Politecnico di Milano
Facoltà di Ingegneria dei Sistemi

Corso di Studi in

Ingegneria Matematica

Tesi di Laurea Specialistica

Text Mining: riconoscimento di elementi diagnostici in lettere di dimissione ospedaliera

Candidato:
Lorenzo Vayno
Matr. 736136

Relatore:
Prof.ssa Anna Maria Paganoni

Correlatore:
Dott. Stefano Ballerio

Anno Accademico 2009/2010

A mio padre, dispensatore insostituibile di affetto e saggezza

Multa non quia difficilia sunt non audemus,
sed quia non audemus sunt difficilia.
Seneca

Un cuore matto, matto da legare.
Antonio Ciacci

Ringraziamenti

Ringrazio la professoressa Anna Maria Paganoni, che dal primo anno ammiro come insegnante e di cui ora posso apprezzare le straordinarie qualità umane. Perché la sua disponibilità, la sua schietta gentilezza e la passione che trasmette per lo studio della statistica ne fanno un esempio, rendono possibile un vero scambio di idee e ti fanno sentire quasi un amico, più che un semplice studente. Ringrazio il Dott. Stefano Ballerio per la sua grandissima disponibilità e simpatia, per la straordinaria passione che mette nel suo lavoro e l'interesse che ha dimostrato verso il mio. Ringrazio poi il Dott. Pietro Barbieri per la fiducia accordatami, dandomi la possibilità di lavorare sulle lettere di dimissione.

Le persone che mi sono state vicine negli anni del mio variegato percorso universitario sono tante. Inutile provare a ringraziarle tutte. Di certo, più di tutti sono grato ai miei genitori e a tutta la mia famiglia. Grazie mamma per la tua vicinanza, il tuo supporto e le tue cure amorevoli soprattutto durante gli ultimi esami: ti devo almeno un paio di 30 e lode. Grazie papà per aver creduto in me, per avermi sempre indicato la strada migliore, pur lasciandomi sempre libero di scegliere. Di sicuro avrò bisogno ancora per lungo tempo dei tuoi consigli. Ringrazio poi mia nonna Savina, per le specialità culinarie venete e perché un giorno mi disse "*vorrei vederti laureato*". Hai aspettato un po', ed eccomi qua addirittura con una doppia laurea!

Ringrazio poi tutti gli amici, vicini e lontani. Quelli con cui ho condiviso i primi e gli ultimi anni al Poli e tutti quelli che ho incontrato a Lione. Tutti quelli che hanno reso questi anni speciali con le loro risate ed il loro affetto. Non posso di certo nominarli tutti, ma un grazie speciale va ai compagni di avventure che mi sono stati sempre vicini a Milano e Lione, primi fra tutti Guido, Max e Vale.
Ringrazio gli amici di Pinzolo & dintorni, alcuni dei quali si riferiscono al sottoscritto con l'appellativo di *dottore*, sulla fiducia, dal 2001 (anno del passaggio in terza superiore - se questo non è credere in qualcuno...).
Ringrazio gli amici del liceo, che sbaglio a chiamare così poiché alcuni di loro sono ormai diventati amici di sempre.
Ringrazio poi tutti gli amici musicisti con cui ho condiviso l'esperienza unica, mistica, della musica. Perché con la musica la comunicazione tra gli esseri umani sublima ad un livello superiore. Grazie quindi ai mitici Clona, ai Kaos, ai Les No, a Jo e i ragazzi della MALA, ai Victoria Caffè e a tutti quelli con cui ho suonato anche solo in una jam. Perché sono stati come ossigeno, in mezzo a tanto studio. Perché, prima di tutto, mi hanno fatto sentire libero per davvero.

