichetta più di frequente emessa dai singoli alberi.
Le foreste casuali correggono la tendenza degli alberi di decisione ad avere overfitting sul training set.
L'algoritmo di training per le foreste casuali applica la regola generale del bagging agli alberi di decisione. Dato il training set vengono selezionati casualmente N campioni e su di essi vengono allenati gli alberi di decisione.
Inoltre durante il processo di apprendimento, ad ogni split viene selezionato un sottoinsieme casuale di features da considerare come criterio di divisione (feature bagging). In questo modo si evita che gli stessi predittori vengano usati da tutti gli alberi della foresta rendendoli fortemente correlati tra loro. Tipicamente in un problema di classificazione con p features per ogni split ne vengono selezionate casualmente $\sqrt{p}$.
Questa procedura conduce a delle migliori performance poiché diminuisce la varianza del modello. Infatti mentre i singoli alberi sono altamente sensibili al rumore e tendono a avere overfitting, la media di questi ultimi non lo è se gli alberi non sono tra loro correlati (il bootstrap ci permette di de-correlare tra loro i modelli mostrando di volta in volta training set diversi).
Inoltre può essere fatta una stima dell'incertezza della predizione come deviazione standard delle predizioni di tutti i singoli alberi:

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}} \tag{4.10}$$

4.2 Topic modeling

Nel Natural Language Processing un topic model è un modello statistico il cui obiettivo è trovare i topics (o tematiche) astratti contenuti in un insieme di documenti. I topic non sono noti a priori, ma vengono identificati autonomamente dall'algoritmo in base alla frequenza e al numero di occorrenze delle parole nei vari testi. Ad esempio le parole "cane" ed "osso" compariranno molto di frequente nei testi relativi agli animali mentre parole come "parlamento" e "legislatura" saranno univocamente associate a testi relativi alla politica.
Sfruttando statistiche di questo tipo l'algoritmo sarà quindi in grado di individuare un numero arbitrario di tematiche generali (i cosiddetti topics) presenti nei vari testi ed assegnare correttamente ad ogni testo il suo rispettivo topic semantico.

4.2.1 Latent Semantic Analysis

LSA, conosciuta in alcuni contesti anche come Latent Semantic Indexing (LSI), è una tecnica utilizzata nel NLP per analizzare dei documenti al fine di trovarne una rappresentazione basata sulla co-occorrenza delle parole.

Se ogni parola fosse usata esclusivamente all'interno di un solo argomento sarebbe semplice trovare un mapping tra parole e concetti.

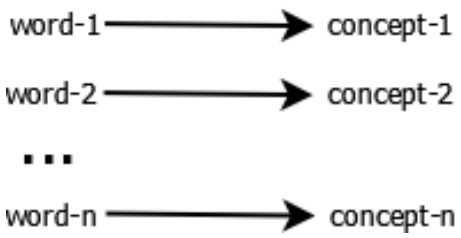

Figura 4.4. Idea di partenza di LSA

Tuttavia nel linguaggio naturale ogni parola è utilizzata di volta in volta all'interno di contesti differenti. Per questa ragione la soluzione al problema non è semplice.

LSA assume che le parole di significato simile siano usate in pezzi di testi simili tra loro.

La matrice di partenza è una rappresentazione di tipo TF-IDF della distribuzione delle parole all'interno dei vari documenti.

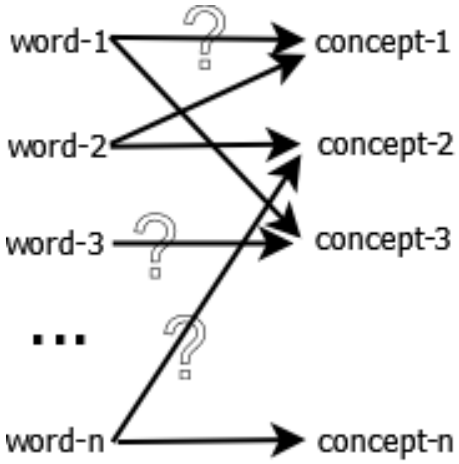

Figura 4.5. LSA nei casi reali

A questa viene conseguentemente applicata una tecnica matematica chiamata Singular Value Decomposition (SVD), che permette di ridurre la dimensionalità dei dati. Dietro a questo passaggio ci sono diverse ragioni:

- La matrice originaria si presume di dimensione troppo grande per poter essere affrontata computazionalmente. In questo senso la dimensionality reduction è vista come una contromisura necessaria.

- Ridurre la dimensionalità permette anche di rimuovere parte del rumore intrinsecamente presente nella matrice originale.

- La matrice di partenza è sparsa, quindi è possibile ridurre notevolemente la cardinalità e al contempo preservare gran parte dell'informazione.

L'applicazione di SVD al recupero delle informazioni è chiamata Latent Semantic Indexing poiché le rappresentazioni dei documenti nello spazio originale vengono trasformate in rappresentazioni in un nuovo spazio ridotto.
L'obiettivo è fare in modo che queste ultime siano rappresentazioni migliori dei documenti in termini di compattezza e contenuti (ovvero che contengano le stesse informazioni di partenza ma con una rappresentazione numerica più compatta e pulita rispetto a cardinalità e rumore). [13]
Il nuovo spazio dimensionale può quindi essere usato efficacemente per misurare la somiglianza tra i testi utilizzando la cosine similarity, ossia calcolando il prodotto scalare (coseno) tra i vettori. Più alto sarà il valore calcolato più i testi saranno "simili" tra loro.

4.2.2 Latent Dirichlet Allocation

LDA è un modello statistico che permette di identificare un numero predefinito di topic ed attribuire ad ogni documento una distribuzione statistica lungo i vari topic. Il modello considera ogni documento come una distribuzione di topic e ogni topic come una distribuzione di parole.

A differenza di LSA la distribuzione dei topic è basata su una distribuzione a priori chiamata Dirichlet prior. Essa è basata sull'idea che ogni documento ricopra solo un insieme limitato di topic e che a loro volta i topic utilizzino un limitato sottoinsieme di parole.

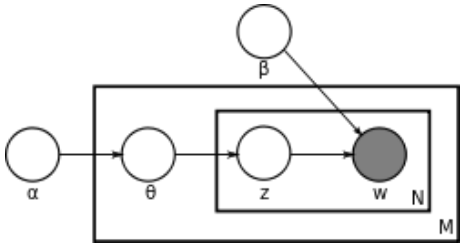

Figura 4.6. Rappresentazione grafica del modello LDA.

Ad esempio, in presenza di due topic classificabili rispettivamente come relativi a "gatto" e "cane", l'algoritmo genererà un insieme di parole che rappresentano ognuno dei due topic (es. latte e miagolare per il topic "gatto", osso e abbaiare per il topic "cane"), poi assocerà ad ogni parola chiave un peso (maggiore è il valore del peso più la keyword sarà discriminante un determinato topic) e infine userà il modello così costruito per assegnare ad ogni documento una distribuzione probabilistica lungo i due topic.
In questo modo documenti appartenenti ad una specifica classe avranno un alto valore di probabilità all'interno del rispettivo topic, mentre documenti non appartenenti a nessuno dei due topic avranno distribuzioni pressoché uguali.

Capitolo 5

Strumenti utilizzati

5.1 Tools

- *Apache SolR*: è un motore di ricerca altamente scalabile utilizzato per effettuare ricerca su dati testuali di grandi dimensioni.

- *Dbeaver*: è un client SQL e uno strumento di database administration multipiattaforma e open source. Fornisce supporti all'utente come un editor intelligente e funzioni di highlighting automatico oltre ad una serie di plugin per interfacciarsi con i diversi database.

- *Vertica*: Database management system per basi dati relazionali e column-oriented.

- *PyCharm*: Integrated Development Environment (IDE), cioè un ambiente di sviluppo integrato multipiattaforma specifico per il linguaggio Python. Fornisce strumenti avanzati come un debugger grafico e l'integrazione con sistemi di version control (VCS).

5.2 Software e linguaggi di programmazione

- *Python*: linguaggio di programmazione ad alto livello interpretato orientato agli oggetti. Possiede una grande quantità di avanzate librerie, le quali costituiscono il suo punto di forza, specialmente per quanto riguarda l'ambito del machine learning nel quale è per distacco il linguaggio più utilizzato.

- *SQL*: Structured Query Language, usato per interrogare il database.

- Librerie *math*, *pandas*, *matplotlib*: librerie Python che offrono una serie di API utili ad eseguire efficacemente funzioni matematiche complesse e a disegnare grafici.
- Libreria *gensim*: libreria contenente i modelli di machine learning utilizzati.
- Libreria *NLTK*: contiene tutte le funzioni necessarie alla manipolazione e al preprocessing dei testi.
- *Snowball Stemmer*: il software utilizzato per eseguire lo stemming delle parole. Esso fornisce un'implementazione ad hoc dello stemmer per 13 diversi linguaggi, compreso l'italiano.
  In particolare l'implementazione dello stemmer italiano combina un approccio di tipo suffix-stripping con delle lookup tables.
- *Cisco VPN*: utilizzata per collegarsi da remoto ai database contenenti i dati medici.

5.3 Hardware

Il dispositivo hardware adottato: Intel Core i7-8550U Quad-Core: 2 GHz, 1.80 GHz con 8GB di RAM e sistema operativo Windows 10 x64.

Parte II

Seconda Parte

Capitolo 6

Caso di studio

6.1 Descrizione problema

Il caso di studio in oggetto prende in considerazione dati provenienti dal settore medico, messi a disposizione in una banca dati condivisa.
I testi considerati sono redatti manualmente dai vari medici specialistici in seguito a visite o dimissioni di pazienti e contengono tutte le informazioni che normalmente vengono refertate in seguito ad eventi di questo tipo.
L'obiettivo è quindi utilizzare l'informazione contenuta nel campo di testo per effettuare alcune analisi:

- Identificare la presenza di raccomandazioni, consigli, indicazioni di approfondimenti diagnostici.
- Identificare la presenza di descrizioni di particolari condizioni psico-fisiche e cliniche dei pazienti.
- Identificare la presenza di prescrizioni o consigli di terapie farmacologiche.
- In relazione a quest'ultima categoria identificare se la terapia farmacologica è citata in anamnesi o in dimissione e riportare (ove presente) la posologia o il dosaggio.

Nella base di dati fornita i dati sono organizzati in due tabelle, ognuna delle quali contiene un campo di testo libero oltre ai dati di tipo strutturato come ad esempio l'anagrafica del paziente, l'identificativo del medico specialista e la data di refertazione.
Per capire come affrontare il problema è opportuno svolgere un'analisi preliminare sulla struttura dei dati considerati.

6.1.1 Tabella DEMA_SPECIALISTICA

PRESCRIZ_CODICE	PRESCRIZ_DATA	PRESCRIZ_ESENZIONE	PRESCRIZ_CLASSIFICAZIONE	PRESCRIZ_DESC_QUESITO_DIAGN
104Z28JC09PRB2CQ	2017-06-23	E01	DEMATERIALIZZATA	IPERTENSIONE ARTERIOSA E DISLIPIDEMIA
104L5QSSW9PRB2DW	2017-06-23	E01	DEMATERIALIZZATA	IPERTENSIONE ARTERIOSA E DISLIPIDEMIA
104HGTPBV9PRB2DB	2017-06-23	E01	DEMATERIALIZZATA	per controllo
104AFSZR09PRB2EB	2017-06-23	E01	DEMATERIALIZZATA	per controllo
904CZKB6K9PRCWTX	2017-06-23	016	DEMATERIALIZZATA	Follow-up epatite B
104A07X2S9N0JKC3	2017-03-24	RE1	DEMATERIALIZZATA	DISPEPSIA- ASTENIA- PREVENZIONE;
104VAHJEG9PRB2H1	2017-06-23	E01	DEMATERIALIZZATA	AMNESIA PERSISTENTE DOPO EPISODIO ISCHEMICO CEREBRALE
104VAEYEX9PRB2JM	2017-06-23	RE2	DEMATERIALIZZATA	sintomatologia da cistite
104A3UQ0B9PRB2PD	2017-06-23	RE1	DEMATERIALIZZATA	DISLIPIDEMIA
904G40C9R9PRCX2D	2017-06-23	048	DEMATERIALIZZATA	quadro emogas analitico e chimico suggestivo per embolia polmonare
104P381541C5E6DU	2017-06-23	RE1	DEMATERIALIZZATA	valutazione in cardiopatia ipert. e pregresso IMA con stent
104AVCM4N9N0JKC3	2017-03-24	RE1	DEMATERIALIZZATA	DISPEPSIA- ASTENIA- PREVENZIONE;
104PSQ52W9PRB32D	2017-06-23	E01	DEMATERIALIZZATA	nevo congenito dorso piede sx si richiede videomicroscopia annuale
104B1MNWV9N0JKGV	2017-03-24	E01	DEMATERIALIZZATA	CATARATTA
104ACK1Y69PRB3AN	2017-06-23	013	DEMATERIALIZZATA	per follow-up di diabete mellito
104AMBJQA9PRB3G1	2017-06-23	RE1	DEMATERIALIZZATA	reumatismo palindromico.
104P6DP379PRB3PB	2017-06-23	E01	DEMATERIALIZZATA	parestesie arto sup sn
104HAF3TA9PRB3WG	2017-06-23	RE2	DEMATERIALIZZATA	nevo irregolare arto inf. ds
104NRKDJR9PRB3QS	2017-06-23	RE2	DEMATERIALIZZATA	GONALGIA SX CON VERSAMENTO
104SMMACS9PRBDLP	2017-06-23	RE1	ROSSA	VALUTAZOINE ACCERTAMENTI PROVEDIMENTI - SOSPETTO: TROMBO
104LLV53E9PRB42D	2017-06-23	E01	DEMATERIALIZZATA	follow up vasculopatia in ipertesa CARDIRENE
104Q8Z5B09PRB415	2017-06-23	RE1	DEMATERIALIZZATA	astenia -

Figura 6.1. Tabella DEMA_SPECIALISTICA

La figura 6.1 riporta un esempio di contenuto per la prima delle due tabelle da considerare:

- Contiene 7.3 milioni di record (ogni record corrisponde ad un episodio).
- Contiene episodi provenienti da 650 mila pazienti diversi.
- I testi in essa contenuti sono estremamente corti: 3-4 parole in media contenenti solamente un breve riferimento alla problematica in questione.
- Solo 130 mila record (1.5% del totale) hanno una lunghezza significativa (>100 caratteri) e contengono informazioni più dettagliate.

La tabella viene aggiornata al termine di ogni consulenza medica, quando si ritiene necessaria l'emissione di un'impegnativa per visite specialistiche. Alcuni esempi di testi in questione sono:

"Accertamenti"
"Per follow up"
"Dolore polso sx"

I campi strutturati contengono informazioni relative a:

- Un identificatore univoco per ogni evento.
- Data di emissione dell'impegnativa.
- Diritto ad eventuale esenzione.
- Codice di priorità dell'intervento.

Per questi motivi la tabella in questione non è stata considerata nelle analisi effettuate vista la ridotta quantità di informazioni in essa contenute.

6.1.2 Tabella PFA

CODICE_DOC	CODICE_EP	DESCRIZIONE_UO	DATA_	TIPOLOGIA	TESTO
12877635	13933777	Ca medicina riabilitativa	2017-12-11	Referto Ambulatoriale	DIAGNOSI: Cervicalgia ==== DIAGNOSI: Dor
12950286	14011702	Ba medicina riabilitativa	2018-01-04	Referto Ambulatoriale	
12470881	13505448	Pa medicina interna	2017-08-28	Referto Ambulatoriale	Parametri base ==== ====Mitrale ==== Morf
13223257	14300973	Ba endocrinologia	2018-03-13	Referto Ambulatoriale	Ambulatorio delle Malattie Metaboliche dell'Oss
12051649	13067678	Ca otorinolaringoiatria	2017-04-27	Referto Ambulatoriale	Esiti stabilizzati di intervento per otosclerosi dx.
12938981	13999080	Mi cardiologia	2017-12-28	Referto Ambulatoriale	Anamnesi:Pz di anni 32. Da circa due settimane
13521390	14610516	Sa urologia	2018-05-29	Lettera Dimissione	Diagnosi: Idrocele sinistro. pregressa cvaricocele
12403435	13433530	Ba cardiologia	2017-07-31	Referto Ambulatoriale	al controllo attuale limitatamente alle pessime fi
13472047	14565372	D4 SA spec.amb. distrettuale	2018-05-17	Referto Ambulatoriale	========estrazione di 38 inh anestesia locale
13594391	14695693	D6 specialistica amb. distrettuale Vignola	2018-06-15	Referto Ambulatoriale	Fibrolaringoscopia: rinofaringe libero, test di Mu
13160853	14233650	D3 centro prelievi	2018-02-26	Referto Ambulatoriale	Rivalutazione terapia anticoagulante in paziente
14157774	15301733	Ba neurochirurgia	2018-11-21	Referto Ambulatoriale	Lavora come badante di una persona anziana in
13536142	14633340	Pp diagnosi e cura	2018-06-01	Referto Ambulatoriale	Paziente visitato ieri in consulenza per stato di ir
12023357	13016491	Pa medicina interna	2017-04-18	Lettera Dimissione	Diagnosi di dimissione: Insufficienza renale acut
12179292	13201554	Ca medicina indirizzo oncologico	2017-05-30	Referto Ambulatoriale	Egregio Collega, ====vedo per visita supportiv
13511090	14606745	Ca urologia	2018-05-26	Referto Ambulatoriale	Cv mal posizionato, sostituito Cv , effettuate lav
14267387	15416741	Sa chirurgia	2018-12-18	Referto Ambulatoriale	BCC superficiali del braccio sinistro dove si ripet
12315807	13343665	Sa medicina	2017-07-05	Referto Ambulatoriale	======In paziente sottoposta a tiroidectomia
12089533	13107608	Ba cardiologia	2017-05-08	Referto Ambulatoriale	Rinnovo piano terapeutico Xarelto 20mg per
11708491	12707505	D6 specialistica amb. distrettuale Vignola	2017-01-30	Referto Ambulatoriale	ECG : ritmo sinusale con fc media 69 b m===
13336474	14408832	Sa medicina	2018-04-11	Lettera Dimissione	In base alle risultanze clinico-laboratoristiche e c
11814963	12819229	D5 spec. ambul. distrettuale Pavullo	2017-02-24	Referto Ambulatoriale	diabete tipo 2 dal 2012 in terapia con ipoglicem

Figura 6.2. Tabella PFA

La seconda tabella:

- Contiene circa 2 milioni di record (ogni record corrisponde ad un episodio)
- Contiene informazioni riguardanti 400 mila pazienti, 1500 medici e 250 reparti.

- I record sono suddivisi in due diverse tipologie: Lettere di dimissione e Referti ambulatoriali.

I campi strutturati contengono informazioni relative a:

- Una coppia di chiavi univoche.
- Reparto ospedaliero di competenza.
- Data di emissione del referto.
- Tipologia del documento (lettera di dimissione o referto ambulatoriale).

Le **lettere di dimissione** rappresentano la minoranza (solo 120 mila tuple su 2 milioni) e presentano un pattern predefinito e ricco di parole chiave.

Alcuni esempi di estratti di testo sono:

"Diagnosi: Colecistite in paziente con colelitiasi e recente colica biliare."
"Si consiglia: riposo ed adeguata idratazione, ripresa dell'abituale terapia domiciliare."
"A domicilio: aerosolterapia ancora per qualche giorno."

Per determinare la presenza di prescrizioni, raccomandazioni o descrizioni cliniche all'interno di questa tipologia di testi la migliore soluzione è seguire un approccio di tipo dizionario in grado di identificare parole chiave come diagnosi, intervento, a domicilio, terapia, decorso, consigli e metterle in relazione con le varie categorie a cui appartengono.
Poiché le lettere di dimissione seguono un pattern standard che antepone una o più parole chiave ad ogni sezione del testo ogni testo potrà essere diviso in paragrafi ed ogni paragrafo catalogato correttamente in base alle parole chiave in esso contenute.

I **referti ambulatoriali** invece rappresentano la maggioranza dei record presenti nella base di dati (più di 1.8 milioni di tuple) e consistono in testi più eterogenei e meno strutturati. Essi presentano una lunghezza media di 600 caratteri e contengono la presenza di alcuni outliers (testi vuoti o troppo lunghi/corti).

Per avere un'idea più concreta della struttura dei dati stampiamo l'istogramma delle lunghezze dei testi.

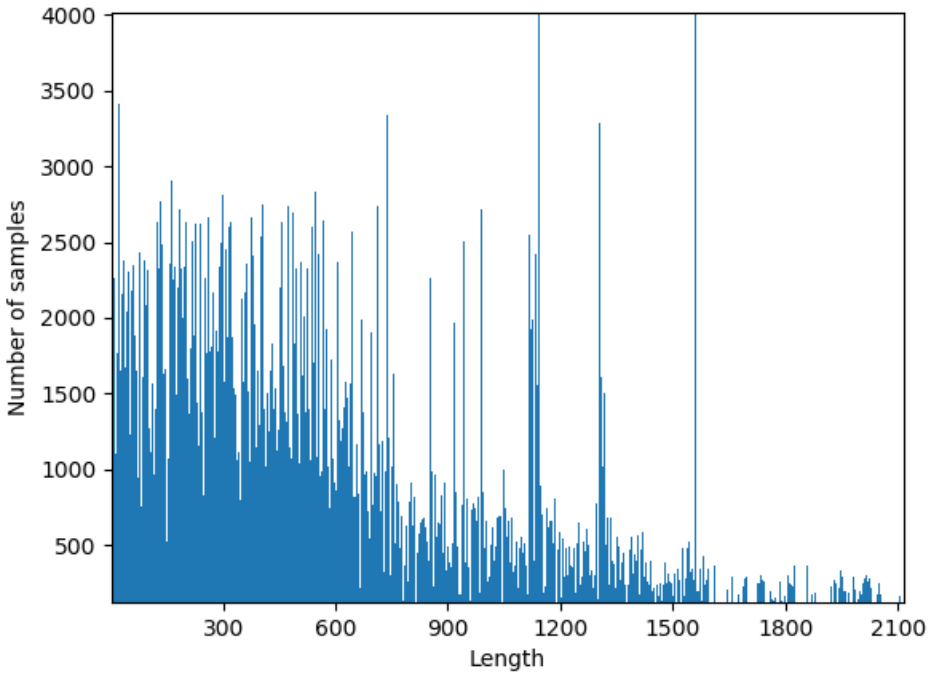

Figura 6.3. Distribuzione delle lunghezze dei testi

Alcuni esempi di estratti di testo sono:

"Paziente iperteso già sottoposto ad ernioplastica inguinale destra."
"Delecit 500 una al dì per 15 giorni"
"Iperten 20 mg 1 cp h 20"

Per quanto riguarda quest'ultima tipologia di testi (la grande maggioranza di quelli in nostro possesso) la soluzione proposta consiste nel creare un algoritmo capace di combinare un approccio di apprendimento automatico con una più tradizionale tecnica a dizionario.
L'obiettivo del lavoro è creare un modello capace di identificare autonomamente un insieme di "topic" e catalogare con un certo livello di confidenza le diverse frasi all'interno di questi.
Una volta identificati automaticamente i topic il modello fornisce per ognuno di questi una lista di parole che li contraddistingue; su queste parole viene eseguita la tecnica a dizionario al fine assegnare ad ogni topic un **significato semantico**.