Abstract

Text mining is a quite recent research area in computer science whose aim is to automatically provide information from electronic textual resources. It combines statistical, linguistic and machine learning techniques to mine unstructured data as text documents. In this work, an overview of text mining techniques is given, together with a general presentation of possible applications. Moreover, an information extraction system is built to mine diagnostic medical reports. The process consists of recognizing specific elements that indicate a myocardial infarction diagnosis, using a machine learning approach based on SVM algorithmes. Results are presented, focusing on the influence of a priori linguistic knowledge used to define the machine learning system.

Sommario

Il text mining è un ambito di ricerca relativamente recente, che si pone l'obiettivo di ricavare informazioni automaticamente da risorse testuali digitali. Esso si basa su tecniche statistiche, linguistiche e di machine learning per analizzare dati non strutturati come i documenti di testo. In questo lavoro di tesi viene presentata una panoramica sulle principali tecniche di text mining, con un accenno alle possibili applicazioni. In seguito, è realizzato un sistema per l'estrazione automatica di informazioni da lettere di dimissione ospedaliera, che consiste nel riconoscimento di specifici elementi diagnostici tipici dell'infarto miocardico acuto. Il sistema è realizzato con un approccio di tipo machine learning, basato su algoritmi SVM. Se ne presentano i risultati, con particolare attenzione all'influenza della conoscenza a priori linguistica usata per definire il sistema di machine learning.

Indice

Introduzione

L'idea di applicare tecniche di data mining su documenti scritti risale ai primi anni sessanta, quando comparirono i primi studi applicati alla ricerca di documenti all'interno di vaste collezioni. Nel corso degli anni, le tecnologie informatiche hanno reso facilmente disponibile un'enorme quantità di documenti di testo in formato elettronico. Tale quantità non solo ha favorito la ricerca, ma ha reso sempre più necessario l'utilizzo di tecniche automatiche di estrazione o catalogazione dell'informazione presente nei testi scritti. Basti pensare ad internet: la quantità di testi disponibili è infatti esorbitante, e le informazioni cui l'utente può avere accesso sarebbero ingestibili, quindi inutili, senza metodi efficienti di ricerca e catalogazione automatica. L'insieme delle tecniche statistiche applicate ai testi scritti prende il nome di *Text Mining*. Molti modelli linguistici, in particolare legati all'ambito della linguistica computazionale, supportano il text mining fornendo strumenti per la comprensione automatica del linguaggio naturale. L'ambito di studio di questi modelli è detto *Natural Language Processing*.

Le applicazioni legate al text mining sono innumerevoli. Un esempio particolarmente significativo è il riconoscimento automatico delle e-mail indesiderate, il cosiddetto *filtro antispam*, fornito da tutti i provider di posta elettronica. Un'altra applicazione possibile legata al web è la catalogazione automatica delle pagine in base agli argomenti trattati. Possiamo considerare anche problemi più complessi, come ad esempio l'analisi di una serie di notizie fornite da un'agenzia di informazione. Potrebbe interessarci, ad esempio, individuare i concetti chiave riportati negli articoli pubblicati in relazione ad un certo soggetto, oppure analizzare dei trend temporali (ad esempio l'andamento di alcuni settori di mercato).

Anche problemi di carattere scientifico possono essere risolti con tecniche simili. Si pensi ad esempio alla necessità di suddividere una grande collezione di articoli scientifici nel campo della biologia, in base alle diverse famiglie di proteine. Volendo andare oltre, è possibile immaginare applicazioni per

studiare automaticamente le relazioni tra due proteine. Poiché le risorse disponibili in letteratura sono tantissime, troppo sarebbe il tempo necessario ad un ricercatore per leggere tutti gli articoli che parlano simultaneamente di due date proteine. Un sistema in grado di individuare automaticamente in un articolo il tipo di relazione descritto tra le due proteine in causa costituirebbe quindi un supporto di grande utilità alla ricerca in campo biologico.