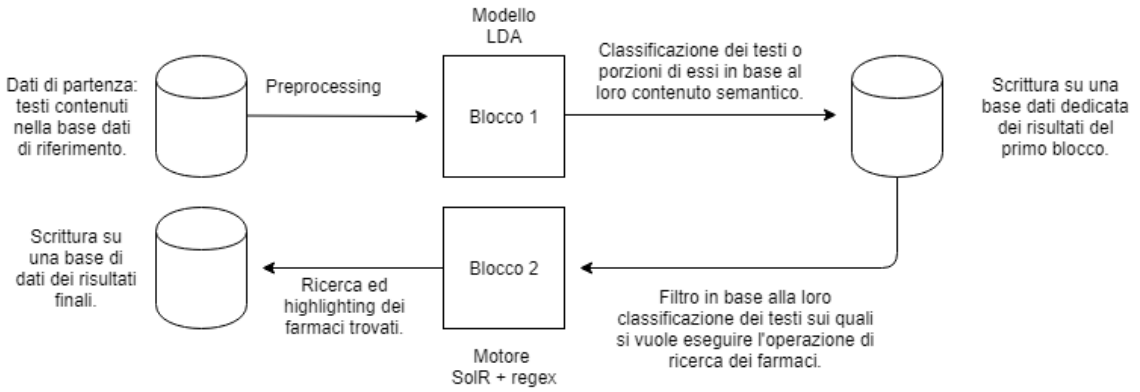

Figura 6.4. Diagramma di flusso.

6.2 Classificazione delle informazioni contenute nei testi ambulatoriali

6.2.1 Selezione dei dati

Interrogando la base di dati selezioniamo i testi appartenenti alla categoria da noi considerata (referti ambulatoriali) filtrando gli outliers (testi troppo lunghi/corti) che potrebbero inquinare il modello.

```
SELECT TESTO
FROM DWH.DC_L1_PFA
WHERE ( TIPOLOGIA = 'Referto Ambulatoriale'
                AND LENGTH(TESTO) > 100
                AND LENGTH(TESTO) < 2000 )
```

A scopo esplicativo vengono riportati dei campioni di testi come esempi di dati da processare:

1. *"Artropatia degenerativa rachide dorso lombare coxartrosi dx e gonartrosi. ========Buon risultato dal trattamento infiltrativo. ====Confermo l'indicazione data dal collega pochi giorni orsono. ========Consiglio:====-Palexia 50 mg 1 cp mattino e sera per 20 gg (fino a 4 cicli in un anno). ====-Fosavance 1 cp alla settimana."*

2. *"Riferita contusione polso dx.========EO: segni clinici di frattura, cute indenne, non defict vnp, gomito indenne.========RX: frattura dipo Barton (volare).========Si pratica riduzione e gesso ABM che appare ben tollerato.====RX in gesso.========Per il tipo di frattura indicata la riduzione e sintesi chirurgica con placca e viti.========Si mette in lista per intervento (priorità A).========Nel frattempo barccio al collo, mobilizzazione attiva delle dita della mano, se dolore Tachipirina 1000 1 cp (max 3 die).===="*

Come si può notare dagli esempi riportati i testi presentano una serie di peculiarità specifiche dei testi redatti in ambito medico/ospedaliero.
Tra queste caratteristiche evidenziamo ad esempio un vocabolario ristretto e fortemente specifico, un uso inconsueto di simboli e separatori non standard (ad esempio "="), errori di battitura ("*barccio*") ed una sintassi estremamente snella (periodi tendenzialmente brevi e concisi).
Come di consueto, per le ragioni in precedenza specificate, su questi dovremo effettuare le varie operazioni di pre-processing prima di iniziare ad addestrare il modello di apprendimento automatico.

6.2.2 Tokenizzazione

La prima operazione è la tokenizzazione: ogni testo viene pulito da separatori e caratteri speciali (ad esempio "=") e successivamente viene scomposto in frasi.
Lo strumento utilizzato per realizzare la tokenizzazione è il tokenizer contenuto nel pacchetto NLTK.

```
# tokenize sentences and remove special characters
sent_tokenized_list = nltk.tokenize.sent_tokenize(
        it_string.replace("\'"," ").replace("="," "))
```

Nel caso dei testi riportati in precedenza l'operazione di tokenizzazione produce in output i seguenti risultati:

1. *['Artropatia degenerativa rachide dorso lombare coxartrosi dx e gonartrosi .', 'Buon risultato dal trattamento infiltrativo.', 'Confermo l'indicazione data dal collega pochi giorni orsono. ', 'Consiglio:', '- Palexia 50 mg 1 cp mattino e sera per 20 gg (fino a 4 cicli in un anno) .', '- Fosavance 1 cp alla settimana.']*

2. *['Riferita contusione polso dx.', 'EO: segni clinici di frattura, cute indenne, non defict vnp, gomito indenne.', 'RX: frattura dipo Barton (volare).', 'Si pratica riduzione e gesso ABM che appare ben tollerato.', 'RX in gesso.', 'Per il tipo di frattura indicata la riduzione e sintesi chirurgica con placca e viti.', 'Si mette in lista per intervento (priorità A).', 'Nel frattempo barccio al collo, mobilizzazione attiva delle dita della mano, se dolore Tachipirina 1000 1 cp (max 3 die).']*

6.2.3 Lowercasing e rimozione della punteggiatura

A questo punto ad ogni token vengono applicate le operazioni di lowercasing e di rimozione della punteggiatura.
Utilizziamo il package *string* per effettuare agevolmente queste trasformazioni.

```
# lowercasing
sent_tokenized_no_punct = [
        x.lower() for x in sent_tokenized_list
        if x not in string.punctuation]

# remove punctuation
sent_tokenized_no_punct = [
        s.translate(str.maketrans('', '', string.punctuation))
        for s in sent_tokenized_no_punct]
```

Il risultato delle due operazioni ci restituisce token del seguente tipo:

1. *['artropatia degenerativa rachide dorso lombare coxartrosi dx e gonartrosi ', 'buon risultato dal trattamento infiltrativo ', 'confermo l indicazione data dal collega pochi giorni orsono ', 'consiglio', ' palexia 50 mg 1 cp mattino e sera*

per 20 gg fino a 4 cicli in un anno ', ' fosavance 1 cp alla settimana']

2. *['riferita contusione polso dx', 'eo segni clinici di frattura cute indenne non defict vnp gomito indenne', 'rx frattura dipo barton volare', 'si pratica riduzione e gesso abm che appare ben tollerato', 'rx in gesso', 'per il tipo di frattura indicata la riduzione e sintesi chirurgica con placca e viti', 'si mette in lista per intervento priorità a', 'nel frattempo barccio al collo mobilizzazione attiva delle dita della mano se dolore tachipirina 1000 1 cp max 3 die']*

6.2.4 Stopword filtering

A questo punto possiamo procedere con la rimozione delle cosiddette stopword.
A tale proposito il pacchetto NLTK ci fornisce per ogni linguaggio supportato una lista contenente tutte le stopword, quindi tutto ciò che dobbiamo fare è selezionare la lista di stopword relative all'italiano e filtrare queste ultime dai token originali.

```
# dictionary of Italian stop-words
it_stop_words = nltk.corpus.stopwords.words('italian')
```

Di seguito riporto a scopo esplicativo un estratto della lista di stopword restituita da *NLTK*:

it_stop_words = ['ad', 'al', 'allo', 'ai', 'agli', 'all', 'agl', 'alla', 'alle', 'con', 'col', 'coi', 'da', 'dal', 'dallo', 'dai', 'dagli', 'dall', 'dagl', 'dalla', 'dalle', 'di', 'del', 'dello', 'dei', 'degli', 'dell', 'degl', 'della', 'delle', 'in', 'nel', 'nello', 'nei', 'negli', 'nell', 'negl', 'nella', 'nelle', 'su', 'sul', 'sullo', 'sui', 'sugli', 'sull', 'sugl', 'sulla', 'sulle', 'per', 'tra', ...]

Con l'ausilio di questa lista procediamo quindi a filtrare le stopword dai vari token.

```
# create list of sentences without stopwords
sent_tokenized_no_punct_no_sw = []
for sent in sent_tokenized_no_punct:
    sent = ' '.join([word for word in sent.split()
                if word not in it_stop_words])
    sent_tokenized_no_punct_no_sw.append(sent)
```

Giungeremo ad una situazione di questo tipo:

1. *['artropatia degenerativa rachide dorso lombare coxartrosi dx gonartrosi', 'buon risultato trattamento infiltrativo', 'confermo indicazione data collega pochi giorni orsono', 'consiglio', 'palexia 50 mg 1 cp mattino sera 20 gg fino 4 cicli anno', 'fosavance 1 cp settimana']*

2. *['riferita contusione polso dx', 'eo segni clinici frattura cute indenne defict vnp gomito indenne', 'rx frattura dipo barton volare', 'pratica riduzione gesso abm appare ben tollerato', 'rx gesso', 'tipo frattura indicata riduzione sintesi chirurgica placca viti', 'mette lista intervento priorità', 'frattempo barccio collo mobilizzazione attiva dita mano dolore tachipirina 1000 1 cp max 3 die']*

6.2.5 Stemming

In ultimo, sui testi così processati e ripuliti, possiamo quindi operare la più importante delle trasformazioni: lo stemming.
Per farlo utilizzeremo ancora una volta il pacchetto *NLTK* il quale al suo interno contiene una serie di stemmers sviluppati ad hoc per ogni singolo linguaggio. Nel nostro caso lo stemmer che andremo ad utilizzare è lo *"Snowball stemmer"*.

```
# Snowball stemmer with rules for the Italian language
ita_stemmer = nltk.stem.snowball.ItalianStemmer()
```

Esso contiene regole ed algoritmi sviluppati ad hoc per la lingua italiana, pertanto risulta ottimo per il caso di studio in analisi.

```
# create list of stemmed sentences
sent_tokenize_list_no_punct_lc_no_stopwords_stem = []
for sent in sent_tokenized_no_punct_no_sw:
    rebuilt_sent = []
    for word in sent.split():
        rebuilt_sent.append(ita_stemmer.stem(word))
    rebuilt_sent = ' '.join(rebuilt_sent)
    sent_tokenize_list_no_punct_lc_no_stopwords_stem.append(
        rebuilt_sent)
```

Il risultato finale prodotto dalle operazioni sopra descritte applicate in sequenza restituisce i seguenti risultati:

1. *['artropat degener rachid dors lomb coxartr dx gonartr', 'buon risult tratt infiltr', 'conferm indic dat colleg poch giorn orson', 'consigl', 'palex 50 mg 1 cp mattin ser 20 gg fin 4 cicl anno', 'fosavanc 1 cp settiman']*

2. *['rifer contusion pols dx', 'eo segn clinic frattur cut indenn defict vnp gom indenn', 'rx frattur dip barton vol', 'pratic riduzion gess abm appar ben toller', 'rx gess', 'tip frattur indic riduzion sintes chirurg placc vit', 'mett list intervent priorit', 'frattemp barcc coll mobilizz attiv dit man dolor tachipirin 1000 1 cp max 3 die']*

Si può quindi facilmente notare come queste operazioni preliminari contribuiscano a manipolare in maniera decisiva e fondamentale i testi, e a prepararli al meglio per i successivi algoritmi di apprendimento automatico.
Le frasi sono state ripulite ed ordinate, ed in esse sono state conservate solo le informazioni chiave, rimuovendo invece quelle superflue o rumorose.

6.2.6 Creazione del dizionario

Ora possiamo quindi utilizzare le frasi così ripulite per generare un dizionario di parole.
Il dizionario sarà una struttura dati contenente l'insieme univoco di tutte le parole presenti nei testi associate ad un identificativo (banalmente un intero progressivo).
La libreria *gensim* ci fornisce una funzione implementata al suo interno per creare il dizionario.

```
# create dictionary
dictionary = gensim.corpora.Dictionary(corpus_processed)
```

Il dizionario conterrà circa 20 mila parole e presenterà una forma del tipo:

{ ... , 'consigl': 20, 'cp': 21, 'esam': 22, 'liev': 23, 'terap': 24, 'bilateral': 25, 'ecograf': 26, 'controll': 27, ... }

Su questo dizionario abbiamo la possibilità di calcolare delle statistiche ed applicare dei filtri più sofisticati rispetto alla semplice rimozione delle stopword.