A differenza del data mining, che comprende tutte le tecniche di analisi e interpretazione di dati quantitativi, derivanti generalmente da osservazioni in ambito scientifico o economico, il text mining comprende quindi tutte le tecniche di statistica e gli algoritmi per la classificazione, l'estrazione automatica di informazione ed il raggruppamento di documenti di testo. Il set di dati studiato sarà dunque una collezione di documenti. Spesso, il singolo documento sarà considerabile come una singola osservazione, ma questa non è la regola.

L'area di interesse è vasta: oltre alla teoria statistica ed alla componente algoritmica, il text mining si interessa ad una serie di tecniche per trasformare i documenti in un formato più adatto all'analisi. Non è infatti semplice trattare direttamente dati che sono, in sostanza, delle stringhe di caratteri. È perciò necessario introdurre dei metodi efficaci e rigorosi per descrivere i concetti presentati dal testo in termini di frequenze relative, sfruttando le tecniche e le applicazioni di Natural Language Processing per decodificare le informazioni. In seguito, è necessario introdurre opportuni vettori per descrivere i testi, nei quali ogni componente sarà riferita ad un singolo concetto (più spesso ad una singola parola). Grazie all'introduzione di tali vettori, si avrà a disposizione un potente strumento di modellizzazione che permetterà di confrontare documenti simili tra di loro, o di estrarre da essi l'informazione rilevante.

Le tecniche di text mining poggiano su un'analisi linguistica preliminare che risulta di vitale importanza. Più la conoscenza linguistica sarà approfondita, più le applicazioni costruite saranno efficaci. Tuttavia, poiché il linguaggio varia sensibilmente al variare dell'argomento trattato, un'analisi linguistica approfondita sarà funzionale solo ad uno specifico ambito. In generale, più le applicazioni di text mining svolgono analisi approfondite, più è necessario un background linguistico specifico, più l'ambito di utilizzo sarà limitato. È dunque di fondamentale importanza essere in grado di integrare conoscenze linguistiche alle tecniche statistiche.

In questa tesi, viene data in un primo momento una panoramica sul text mining e sulla costruzione di un set di dati quantitativi a partire da testi scritti (Capitolo 1). In seguito, si descrivono più in dettaglio le principali tecniche statistiche usate nell'ambito del text mining: la classificazione supervisionata (Capitolo 2) e l'analisi cluster (Capitolo 3). Nel Capitolo 4 sono invece presentate le problematiche relative all'estrazione automatica di informazione da documenti di testo. Il Capitolo 5 presenta le SVM, tecniche di classificazione supervisionata tra le più usate per l'estrazione di informazioni.
Le tecniche descritte verranno poi impiegate in un'applicazione pratica, oggetto della tesi: il riconoscimento di elementi diagnostici legati all'infarto miocardico acuto in lettere di dimissione ospedaliera. I dati, insieme ad alcuni dettagli sulle tecniche usate per analizzarli, vengono presentati nel Capitolo 6 ed i risultati ottenuti sono infine discussi nel Capitolo 7. In appendice si riporta invece una brevissima introduzione sui modelli statistici di linguaggio (Appendice A), seguita da un tutorial di presentazione al software usato per la trattazione dei testi: GATE (Appendice B). Infine, presentiamo i listati di alcuni codici usati per questa tesi insieme ad alcuni codici di esempio nell'Appendice C.
Prima di iniziare la trattazione sul text mining, viene presentata una breve descrizione del progetto IMA, nel cui ambito si situa il presente lavoro.

Progetto IMA

Il progetto IMA (Infarto Miocardico Acuto) rientra in una serie di iniziative che la Regione Lombardia sta attuando per il miglioramento del servizio sanitario. In particolare, si punta ad introdurre nuove tecniche per migliorare la valutazione dei servizi e per ottimizzare le ingenti risorse immesse nel sistema sanitario. In questo senso, il progetto IMA si focalizza sul trattamento dei pazienti affetti da infarto miocardico acuto.

I sistemi di informazione sanitari in Italia si differenziano da regione a regione. Tuttavia, in molti casi esistono dei database composti da testi in formato elettronico (rapporti medici, lettere di dimissione ospedaliera, ecc...). Per sfruttare le importanti informazioni contenute in tali risorse, si rende necessario l'impiego di tecniche di text mining. In particolare, si punta al raggiungimento di tre principali obiettivi:

1. l'introduzione di nuovi indicatori di qualità, che tengano conto delle informazioni contenute nei rapporti scritti;

2. la validazione delle informazioni, relative a diagnosi e trattamenti, contenute in appositi database e codificate seguendo delle procedure non sufficientemente standardizzate;
3. la valutazione della continuità assistenziale, analizzando successioni temporali di rapporti relativi a singoli pazienti, oppure incrociando le informazioni ricavate dai testi scritti con quelle contenute in altri database (ad esempio relative alle prescrizioni di farmaci).

Nello specifico, il presente lavoro si focalizza sul secondo degli obiettivi citati. Analizzando le lettere di dimissione ospedaliera redatte dai medici al momento del rilascio di un paziente, è possibile riconoscere alcuni dettagli relativi alla diagnosi. D'altro lato, esistono già database, compilati dal personale amministrativo, che riportano le diagnosi relative ad ogni ricovero. In questi database, le informazioni sono codificate attraverso dei sistemi standard (ICD-9-CM e ICD-10-CM). Queste informazioni potrebbero quindi essere integrate con quelle più dettagliate estratte dai database testuali, in supporto di eventuali strumenti di controllo e valutazione.
Il progetto rappresenta quindi un banco di prova, per verificare le potenzialità del text mining nel ricavare informazioni dalle banche dati testuali, più ricche rispetto a quelle presenti nei database strutturati.

I testi analizzati sono stati gentilmente messi a disposizione dell'autore (mantenendo la proprietà e la riservatezza del dato) dal Dott. Pietro Barbieri, cardiologo e dirigente medico dell'Azienda Ospedaliera *Ospedale di Circolo di Melegnano*. Il trattamento dei dati è avvenuto sotto la supervisione ed in collaborazione con il Dott. Stefano Ballerio, docente di Letterature Comparate all'Università degli Studi di Milano e responsabile del lavoro di linguistica sulle lettere di dimissione, nella sede ospedaliera di Cernusco sul Naviglio (ospedale Uboldo).

Capitolo 1

Panoramica sul Text Mining e costruzione del data set

Questo capitolo riporta una descrizione generale delle tecniche e delle problematiche legate al text mining e si danno alcuni cenni storici sullo sviluppo della ricerca in tale ambito. In seguito si procede con una trattazione più approfondita, descrivendo le tecniche di preprocessing necessarie alla costruzione di un set di dati quantitativi a partire dai testi scritti.

Il *text mining* può essere definito come l'*analisi delle informazioni contenute in documenti scritti attraverso i metodi di data mining*. Il tipo di informazioni analizzate può variare, e si va da sistemi che classificano automaticamente i documenti in base al contenuto, alla ricerca di documenti contenenti informazioni specifiche, fino ad arrivare ad applicazioni che ritrovano un'informazione puntuale richiesta da un utente, permettendo di catalogarla e descriverla in forma strutturata. Gli ambiti di studio legati al text mining ricoprono diverse problematiche, compresi i metodi statistici in uso, una parte algoritmica ed un'importante parte di linguistica.
Volendo elencare i problemi chiave oggetto della ricerca sul text mining, si ha:

- la costruzione di un set di dati a partire da informazioni non strutturate presenti nel testo scritto;
- la classificazione supervisionata di documenti in base al contenuto;
- la ricerca di informazioni sulla base di specifiche richieste da parte di un utente;
- la classificazione non supervisionata e l'organizzazione di documenti;
- l'estrazione automatica di informazioni;
- la scelta degli algoritmi e la loro valutazione.