Ad esempio possiamo impostare delle soglie percentuali al fine di rimuovere parole troppo rare o troppo frequenti.
Le soglie utilizzate in questo caso specifico ci permettono di rimuovere dal dizionario:

1. Le parole che compaiono in meno dell'8% dei documenti
2. Le parole che compaiono in più del 50% dei documenti

Esse vengono rimosse in quanto aumenterebbero la complessità computazionale senza peraltro apportare benefici al modello. Potrebbero invece essere addirittura dannose in quanto parole troppo frequenti o troppo rare sono fonte di rumore.
Inoltre per limitare la dimensione del vocabolario in esso verranno conservate solo le 10 mila parole più frequenti.
Queste soglie sono state scelte molto basse poiché i testi sono caratterizzati da un linguaggio molto tecnico e settoriale e pertanto presentano un vocabolario ristretto.
E' stato inoltre sperimentato il fatto che provando ad innalzarle ulteriormente il modello non presenta miglioramenti in termini di accuratezza, pur pagando in termini di tempi di esecuzione.

```
# Remove very rare and very common words:
# - words appearing less than x times
# - words appearing in more than y% of all documents
# Keep only 10k most frequent words

threshold = 0.08
dictionary.filter_extremes(no_below=math.ceil(
        threshold*n_records), no_above=0.5, keep_n=10000)
```

6.2.7 Modello BoW

Utilizzando il dizionario così costruito possiamo generare il modello BoW, il quale verrà direttamente utilizzato per addestrare i modelli di apprendimento automatico.

```
# create BoW model
bow_corpus = [dictionary.doc2bow(text)
        for text in corpus_processed]
```

Il modello Bag of Words conterrà la rappresentazione in forma numerica delle frasi in input.

In particolare ogni frase è rappresentata come un array di parole ed ogni singola parola è rappresentata come una coppia di valori che rappresentano rispettivamente l'identificativo della parola nel dizionario e il numero delle sue occorrenze all'interno della frase.

Frase	**Rappresentazione**
String	$Array[(int, int)]$
Frase 1	[(0, 1), (1, 1), (2, 1)]
Frase 2	[(3, 1), (4, 2), (5, 1), (6, 1), (7, 1)]
Frase 3	[(8, 1), (9, 1), (10, 2), (11, 1), (12, 2), (13, 1), (14, 1)]
Frase 4	[(15, 1), (16, 1)]
Frase 5	[(5, 2), (11, 1), (17, 1), (18, 1), (19, 1)]
Frase 6	[(14, 1), (20, 1)]
...	...

Tabella 6.1. Modello BoW.

6.2.8 TF-IDF

A questa rappresentazione numerica possiamo anche applicare la trasformazione TF-IDF, la quale sostituisce al numero di occorrenze un valore di frequenza pesata.

Frase *String*	**Rappresentazione** *Array*[(*int*, *int*)]
...	...
Frase 500	[(205, 0.329), (1376, 0.432), (7406, 0.574)]
Frase 501	[(1297, 0.381), (2557, 0.418), (7822, 0.516)]
Frase 502	[(47, 0.211), (1305, 0.370), (1340, 0.321), (2272, 0.745), (4355, 0.399)]
Frase 503	[(363, 0.342), (521, 0.345), (679, 0.279), (858, 0.464), (1941, 0.406)]
Frase 504	[(861, 0.332), (1078, 0.334), (1115, 0.332), (2045, 0.363), (3963, 0.495)]
Frase 505	[(155, 0.382), (5739, 0.650), (5740, 0.656)]
...	...

Tabella 6.2. Modello TF-IDF.

6.2.9 Modello LDA

Con il modello *BoW/TF-IDF* possiamo dunque addestrare il modello LDA.

Prima di iniziare la fase di training è necessario specificare il numero di topic che il modello dovrà identificare all'interno dei vari testi.
Per capire quale sia il miglior valore di *n_topics* eseguiamo una grid search e confrontiamo la chiarezza dei vari modelli addestrati con diversi valori di *n_topics*.

La misura utilizzata per valutare la chiarezza di un modello è il **coherence score**. Esso è una misura di quanto "confusi" o "ben distinti" siano i vari topic tra loro. Se immaginiamo i topic come dei cluster di dati il coherence score non è altro che una misura della distanza tra i vari cluster.

Esistono due diversi tipi di misura della coerenza che si basano sulla stessa idea di fondo: intrinsic UMass e extrinsic UCI. Entrambi calcolano il coherence score come somma degli score tra le varie coppie di parole usate per descrivere i topic.

$$Coherence = \sum_{i<j} score(wi, wj) \tag{6.1}$$

Questa misura può essere vista come la somma di tutti gli archi di un grafo completo.

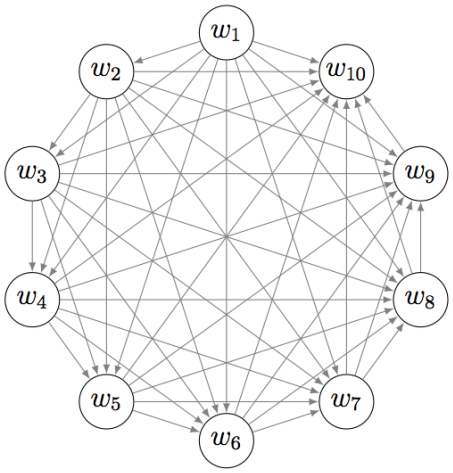

Figura 6.5. Rappresentazione grafica del coherence score.

Tra le due misure scegliamo di utilizzare la Umass:

$$Score_{UMass}(w_i, w_j) = \log \frac{D(w_i, w_j) + 1}{D(w_i)} \tag{6.2}$$

Dove $D(w_i)$ è il numero di documenti contenenti la parola w_i e $D(w_i, w_j)$ è il numero di documenti contenti entrambe le parole w_i e w_j.
Più alto sarà il coherence score meno i topic saranno sovrapposti tra loro e pertanto sarà più semplice per il modello, data una frase in input, classificarla all'interno di uno di essi.

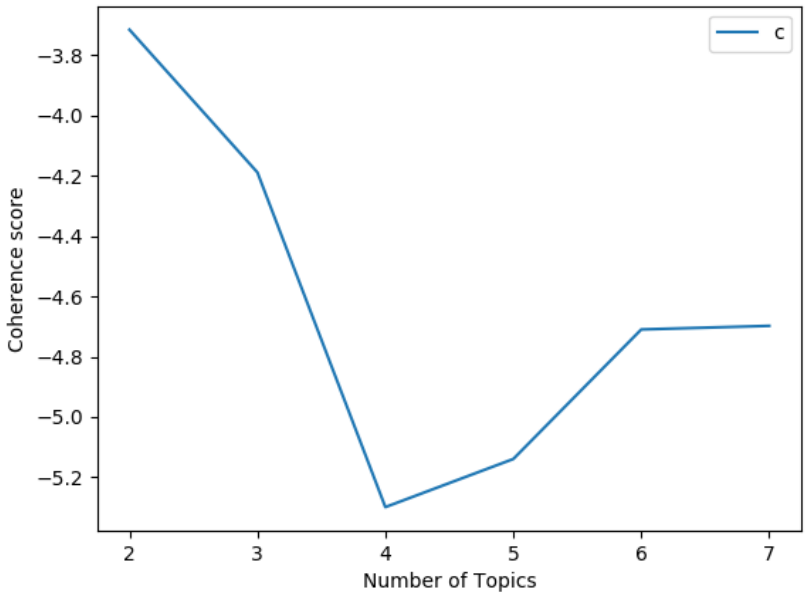

Figura 6.6. Purezza del modello al variare del numero di topics da isolare.

Come evidenziato dal grafico, il modello con due soli topic è nettamente il più performante. L'accuratezza del modello cala drasticamente quando il numero di topics viene settato a 3-4 e risale quando il numero di topics è maggiore di 5. Considerando che per noi sarà necessario attribuire un significato semantico ai vari topic è fortemente sconsigliato aumentare troppo il valore di *n_topics*, poichè diventerebbe difficile distinguere semanticamente tra loro i vari topic. Per questo motivo il modello LDA verrà addestrato con $n\_topics = 2$.

Il modello LDA viene fornito dalla libreria *gensim* insieme ad un set di API per gestirlo opportunamente.

```
# train LDA model
lda_model = gensim.models.ldamodel.LdaModel(
        corpus=bow_corpus, id2word=dictionary,
        num_topics=2, passes=10)
```

Una volta addestrato il modello e ricavati i due *topic*, per capire a che cosa essi si riferiscano dal punto di vista semantico, dobbiamo osservare le parole chiave associate con un peso maggiore relativamente ad ognuno dei due topics e provare a

contestualizzarle all'interno di un ambito semantico.
Nel nostro caso i due *topic* sono associati a parole chiave del tipo:

Topic 0			**Topic 1**	
Peso	*Parola*		*Peso*	*Parola*
0.078	1		0.055	dolor
0.062	controll		0.050	esam
0.048	terap		0.048	mg
0.045	2		0.042	esegu
0.039	pazient		0.033	vis
0.036	cp		0.033	mm
0.036	10		0.032	sinistr
0.030	mes		0.031	norm
0.029	giorn		0.029	destr
0.027	3		0.028	normal
...	...		...	...

Tabella 6.3. Modello LDA.

E' quindi semplice per un essere umano comprendere la distinzione tra i due topic e le loro categorie.

In particolare:

1. Il *primo topic* contiene al suo interno parole relative a terapie farmacologiche (terapia, controllo), quantità di medicinali (1, 3, cp/compresse) e durata della terapia (giorni, mese). Ad esso verranno con ogni probabilità associate frasi appartenenti all'ambito delle terapie farmacologiche.

2. Il *secondo topic* contiene parole relative alle condizioni dei pazienti (dolore, destra, sinistra) e agli approfondimenti diagnostici (visita, esame, eseguire). Ad esso verranno quindi verosimilmente associate le frasi che descrivono le condizioni di salute dei pazienti.

A questo punto siamo quindi in grado dati i testi in input di identificare due diversi topic, assegnare loro un **significato semantico** (terapia farmacologica, condizioni paziente), e successivamente utilizzare il modello per catalogare ogni frase come appartenente o meno ad uno di questi due topic.

Vista la grande mole di dati e l'impossibilità computazionale di processarli tutti insieme, essi sono stati suddivisi in 200 batch da 10 mila campioni l'uno. Su ognuno

di questi batch verranno ripetute separatemente le varie operazioni di preprocessing, training e classificazione.
Ad ogni esecuzione del training del modello rimarranno invariati i topic identificati insieme alle loro parole chiave, ma cambierà l'ordine in cui i topic vengono etichettati (quello che in un run è identificato come *"topic 0"* potrebbe al run successivo essere identificato come *"topic 1"* e viceversa).
Bisogna quindi fare in modo che l'etichetta semantica (*terapia farmacologica*, *condizioni paziente*) possa essere associata in maniera dinamica ai topic generati dal modello LDA.

Per realizzare ciò la soluzione elaborata prevede di eseguire diverse volte l'addestramento del modello e costruire un vocabolario statistico contenente le parole che più frequentemente compaiono associate ad ognuno dei due topic con un alto valore di peso specifico ed assegnare ad ognuno dei due vocabolari la corrispettiva etichetta semantica.

```
for i in range(0,100):
    #train lda model
    print('Training model',  i+1,'...\n')
    lda_model = gensim.models.ldamodel.LdaModel(corpus=bow_corpus,
                id2word=dictionary, num_topics=2, passes=10)
    for idx, topic in lda_model.print_topics(-1):
        print("Topic {}: \n{}".format(idx, topic))
    print('\n')
```

Ad ogni run del programma un nuovo modello viene addestrato e da esso vengono estratte le 10 parole più significative per ognuno dei due topics.
Per ogni parola viene incrementato il relativo contatore delle occorrenze che a fine procedimento verrà utilizzato per determinare quale siano le parole più frequenti.

```
for word in values:
    occ[word] += 1
```

Dalla lista così generata estraiamo le 50 parole più frequenti e collochiamo ognuna di esse nella rispettiva categoria semantica.