Queste problematiche sono generalmente trattate in tutta la letteratura riguardante il Text Mining, e l'esposizione in questo lavoro prenderà spunto principalmente da [7] e [29]. Per ulteriori approfondimenti sul text mining e per una rassegna di studi di caso, si consiglia la lettura di [11] e di [27]. Se invece si vogliono approfondire gli aspetti linguistici e di linguistica computazionale, può essere utile la lettura di [18] e di [3]. Prima di iniziare, è interessante presentare un breve accenno sulla storia del Text Mining, ripercorrendo le tappe fondamentali della ricerca.

1 Cenni storici

Gli studi alla base del text mining iniziano negli anni '60, per poi svilupparsi lentamente ed esplodere in tempi recenti, grazie all'enorme disponibilità di testi in formato elettronico reperibili negli archivi e nel web. Le tappe fondamentali della ricerca in tale ambito possono essere così sintetizzate:

Anni '60 I primi studi degli anni '60 si focalizzano sull'indicizzazione e la ricerca dei documenti. Da un lato si trattano problemi analoghi alla classificazione che prevedono, dato un insieme di categorie, di assegnare automaticamente una categoria ad un dato documento. Dall'altro, ci si interessa al problema di ricercare i documenti che trattano un determinato argomento all'interno di grandi collezioni. Gli articoli di riferimento sono [17], [19] e [25].

Anni '70 Negli anni '70 si inizia a diffondere l'idea di una descrizione matematica dei testi scritti. Questa descrizione permette di compiere i primi studi sull'analisi cluster applicata ai documenti. Si affermano quindi diverse definizioni di similarità tra testi scritti, che permettono l'implementazione dei principali algoritmi di data mining. Citiamo in particolare gli articoli di Salton et al. [26], e di Jardine e Van Rijsbergen [10].

Anni '80 - '90 Successivamente, la diffusione e la popolarità degli studi nell'ambito dell'intelligenza artificiale coinvolge anche il text mining e, più in generale, l'ambito linguistico. Molteplici algoritmi tipicamente trattati dai testi di intelligenza artificiale vengono applicati, ad esempio, per l'assegnazione di documenti a categorie [9] , ma anche per trattare problemi più legati alla linguistica come l'estrazione di prefissi [21] o l'analisi logica o del periodo [23]. La linguistica computazionale inoltre vede un grande sviluppo, e di conseguenza le potenzialità del text mining aumentano. Basandosi su

strumenti linguistici più efficaci, maggiore è l'informazione che si riesce ad analizzare.

Anni '90 - oggi Gli ultimi anni hanno visto il moltiplicarsi degli studi e delle applicazioni di text mining. La diffusione di internet e dell'informatica ha reso disponibile un'ingente quantità di testi in formato elettronico. Uno dei problemi principali negli anni '60 - '70 era infatti la scarsa disponibilità di set di dati, legata alle difficoltà nella codifica digitale dei testi. Non solo: la quantità di documenti scritti accessibili da chiunque è spropositata. È perciò necessario disporre di validi strumenti per trattare le informazioni disponibili nei testi rendendole effettivamente sfruttabili. Di qui la necessità di sviluppare motori di ricerca efficaci, filtri per mettere in risalto i documenti rilevanti, sistemi per rintracciare le informazioni principali presenti in un documento e applicazioni di supporto ai servizi clienti.
Come spesso accade, la spinta iniziale alla ricerca nel settore è data soprattutto da finanziamenti pubblici, spesso legati a progetti militari. In questo modo sono state indette numerose conferenze, ciascuna rivolta ad un tema specifico. Le più rilevanti sono la MUC (*Message Understanding Conference)*, la TREC (*Text REtrieval Converence*), la CoNLL (*Conference on Natural Language Learning*) e la ACE (*Automatic Context Extraction*). Ogni conferenza pone un problema specifico, che i ricercatori partecipanti affrontano autonomamente, cercando di proporre tecniche di risoluzione efficaci e innovative.

2 Preparazione dei dati

Ogni analisi di data mining necessita una fase di preparazione dei dati. Tale fase risulta ancora più importante nel caso del text mining. Non solo è necessario disporre di documenti in un formato standard, ma un trattamento preliminare più approfondito dei testi porterà ad una maggiore efficienza degli algoritmi usati, e renderà possibile analisi più approfondite. Vediamo dunque in questa sezione le principali fasi che caratterizzano la preparazione e la costruzione del data set, inteso come collezione di documenti.

2.1 Formato dei documenti

A seconda del software utilizzato, diversi sono i formati accettati per i documenti in input. In generale, un testo senza formattazione (estensione `.txt`) è sempre utilizzabile. Tuttavia, spesso l'informazione contenuta nella formattazione e nella struttura del testo potrebbe risultare importante. Ad

esempio, la stringa di caratteri che compone il titolo sarà molto importante per individuare l'argomento trattato dal testo, così come il campo *oggetto* in una e-mail. Parole evidenziate in grassetto possono indicare concetti chiave, così come spesso possono essere utili le informazioni riguardanti la data e l'autore. Fra i vari formati disponibili, si è diffuso l'utilizzo dell'XML come standard per il text mining. Ad esempio, la piattaforma GATE[1], di cui si parlerà nel seguito, si basa su testi XML, così come il pacchetto `tm`[2] di R[3] accetta l'XML come formato standard.
L'XML si basa sulla divisione del testo in campi, che possono dare indicazioni sul contenuto (ad esempio i campi *titolo*, *autore*, *sommario*, ecc.) oppure sul formato del carattere (stile, dimensione, ecc.). I campi sono definiti tramite delle *tags* di inizio e fine, ad esempio:

`<DOC> ... </DOC>` : indica l'inizio e la fine di un documento

`<AUTOR> ... </AUTOR>` : individua il nome dell'autore

`<b><i> ... </i></b>` : indica una parte di testo in corsivo e grassetto

In buona parte, la scelta è dovuta al fatto che quasi tutti gli editor permettono l'esportazione o la conversione automatica del testo in XML, e lo stesso vale per l'esportazione di e-mail (si pensi a Thunderbird o Outlook) o di pagine web. Inoltre, a differenza di un formato *solo testo*, l'XML permette di conservare la struttura del testo, rimuovendo l'informazione relativa alla formattazione, che spesso risulta inutile ai fini di un'analisi di text mining.
Ma l'uso dell'XML va oltre il semplice formato di input. Solitamente, i software di Natural Language Processing forniscono informazioni in output aggiungendo nel testo delle tags XML. Ad esempio, un'applicazione per l'analisi logica potrà indicare soggetto e complemento oggetto tramite opportune tags.

3 Preprocessing

Oltre alla preparazione e conversione dei documenti in un formato standard, possono essere necessarie varie fasi di rielaborazione dei testi, che coinvolgono strumenti linguistici e di Natural Language Processing a volte molto sofisticati. Non tutte queste fasi sono però necessarie ad un'analisi di text mining, al contrario, alcune potrebbero fornire informazioni inutili a secon-

[1] per una descrizione di GATE si rimanda all'appendice o alla guida [8]
[2] si rimanda a [5] e [6] per una descrizione del pacchetto
[3] per maggiori informazioni sul software statistico R si veda [22]

da del tipo di analisi. Sono presentati quindi in un primo momento tutti gli strumenti linguistici che possono intervenire nelle fasi di preprocessing ed in seguito, nella trattazione delle tecniche specifiche, saranno richiamate solo le fasi importanti per il caso in esame.
Le fasi principali di preprocessing sono le seguenti[4]:

- Tokenization
- Stemming
- Separazione di frasi
- Part of speech (POS) tagging
- Riconoscimento di sintagmi (phrase recognition)
- Named entity recognition
- Parsing

Di seguito, ognuna di queste fasi è presentata a grandi linee. Va precisato che in molti casi, gli strumenti linguistici adottati si basano a loro volta su modelli statistici, e richiedono un notevole sforzo di modellazione alla base. Fortunatamente, i software disponibili implementano già molte di queste fasi con risultati più che accettabili. Tuttavia, la maggior parte delle applicazioni è stata sviluppata per la lingua inglese, e molti sviluppi sono stati fatti di recente per tedesco, spagnolo, giapponese e cinese. Gli strumenti linguistici equivalenti per l'italiano sono ancora in buona parte non disponibili.