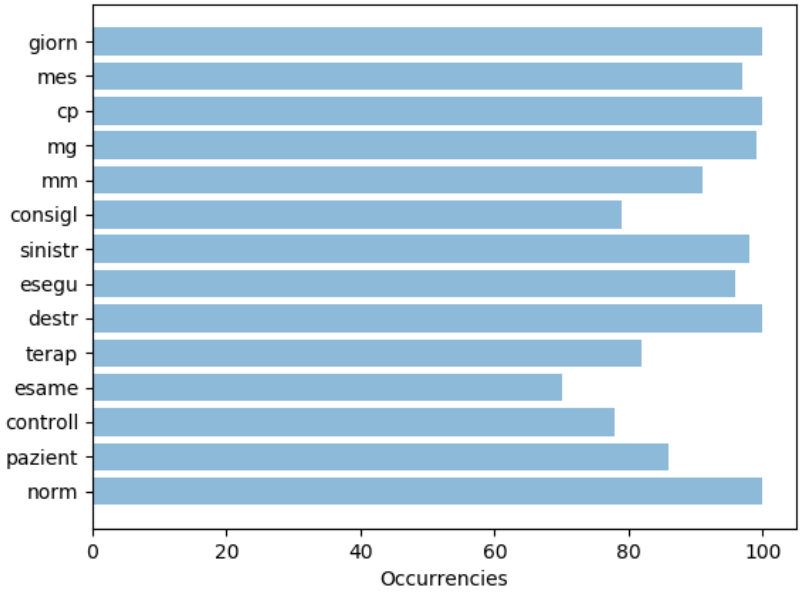

Figura 6.7. Istogramma delle frequenze

Così facendo otterremo quindi due vocabolari – uno per ogni topic – ognuno dei quali contiene le parole chiave più frequentemente associate ad ognuno dei due topic.

Dizionario 1 *Terapia farmacologica*	**Dizionario 2** *Condizioni paziente*
1	pazient
controll	esam
terap	destr
2	rifer
consigl	mm
mg	sinistr
cp	norm
mes	esegu
giorn	vis
3	controll
...	...

Tabella 6.4. Dizionario statistico.

Dal dizionario così costruito quindi, al termine del vero training andremo quindi ad estrarre le parole chiave associate ad ognuno dei due topic, controlleremo in quale dei due vocabolari sono contenute ed estrarremo l'etichetta semantica corretta.

```
keywords_count = [0,0]
for idx, topic in lda_model.print_topics(num_topics=-1, num_words=10):
    for keyword in therapy_vocabulary:
        if topic.find(keyword) > -1:
            keywords_count[idx] += 1
if keywords_count[0] > keywords_count[1]:
    labels = ["Terapia farmacologica", "Condizioni paziente"]
else:
    labels = ["Condizioni paziente", "Terapia farmacologica"]
```

Il risultato finale fornito dal modello LDA e combinato con la tecnica a dizionario per assegnare il significato semantico è il seguente:

Topic 0 ***(Terapia farmacologica)***			**Topic 1** ***(Condizioni paziente)***	
Peso	*Parola*		*Peso*	*Parola*
0.078	1		0.055	dolor
0.062	controll		0.050	esam
0.048	terap		0.048	mg
0.045	2		0.042	esegu
0.039	pazient		0.033	vis
0.036	cp		0.033	mm
0.036	10		0.032	sinistr
0.030	mes		0.031	norm
0.029	giorn		0.029	destr
0.027	3		0.028	normal
...	...		...	...

Tabella 6.5. Modello LDA con associata categoria semantica.

L'algoritmo sviluppato fino ad ora è quindi in grado di analizzare autonomamente i testi, isolare due topic in essi presenti ed assegnare automaticamente un significato semantico ad ognuno dei due topic.
Il modello in nostro possesso verrà quindi ora utilizzato per "catalogare" le frasi che verranno via via fornite al modello come input.
In particolare, data una frase da classificare, il modello sarà in grado di assegnare ad essa due percentuali: la probabilità che essa appartenga al topic 0 e la probabilità che essa appartenga al topic 1.
Tali probabilità vengono calcolate attraverso la seguente formula

$$P(\theta_{i:M}, z_{i:M}, \beta_{i:k}|D; \alpha_{i:M}, \eta_{i:k})$$

nella quale:

- M = numero di documenti.
- N = numero di parole.
- k = numero di topic.
- θ = distribuzione dei topic in ogni documento.
- z = numero di topic per ogni documento.
- β = distribuzione delle parole per ogni topic.
- D = dati a disposizione (corpus).
- α, η parametri.

Fissiamo quindi una soglia, in questo caso 80%: nel caso in cui una delle due percentuali sia superiore a questo valore considereremo la frase come appartenente al rispettivo topic, in caso contrario concluderemo che la frase non appartiene a nessuno dei due topic in quanto tratta altre tematiche.
Iterando questo procedimento su tutte le frasi componenti il nostro dataset saremo quindi in grado di assegnare ad ognuna di esse la relativa etichetta.
Di seguito riporto un esempio di output della classificazione.

1. **zonegran 100mg 1cp ore 8:20**
   Terapia farmacologica: 0.86
   Condizioni paziente: 0.14
   Assigned label ⇒ Terapia farmacologica

2. **da 10 anni grave sindrome depressiva con riacutizzazioni**
   Terapia farmacologica: 0.17
   Condizioni paziente: 0.83
   Assigned label ⇒ Condizioni paziente

3. **si allegano indicazioni intervento chirurgico**
   Terapia farmacologica: 0.32
   Condizioni paziente: 0.68
   Assigned label ⇒ Altro

4. **riferisce dispena da sforzo presente da anni**
   Terapia farmacologica: 0.13
   Condizioni paziente: 0.87
   Assigned label ⇒ Condizioni paziente

5. **come terapia consiglio bart cp 20 mg 2 cp dopo cena per 5 dì indi 1 cp per 10 dì**
   Terapia farmacologica: 0.91
   Condizioni paziente: 0.09
   Assigned label ⇒ Terapia farmacologica

Il lavoro svolto fino a questo punto ci permette quindi di avere a disposizione uno strumento che partendo da testi liberi e non strutturati sia in grado di analizzarli ed estrarre le informazioni richieste. In particolare siamo ora in grado di determinare se e dove all'interno dei vari testi siano presenti riferimenti a terapie farmacologiche e/o a condizioni psico-fisiche dei pazienti.

6.2.10 Prestazioni

Non essendo in possesso di dati già etichettati, per valutare le performance e la precisione del modello realizzato occorre costruire manualmente un test set. A questo proposito è stato estratto un campione casuale di 400 frasi che sono state etichettate a mano; ad ognuna di esse è stata assegnata una label tra *"Terapia farmacologica"*, *"Condizioni paziente"*, *"Altro"*. I campioni così etichettati verranno confrontati con i risultati forniti dal classificatore al fine di valutarne l'accuratezza. Sebbene la dimensione del test set (400 campioni) sia estremamente ridotta l'operazione di labeling manuale è estremamente dispendiosa in termini di tempo, per questa ragione riterremo sufficienti i dati in nostro possesso.

	Precisione	**Richiamo**
Terapia farmacologica	70%	93%
Condizioni paziente	76%	28%

Tabella 6.6. Precisione del modello.

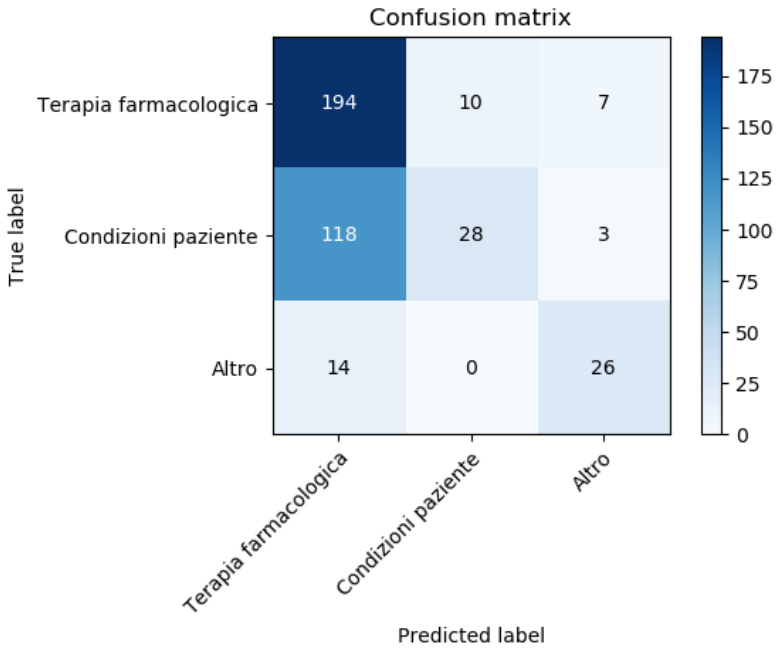

Figura 6.8. Distribuzione dei testi contenenti uno o più farmaci delle varie categorie.

Come evidenziato in tabella il valore di **precisione** è simile per entrambe le categorie.
Esso è discreto per quanto riguarda le condizioni dei pazienti, un po' meno per quanto riguarda le terapie farmacologiche. Analizzando le frasi che vengono classificate erroneamente come terapie farmacologiche notiamo che la maggior parte sono misurazioni di parametri vitali del paziente, ad esempio:

"67 bpm sporadiche"

Queste frasi hanno una struttura estremamente somigliante a quella delle prescrizioni dei farmaci (utilizzo di valori numerici, presenza di unità di misura) ed è pertanto difficile per il modello distinguerle tra loro.
In questo senso il modello potrebbe essere migliorato fornendo una lista di eccezioni, ad esempio un vocabolario di unità di misura relative ai parametri vitali, che ci

permetta di filtrare queste frasi in modo che non inquinino il classificatore.

Per quanto riguarda il **richiamo** invece è evidente la differenza tra le due classi.
La prescrizione di *terapie farmacologiche* segue un pattern standard (utilizzo di valori numerici associati a relative unità di misura e/o durata della terapia) ed è pertanto semplice per il modello apprenderne la struttura ed identificarla in fase di classificazione.
Per quanto riguarda la descrizione delle *condizioni dei pazienti* invece, la modalità descrittiva è estremamente variabile in relazione al contesto e alla tipologia di problematica in questione, per questo motivo sarà più complicato modellarne la struttura e riuscire ad identificarla durante la classificazione.

6.2.11 Tempi di esecuzione

I dati sono stati suddivisi in 200 batch da 10 mila campioni l'uno. Ogni batch richiede 25 secondi per il preprocessing e 50 secondi per l'addestramento del modello LDA, per un totale di 1 minuto e 15 secondi. L'iterazione completa su tutti i dati del database richiede quindi circa 4 ore di tempo.

	1 batch	**Intero database**
Preprocessing	25s	1h 25m
Training	50s	2h 45m
Totale	1m15s	4h 10m

Tabella 6.7. Tempi di esecuzione.

6.3 Estrazione delle informazioni associate ai farmaci

Come ulteriore argomento di ricerca può risultare interessante associare informazioni quali tempistiche e dosaggi alle prescrizioni dei vari farmaci.
I farmaci da monitorare appartengono ad una specifica classe di farmaci (J01) e la loro presenza è limitata ad un ristretto sottoinsieme di testi.

Analizzando i dati di partenza osserviamo infatti che i farmaci da monitorare appartengono a 30 famiglie diverse, e che tra tutti i testi contenuti nella base di dati solamente 229.179 di essi contengono uno o più dei farmaci in analisi.