3.1 Tokenization

Il primo passo da compiere è apparentemente banale: data una stringa di caratteri, deve essere suddivisa in parole o, più precisamente, in *token* diversi. I token sono separati l'uno dall'altro da appositi caratteri, i *separatori*, che possono essere spazi bianchi, tabulazioni, terminatori di paragrafo o elementi di punteggiatura. Alcuni elementi di punteggiatura possono inoltre essere considerati come token, ad esempio punti interrogativi o esclamativi se in seguito vogliamo riconoscere domande o esclamazioni.
Le difficoltà insorgono nel caso in cui un elemento di punteggiatura non sia usato come separatore, ma rientri all'interno di un token. Si pensi ad esempio a molti cognomi italiani che si scrivono con un apostrofo (*D'Angelo*, *D'Amico*) o alla negazione inglese *don't*, oppure si pensi a nomi propri composti da più parole, per i quali si dovrebbe tener conto di un solo token (*New York*). Altri casi delicati da gestire sono i tratti orizzontali. Possono infatti essere dei separatori, dei caratteri di un nome composto oppure possono indicare

[4]si adotta spesso la terminologia inglese, per una più facile corrispondenza con la letteratura.

un segno meno se precedono un numero. I casi come quelli citati sono molti. Queste difficoltà sono risolte generalmente stabilendo delle regole, che si basano sulla presenza di caratteri maiuscoli/minuscoli, applicabili in generale a tutti i casi e codificate nei sistemi di Tokenization. Avere a disposizione dei dizionari (nel nostro caso semplicemente elenchi di parole) può aiutare a individuare token specifici.

3.2 Stemming

Una volta riconosciuti i token, è importante trasformarli in una forma standard. In molte applicazioni, infatti, l'analisi si basa sul conteggio di quante volte una parola compare nel documento. Se la stessa parola è presente, ad esempio, tre volte in forme diverse (es. singolare/plurale, iniziale maiuscola/minuscola), è opportuno che le tre forme siano contate come tre istanze della stessa parola. Innanzi tutto, se si lavora con sistemi case-sensitive, è buona norma adottare uno standard per l'utilizzo delle maiuscole, in modo che un termine maiuscolo di inizio frase non sia differenziato dallo stesso termine con iniziale minuscola a metà frase. In seguito, si può agire "normalizzando" nomi comuni, aggettivi e verbi. Questo procedimento può essere portato ad uno stadio più avanzato, in cui si opera una riduzione dei termini ancora più profonda.
In generale, si possono avere due tipi di stemming:

- **Inflectional stemming o lemmatizzazione:** riguarda la normalizzazione di declinazione e coniugazione delle parole. Come già accennato, lo scopo è fare in modo che le differenti forme (singolare, plurale, passato, presente, ecc.) in cui un termine è presente siano contate come istanze di un unico termine. La forma normalizzata è detta *lemma* (ad esempio, il lemma *andare* per le parole *vado* o *andiamo*).

- **Root stemming:** è un procedimento che va più in profondità, mirato ad eliminare prefissi e suffissi delle parole. In questo caso si risale non solo da forme diverse (*vado*, *andiamo*) ad un'unica parola (il lemma *andare*), ma anche da forme diverse di lemmi diversi (*andiamo*, *andamento*) ad un'unica radice (*and-*), detta anche *morfema lessicale*. In questo modo, alla fine della fase di stemming resteranno molti meno termini diversi, mettendo meglio in evidenza le informazioni riguardanti i temi trattati dal documento.