La presenza dei farmaci monitorati presenta la seguente distribuzione lungo le diverse famiglie farmacologiche:

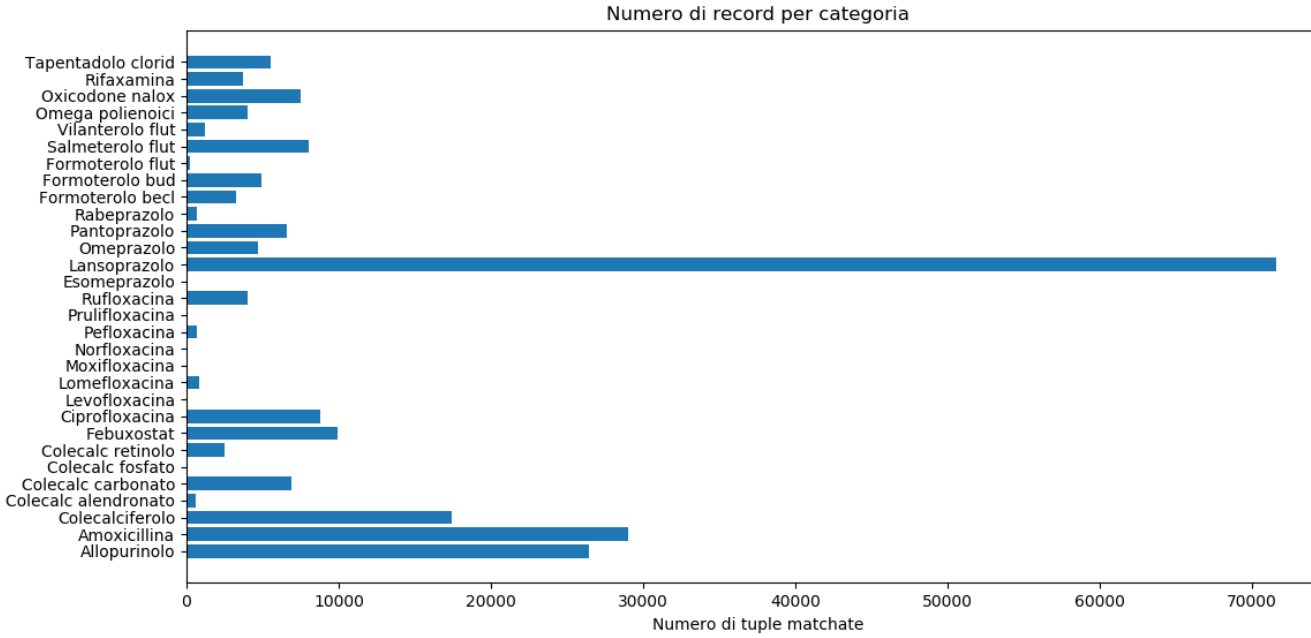

Figura 6.9. Distribuzione dei testi contenenti uno o più farmaci delle varie categorie.

L'idea implementata per risolvere il problema consiste nell'utilizzare SolR come filtro per estrarre rapidamente i testi a noi necessari, e successivamente eseguire sui testi così ottenuti la ricerca della posologia tramite regular expressions.
Le due tecniche vengono combinate in questo modo al fine di:

- Sfruttare la potenza e la rapidità di ricerca testuale di SolR. Infatti eseguire la ricerca tramite regex su tutti i testi sarebbe troppo dispendioso dal punto di vista computazionale.

- Sfruttare la precisione delle regex per ricercare, all'interno dei testi da considerare, le informazioni a noi necessarie.

6.3.1 Vocabolario a disposizione

Il vocabolario di farmaci a noi fornito presenta 224 farmaci e una forma del tipo:

ID	CL L1	CL L2	COD ATC	CHIAVE RICERCA	DATA ULTIMO AGG
210	Allopurinolo	Allopurinolo	M04AA01	Allopurinolo	12/03/2019 19:11
212	Allopurinolo	Allopurinolo	M04AA01	Allopurinolo Doc	12/03/2019 19:11
211	Allopurinolo	Allopurinolo	M04AA01	Allopurinolo Doc generici	12/03/2019 19:11
213	Allopurinolo	Allopurinolo	M04AA01	Allopurinolo Molteni	12/03/2019 19:11
214	Allopurinolo	Allopurinolo	M04AA01	Allopurinolo Sandoz	12/03/2019 19:11
215	Allopurinolo	Allopurinolo	M04AA01	Allopurinolo Teva Italia	12/03/2019 19:11
216	Allopurinolo	Allopurinolo	M04AA01	Allurit	12/03/2019 19:11
217	Allopurinolo	Allopurinolo	M04AA01	Zyloric	12/03/2019 19:11
38	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Abba	07/03/2019 00:00
39	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Abioclav	07/03/2019 00:00
40	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Acadimox	07/03/2019 00:00
41	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Aklav	07/03/2019 00:00
226	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Amoxicillina ac cla	12/03/2019 00:00
227	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Amoxicillina ac clavulanico	12/03/2019 00:00
228	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Amoxicillina acido clavulanico	12/03/2019 00:00
42	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Anival	07/03/2019 00:00
43	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Augmentin	07/03/2019 00:00
44	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Aveggio	07/03/2019 00:00
37	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Clav	07/03/2019 00:00
45	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Clavomed	07/03/2019 00:00
46	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Clavulin	07/03/2019 00:00
47	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Homer	07/03/2019 00:00
48	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	J01CR02	Klavux	07/03/2019 00:00

Figura 6.10. Dizionario dei farmaci da monitorare.

In esso vengono riportate le diverse nomenclature dei vari farmaci, e per ogni farmaco una serie di informazioni utili come identificativo, categoria e codice ATC. I farmaci sono raggruppati nel vocabolario per categoria (codice ATC), quindi per ognuna di queste categorie eseguiremo una ricerca sulle tuple del database per ricercare ed estrarre i testi che al loro interno contengono riferimenti alle chiavi di ricerca in questione.
Le categorie non sono mutuamente sovrapposte, pertanto l'elaborazione dei testi relativi ad ognuna di esse può essere eseguita in parallelo.

Il vocabolario è fornito sotto forma di file .csv, quindi importiamo il file e viste le ridotte dimensioni manteniamolo in memoria.

```
def create_search_dict_from_file(self):
    search_dict = {}
    with open(self.key_file_path, 'r') as fin:
        df_keys = pd.read_csv(fin, sep='|', header=None)

    for index, row in df_keys.iterrows():
        ID = row[0]
        CL_LIV1 = row[1]
        CL_LIV2 = row[2]
        COD_ATC = row[3]
        CHIAVE = row[4]
        DT_ULTIMO_AGG = row[5]

        search_dict[COD_ATC].update(...)

    return search_dict
```

6.3.2 Processing parallelo

Grazie alla libreria *multiprocessing* sfruttiamo la possibilità di creare un **pool** di processi a cui assegnare di volta in volta i vari task.
Ogni task consiste nell'eseguire per una categoria di farmaci la ricerca delle chiavi all'interno dei vari testi, ed in seguito eseguire l'elaborazione sui testi estratti.
Siccome le categorie di farmaci sono indipendenti l'una dall'altra così come il processing da eseguire sui testi, l'operazione può essere facilmente eseguita in parallelo dai vari processi.

```
# Execute the solr search for each key
# that groups the keywords to search
print('Pooling ', self.num_processes, ' workers.')
p = Pool(processes=self.num_processes)
for key in input_dict.keys():
        res = p.apply_async(self.do_processing, args=(...))
p.close()
p.join()
```

Al termine di questo lavoro il processo principale si occuperà di raccogliere i risultati prodotti dai vari sotto-processi e salvarli su database.
Come mostrato dal grafico la parallelizzazione permette di ridurre drasticamente i tempi di esecuzione.

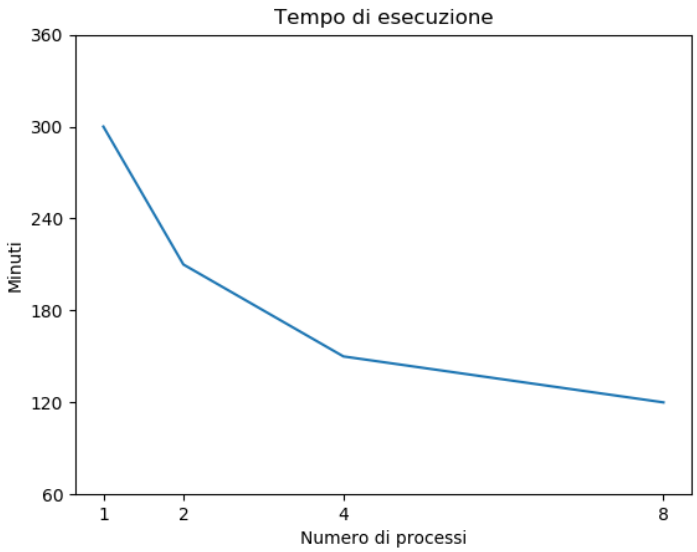

Figura 6.11. Variazione del tempo di esecuzione all'aumentare del numero di processi utilizzati.

6.3.3 Query SolR

Utilizziamo le informazioni così ricavate per costruire delle query SolR da utilizzare per estrarre dal database le tuple che soddisfano i requisiti.
La query SolR viene costruita tenendo conto dei seguenti aspetti:

- Vogliamo estrarre dalla base di dati tutte e sole le tuple che soddisfano i criteri di ricerca
- Vogliamo ricercare la presenza esclusivamente dei farmaci presenti nel vocabolario
- I nomi di farmaci presenti nel dizionario sono talvolta composti da più di una sola parola (es. *"Allopurinolo Sandoz"*), bisogna quindi ricercare la presenza contemporanea di tutti i vocaboli che compongono il nome, tenendo conto del fatto che potrebbero essere divisi da simboli di punteggiatura o altri vocaboli non appartenenti al nome stesso.
- Una volta estratte le tuple richieste serve un metodo per evidenziare all'interno delle stesse le chiavi che hanno generato il match
- Bisogna gestire opportunamente la dimensione dei risultati ritornati da SolR, i quali vengono forniti in file di formato *.json*

Per soddisfare questi requisiti vengono quindi settati i seguenti parametri SolR:

1. Campo della base di dati sul quale eseguire la ricerca: il campo di testo
2. Chiave da ricercare: i nomi dei farmaci del vocabolario
3. Massima distanza reciproca sulla quale eseguire la proximity search per trovare farmaci composti da più di una parola: 5
4. Limite sul numero di tuple ritornate ed esecuzione del paging dei risultati: 100
5. Attivazione dell'highlighting per evidenziare all'interno del testo le parole chiave matchate

```
def query_solr(...):

    baseaddress = "http://" + url + ":"
                + port + "/solr/"
                + solr_schema + "/select?q="

    for idx, c_word in enumerate(word):
            ...
            baseaddress += "(" + default_field
                        + ":" + c_word +")"
```

```
...

# paging of results
baseaddress += "&start=" + str(offset)

# limiting number of rows returned
baseaddress += "&rows=" + str(row)

# highlighting
 baseaddress += "&hl.method=unified&hl.fl=TESTO&
      hl.fragsize=0&hl=on"

response = requests.get(baseaddress)

return word, response
```

Le query solR così generate saranno quindi del tipo:

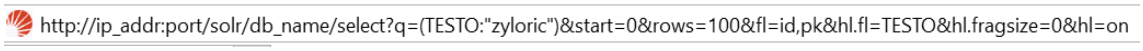

Figura 6.12. Query solR d'esempio.

6.3.4 Elaborazione dei testi restituiti

I risultati da esse ritornati conterranno quindi tra le varie informazioni anche i testi della base di dati contenenti le parole chiave ricercate, con le stesse keywords evidenziate da un **tag** **.

```
{
  "responseHeader":{
    "status":0,
    "QTime":8,
    "params":{
      "q":"TESTO:\"zyloric\"",
      "hl":"on",
      "fl":"id,pk,DATA_ULTIMO_INS,CODICE_EPISODIO",
      "hl.fragsize":"0",
      "start":"0",
      "hl.fl":"TESTO",
      "hl.method":"unified",
      "rows":"100",
      "_":"1559724667470"}},
  "response":{"numFound":7641,"start":0,"docs":[
      {
        "DATA_ULTIMO_INS":"2019-03-04T00:00:00Z",
        "CODICE_EPISODIO":"13696882",
        "id":"c09a625d-5eb5-4642-98ad-e0f8ecdc1653",
        "pk":"12652569"},
      {
        "DATA_ULTIMO_INS":"2019-03-04T00:00:00Z",
        "CODICE_EPISODIO":"13526107",
        "id":"7202fd82-6fbd-4277-a7e7-8d0939ad609f",
        "pk":"12489853"},

},
"highlighting":{
  "c09a625d-5eb5-4642-98ad-e0f8ecdc1653":{
    "TESTO":[" MOTIVO DELL'ACCESSO: Richiede ricettazione <em>Zyloric</em> in tp cronica========ANAMNESI:
  "7202fd82-6fbd-4277-a7e7-8d0939ad609f":{
    "TESTO":["Si valuta anziana paziente con episodio di IRA su IRC per cardiopatia , portatrice di PM.==
  "39ebb176-7abf-4a54-8308-f0db6229e718":{
    "TESTO":["MOTIVO DELL'ACCESSO:algia I dito piede dx========ANAMNESI:pz ipertesa, iperuricemia tratta
```


Figura 6.13. Risultati forniti da SolR sotto forma di file json.

Navigando il json restituito possiamo recuperare dai testi le parole che hanno effettuato il match, e metterle in relazione con le informazioni necessarie come ad esempio l'identificativo del farmaco, la categoria di appartenenza, il codice ATC. Nella sezione *"highlighting"* viene infatti fornito il testo di partenza arricchito con l'aggiunta dei tag, i quali ci permettono di individuare velocemente in quale posizione del testo compaiano i farmaci ricercati.

```
# number of matching records
tot_records = response.json()['response']['numFound']
for doc in response.json()['response']['docs']:
        cur_record += 1  # number of docs processed to this point
        # get text
        text = response.json()['highlighting'][id]["TESTO"][0]
```

L'informazione sulla posizione nel testo è la chiave per eseguire le analisi aggiuntive su posologia e prescrizione dei farmaci.

Per distinguere se la citazione di un farmaco avviene in anamnesi (il paziente sta già assumendo il farmaco al momento del ricovero) o in dimissione (il paziente deve iniziare la terapia farmacologica in seguito all'intervento subito) è sufficiente considerare la posizione della menzione all'interno del testo.
I referti infatti seguono una struttura logica standard nella quale viene prima descritta la condizione del paziente al momento dell'ingresso in clinica, successivamente vengono illustrati gli interventi effettuati sul paziente e riportate eventuali considerazioni dello specialista e solo infine vengono citate terapie e consigli da seguire durante la riabilitazione.
Per questa ragione i farmaci che compaiono citati nella parte finale del referto saranno riferiti a terapie da intraprendere, mentre gli altri si riferiranno a terapie seguite in precedenza o durante la permanenza in clinica.

```
result = regex.search(text, pos)
pos = result.start(1)  # offset of drug in text
if result.group(2) != None:  # if dosage has been found
        qty = result.group(2)  # get dosage
        # write result to output file
        # if we are in the second half of the text:
        if pos / len(text) > 0.5 and second_half == False:
                second_half = True  #dosage found in "dimissione"
```

Per quanto riguarda la posologia invece bisogna eseguire un'ulteriore ricerca sui testi estratti nei passaggi precedenti.
Il dosaggio relativo ai medicinali prescritti non sempre è riportato, talvolta è sottinteso o semplicemente omesso, pertanto non possiamo dare per scontata la sua presenza ma dobbiamo verificarne l'esistenza.
L'idea implementata è quella di sfruttare i tag aggiunti al testo da SolR per localizzare le keywords esaminate tramite l'uso di espressioni regolari, e ricercare se in prossimità di esse compaiono valori numerici e/o cifre decimali (che verosimilmente

rappresenteranno il dosaggio del medicinale di riferimento).
Per ogni match trovato ed evidenziato da SolR viene quindi creata un'espressione regolare che permetta di ricercare se sia presente la posologia ad esso associata.
La posologia viene ricercata come una stringa di massimo 3 caratteri numerici ad una distanza di massimo due parole dal farmaco di riferimento.
Le espressione regolari create saranno quindi del tipo:

```
<em>nome_farmaco<\/em>\W+(?:\w+\W+){0,2}?([0-9]{1,3})\b
```

Figura 6.14. Template delle regex generate.

```
hit_words = re.findall(r'<em>(.*?)</em>', text)
pos = 0
for hit_word in hit_words:
    #per ogni farmaco trovato cerca la relativa posologia
    found_qty = False #determina se la posologia e' stata trovata
    regex = r'\b'
    regex += '.*'.join(hit_word.split())
    # la posologia e' una stringa di massimo 3 caratteri numerici
    # e si puo trovare ad una distanza di massimo due parole
    # dal farmaco
    regex += r'\W+(?:\w+\W+){0,2}?([0-9]{1,3})\b'
    regex = re.compile(regex)
    result = regex.search(text, pos)
    if result != None: # se la posologia e' stata trovata
        found_qty = True #setta il flag
        qty = result.group(1) # quantita di farmaco prescritto
        pos = result.start(1) # posizione della posologia
```

L'algoritmo così concepito permette di produrre risultati del seguente tipo:

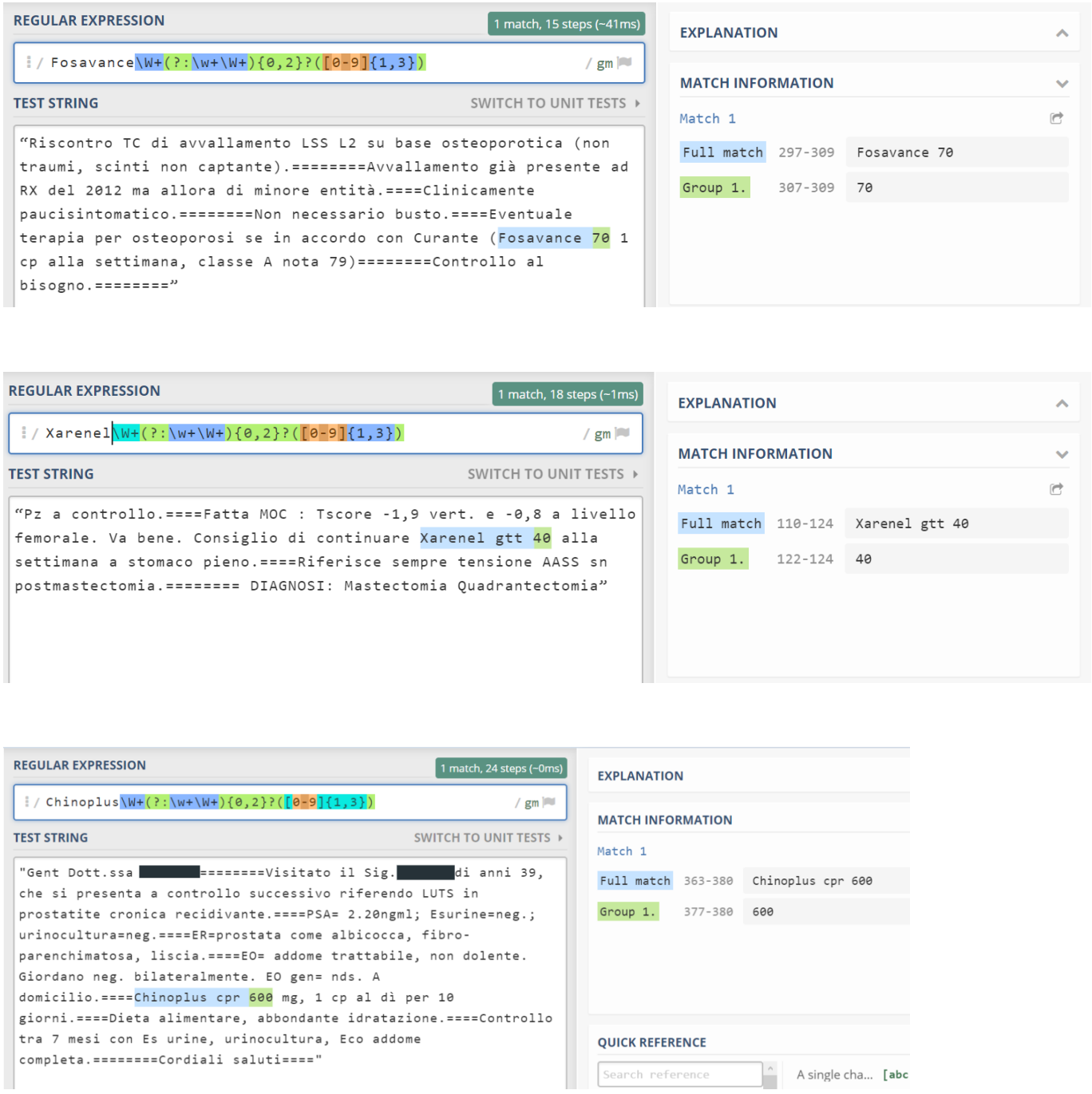

Figura 6.15. Posologia associata ai farmaci da monitorare.

6.3.5 Raccolta dei risultati e scrittura su una base di dati

Ogni processo al termine dell'elaborazione di un gruppo di *chiavi* produce un file contenente per ogni documento le informazioni strutturate (codice documento, codice episodio, codice ATC...) e le terapie da monitorare presenti all'interno del testo. Le terapie sono divise tra anamnesi e dimissione, ed accanto ad ogni farmaco è riportato il relativo dosaggio ove presente.

A questo punto non resta che mettere in join i risultati trovati dai vari processi per

COD DOC	COD EPIS	COD ATC	CL L1	CL L2	ULTIMO AGG	TODAY	TERAPIE ANAMNESI	TERAPIE DIMISSIONE
13202280	14279292	M04AA01	Allopurinolo	Allopurinolo	04/03/2019	15/05/2019		ALLOPURINOLO 100;
14015310	15149411	M04AA01	Allopurinolo	Allopurinolo	04/03/2019	15/05/2019	Zyloric;	
13685610	14793703	M04AA01	Allopurinolo	Allopurinolo	04/03/2019	15/05/2019	allopurinolo;	
13631471	14735907	M04AA01	Allopurinolo	Allopurinolo	04/03/2019	15/05/2019		Allopurinolo 300;
14311969	15464022	M04AA01	Allopurinolo	Allopurinolo	04/03/2019	15/05/2019	Zyloric;	
12518851	13557131	M04AA01	Allopurinolo	Allopurinolo	04/03/2019	15/05/2019		allopurinolo 300;
12054593	13052300	M04AA01	Allopurinolo	Allopurinolo	13/03/2019	15/05/2019		ZYLORIC;allopurinolo 300;
12676888	13672941	M04AA01	Allopurinolo	Allopurinolo	13/03/2019	15/05/2019		Allopurinolo 300;
13291612	14372890	M04AA01	Allopurinolo	Allopurinolo	04/03/2019	15/05/2019		ZYLORIC 300;
12085032	13061012	M04AA01	Allopurinolo	Allopurinolo	13/03/2019	15/05/2019		Zyloric 300;
14371677	15529156	M04AA01	Allopurinolo	Allopurinolo	04/03/2019	15/05/2019	Zyloric;	
14479263	15646794	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	04/03/2019	15/05/2019		Augmentin;
14055413	15163939	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	13/03/2019	15/05/2019	Augmentin;	
13704267	14813295	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	04/03/2019	15/05/2019		Augmentin;
12345446	13374224	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	04/03/2019	15/05/2019	Augmentin 1;	
12681337	13726816	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	04/03/2019	15/05/2019		Augmentin 1;
12669121	13714409	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	04/03/2019	15/05/2019	AUGMENTIN;	
13591488	14692713	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	13/03/2019	15/05/2019	Augmentin 1;	
13044263	14111506	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	04/03/2019	15/05/2019	Augmentin;	
11649505	12615054	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	13/03/2019	15/05/2019	AUGMENTIN 1;	
14814790	15999509	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	15/05/2019	15/05/2019		Augmentin 1;
12632251	13675608	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	04/03/2019	15/05/2019		Amoxicillina Ac Clavulanico;
13481330	14533605	J01CR02	Amoxicillina-acido clavulanico	Amoxicillina+acido clavulanico	13/03/2019	15/05/2019		amoxicillina ac clavulanico;

Figura 6.16. Risultati prodotti dall'analisi sulle varie categorie ATC.

ottenere il risultato complessivo.
Le tuple all'interno del database sono identificate da *codice documento* e *codice episodio*, che saranno gli attributi sui quali eseguire il join. Le terapie verranno messe in join concatenandole le varie stringhe trovate.

```
def doc_group_atc(...):
        """
        method to group string on columns by document and episode
        """
        ...

        format_doc = ",".join(map(lambda x: "'{}'"
                        .format(x), list_of_doc))
        format_episode = ",".join(map(lambda x:
                        "'{}'".format(x), list_of_episode))

        main_query = self.config.get('QUERY',
                        'read_query_vertica')

        with connect(**self.conn_info) as conn:
                cur = conn.cursor()
                cur.execute(main_query.format(
                        format_doc, format_episode))
                df_result = pd.DataFrame(cur.fetchall())

                doc_temp_df = df_result.groupby(
                        ['CODICE_EPISODIO',
                        'CODICE_DOCUMENTO']).agg(
                        lambda x: ','.join([s for s in
                        x.values if s != '']))).reset_index()
                doc_temp_df.to_csv(self.temp_doc_file_path,
                        index=False, header=False)

                # copy data into the empty table
                with open(self.temp_doc_file_path, 'r') as fs:
                        cur.copy(eval(self.config.get('QUERY',
                                'qry_to_copy_doc')), fs)
        return 0
```

La situazione finale del database sarà la seguente.

CODICE_DOCUMENTO	CODICE_EPISODIO	TERAPIE_ANAMNESI	TERAPIE_DIMISSIONE
10958807	10003079	lansoprazolo 30	Augmentin 100
10957145	10017364	Allopurinolo,Allopurinolo	Lansoprazolo
10975031	10024246	lansoprazolo	Augmentin 1,Lansoprazolo 30
10948748	10027459	Augmentin	Lansoprazolo
10994693	10038238	allopurinolo	Dibase 50
10956101	10039984	Augmentin 1	Lansox 15
10968595	10041252	zyloric 300	Zyloric 300,Augmentin 1,Lansox 30
10986277	10041660	Allopurinolo 300,Allopurinolo 300	Augmentin 1
11011205	10054219	Zyloric 300	Lansox 30
11014802	10057894	Lansox 30	Zyloric,Allopurinolo 300
10995701	10062824	amoxicillina acido clavulanico	Lansox 15,Allopurinolo 300
10940009	10062957	Lansoprazolo	Allopurinolo
10991605	10067367	Amoxicillina Acido Clavulanico	Lansoprazolo 15
11031274	10074036	allopurinolo 300	Allopurinolo
11009283	10085937	augmentin	Zyloric 300,Augmentin 1,Lansox 30
10998363	10095681	allopurinolo,allopurinolo	Lansoprazolo 30
10961979	10096374	Augmentin	Dibase 250,Lansox 30
11060319	10102679	allopurinolo,allopurinolo	Allopurinolo 300
11113348	10155951	Allopurinolo 100,Lansox 30	Lansoprazolo
11104496	10159219	amoxicillina acido clavulanico	Allopurinolo
11117794	10160107	Lansox	Lansoprazolo,Lansox 30
11108188	10160655	clav	AUGMENTIN 1
11103557	10161114	ZYLORIC,LANSOX 30	Allopurinolo 300,Lansox 30

Figura 6.17. Risultato finale prodotto dalla ricerca della posologia associata ai farmaci.

Ad ogni record corrisponde un episodio, per ognuno di essi vengono riportate le informazioni strutturate richieste ed i farmaci da monitorare trovati nel testo, la loro classificazione (anamnesi, dimissione) e la relativa posologia ove presente.

6.3.6 Amazon Comprehend Medical

Data l'attualità del problema trattato, colossi dell'hi-tech mondiale si stanno muovendo per sviluppare tecnologie e prodotti che risolvano problemi analoghi a quelli trattati in questa tesi.

L'esempio più importante è Amazon Comprehend Medical, un servizio sviluppato dall'omonima compagnia con l'obiettivo di individuare ed estrarre informazioni utili da testi medici non strutturati.
Amazon Comprehend Medical utilizza il Natural Language Processing e modelli di machine learning per:

- Permettere a medici e specialisti di gestire i bollettini medici ed accedere velocemente alle informazioni in essi contenute, al fine di ottenere in maniera efficiente il quadro clinico di un paziente.
- Permettere ad organizzazioni ed enti sanitari di ottenere statistiche riguardanti l'utilizzo di particolari farmaci o il decorso di determinate patologie.

Amazon Comprehend Medical utilizza un modello pre-addestrato per esaminare ed analizzare insiemi di documenti al fine di estrarre le informazioni richieste. Questo modello viene continuamente addestrato su grandi quantità di testi e pertanto non è necessario fornire dati di training in input al modello.

Il servizio identifica ed estrae informazioni riguardanti cinque diverse categorie:

- ANATOMY
- MEDICAL_CONDITION
- MEDICATION
- PROTECTED_HEALTH_INFORMATION
- TEST_TREATMENT_PROCEDURE

Per ogni entità individuata il modello fornisce la relativa classe di appartenenza, la posizione nel testo ed un confidence score, ovvero un valore numerico rappresentante il livello di accuratezza della previsione.

Capitolo 7

Conclusioni e sviluppi futuri

In questa tesi, si è sviluppata una tecnica di text mining per l'estrazione automatica di informazione da testi di provenienza medica.
I testi sono stati raccolti così come redatti dai vari specialisti, e pertanto non sono stati concepiti per essere sottoposti ad algoritmi di apprendimento automatico. Gli algoritmi sviluppati cercano di superare questo problema cercando di simulare un'intelligenza quanto più umana e completa possibile.
Il concetto innovativo si basa sulla capacità di analizzare autonomamente i testi e identificare al loro interno un certo numero di "topic" o "argomenti", e di essere successivamente in grado di riconoscerne la presenza all'interno dei testi stessi.
Un'analisi preliminare dei dati ha reso necessaria una pesante fase di pre-processing dei testi, allo scopo di ripulirli e uniformarli per poter eseguire un corretto addestramento del modello di apprendimento automatico.
In seguito sui risultati forniti dal modello si è sfruttato l'ausilio di vocabolari per associare ad ogni topic il proprio significato semantico.
Il modello così ricavato è stato usato come classificatore per le frasi fornite in input.
In secondo luogo sono state realizzate delle tecniche in grado di associare ad ogni terapia farmacologica la relativa posologia.

Come sviluppi futuri si possono considerare alcune estensioni ai dizionari e ai modelli esistenti, che permetterebbero di incrementare le performance in fatto di accuratezza.

Sicuramente la raccolta di dati già etichettati permetterebbe di avere a disposizione un training set da poter usare in fase di addestramento, ed aprirebbe la strada a numerose nuove possibilità in fase di classificazione. Un training set sufficientemente grande ci darebbe infatti la possibilità di addestrare i tradizionali classificatori citati nel quarto capitolo, i quali al momento non sono utilizzabili proprio a causa della mancanza di una base di conoscenza.

Per superare i limiti imposti dall'assenza di dati etichettati, si potrebbero utilizzare le label generate dal modello LDA per generare un training set ad-hoc. Con i dati così ottenuti si potrebbe tentare di addestrare un classificatore standard (ad esempio SVM, KNN o Naive-Bayes) andando di fatto a realizzare una sorta di apprendimento semi-supervisionato.
Il classificatore così ottenuto presenterebbe sicuramente lo svantaggio di partire da dati rumorosi, andando quindi ad apprendere anche gli errori commessi dal modello LDA. In compenso però, esso potrebbe essere utlizzato per classificare dati in real-time, a differenza del modello LDA che invece necessita di essere sottoposto di volta in volta ad una lunga fase di training.

E' inoltre possibile potenziare il modello di apprendimento automatico con l'ausilio di nuovi dizionari da utilizzare come filtri, i quali contribuirebbero ad aumentare l'accuratezza.
Il modello risentirebbe infatti di pesanti miglioramenti qualora avesse a disposizione una serie di regole con le quali filtrare le eccezioni e gli outliers che inquinano il modello.
Per esempio, come citato in precedenza, per ovviare al fatto che il modello spesso confonde cifre relative a dosaggi terapeutici con misurazioni di parametri vitali del paziente, sarebbe vantaggioso avere un vocabolario delle unità di misura di una e dell'altra categoria, così da poter associare univocamente ad una delle due la quantità numerica rilevata.

Per quanto riguarda la ricerca della posologia sarebbe vantaggioso arricchire il vocabolario dei farmaci con ulteriori informazioni utili ad eseguire il check dell'integrità dei dosaggi trovati.
Ad esempio siccome i farmaci sono prodotti e rivenduti in quantità prestabilite (ad esempio la tachipirina esiste solo in quantità di 250mg, 500mg e 1000mg) si potrebbero integrare queste informazioni con la lista dei farmaci. In questo modo una volta identificata una posologia, tramite un lookup della tabella, si potrebbe verificare l'integrità della stessa filtrando eventuali incongruenze dovute a errori di battitura o disattenzioni dello specialista.

Bibliografia

[1] Aggarwal C. C., Zhai C., *Mining Text Data*, Springer-Nature, 1 febbraio 2012.

[2] Blei D. M., Ng A. Y., Jordan M. I., *Latent Dirichlet Allocation*, Berkeley, 3 gennaio 2003.

[3] Chapelle O., Scholkopf B., Zien A., *Semi-Supervised Learning*, MIT Press, 24 febbraio 2009.

[4] DataCamp: *https://www.datacamp.com*

[5] DataScience+: *https://datascienceplus.com*

[6] Dorre J., Gerstl P., Seiffert R., *Text mining: Finding nuggets in mountains of textual data*, 1999.

[7] GitHub: *https://github.com*

[8] Goldberg Y., *Neural Network Methods for Natural Language Processing*, Morgan & Claypool, 30 aprile 2017.

[9] Han J., Kamber M., *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 6 settembre 2000.

[10] Hearst M., *What is Text Mining?*, Berkeley, 17 ottobre 2003.

[11] Hearst M., *Untangling text data mining*, Berkeley, 26 giugno 1999.

[12] Hofmann T., *Unsupervised Learning by Probabilistic Latent Semantic Analysis*, Douglas Fisher, gennaio 2001.

[13] Manning C., Schutze H., *Foundations of Statistical Natural Language Processing*, MIT Press, 1 agosto 1999.

[14] Mitchell T. M., *Machine Learning*, McGraw-Hill, 1997.

[15] Nigam K., Mccallum A. K., Thrun S., Mitchell T., *Text Classification from Labeled and Unlabeled Documents using EM*,William Cohen, maggio 2000.

[16] Porter M. F., *An algorithm for suffix stripping*, Cambridge, 1980.

[17] Radim Rehurek, gensim: *https://radimrehurek.com/gensim/*

[18] Sarkar D., *Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from your Data*, Apress, 1 dicembre 2016.

[19] Srividhya V., Anitha R., *Evaluating Preprocessing Techniques in Text Categorization*, International Journal of Computer Science and Application Issue, 2010.

[20] StackExchange: *https://stackexchange.com*

[21] StackOverflow: *https://stackoverflow.com*
[22] Towards Data Science: *https://towardsdatascience.com*
[23] Wallach H. M., *Topic modeling: Beyond Bag-of-Words*, Cambridge, 25 giugno 2006.
[24] Wikipedia: *https://en.wikipedia.org*
[25] Willet P., *The Porter stemming algorithm: then and now*, Emerald Group Publishing Limited, 2006.