Come è facile immaginare, la lingua influenza molto la necessità della fase di stemming. L'italiano ad esempio (così come lo spagnolo o il francese) presenta una grande ricchezza per coniugazioni e declinazioni rispetto alle

lingue anglosassoni. Lo stesso lemma corrisponderà dunque a molte forme, e la normalizzazione data dall'inflectional stemming permetterà una grande riduzione dei termini osservati.
Va inoltre precisato che la fase di stemming non è indispensabile per tutte le applicazioni di text mining. Tuttavia, essa è auspicabile qualora si vogliano classificare documenti o estrarre da essi informazioni molto generali.

3.3 Separazione di frasi

Come la procedura di tokenization prevede la divisione di una stringa di caratteri in token, questa fase prevede la suddivisione del testo in frasi. Chiaramente, il riconoscimento delle frasi è necessario per un trattamento del testo più approfondito rispetto ad un semplice conteggio delle parole presenti in un documento. Diventa perciò utile quando si vogliono estrarre dal testo informazioni precise, contenute in singole frasi. Inoltre, nell'ottica di sistemi in grado di comprendere il linguaggio naturale scritto, la suddivisione del testo in frasi può rappresentare un primo passo.
Il problema della separazione del testo in frasi consiste sostanzialmente nella classificazione degli elementi di punteggiatura. Di solito, le difficoltà entrano in gioco con i punti fermi "." quando vi sono delle abbreviazioni (ad esempio *ecc.* o *sig.*). Poiché il problema si riconduce ad una classificazione binaria, è possibile ricorrere a classificatori supervisionati, oltre che a procedure basate su regole linguistiche. I classificatori saranno allora basati sull'osservazione di token che precedono e seguono il punto.

3.4 Part of speech (POS) tagging

Se lo scopo dell'analisi è l'estrazione di informazioni dettagliate, è opportuno operare un'analisi grammaticale dei token. Dopo aver diviso il testo in frasi, si può quindi procedere con un'assegnazione del ruolo grammaticale (*part of speech*), attribuendo ad ogni token il ruolo di nome, verbo, aggettivo, avverbio ecc.
Naturalmente, per questa fase di preprocessing l'apporto linguistico è fondamentale. Il riconoscimento del ruolo grammaticale di un termine si basa in gran parte sulla costruzione (manuale) di dizionari, detti proprio *grammatiche*, che riportano un elenco di termini associati ai ruoli. Esistono però molti casi ambigui. In tal caso, è possibile costruire degli appositi classificatori supervisionati, ancora una volta basati sui token presenti in un intorno del termine di interesse. La difficoltà maggiore risiede nella scarsità di testi già annotati che fungano da training set.

3.5 Riconoscimento di sintagmi (phrase recognition)

A differenza della separazione di frasi (*sentences*), la fase di *phrase recognition* mira al riconoscimento di sintagmi elementari. In modo semplicistico, possiamo definire il sintagma come un elemento della frase che rappresenta un'unità di significato. Ad esempio, nella frase *il mio cane corre nel giardino*, *il mio cane* è un sintagma nominale (*noun phrase*), *corre* è un sintagma verbale (*verb phrase*) e *nel giardino* è un sintagma preposizionale (*preposition phrase*).
La phrase recognition è perciò utile sia come primo step per l'analisi del periodo, che per individuare relazioni semplici tra elementi nel testo, espresse da frasi elementari.

3.6 Named entity recognition

Per sfruttare le informazioni derivanti dalla phrase recognition, e per permettere una comprensione più precisa del testo, è necessario operare una fase detta *named entity recognition*. Le *named entity* sono elementi noti che apportano molta informazione. Solitamente, si vogliono riconoscere nomi di luoghi, di aziende, espressioni temporali come date ed ore e quantità interessanti (ad esempio, se si è interessati ad un'analisi di text mining per monitorare l'andamento di un certo indicatore economico, sarà indispensabile riconoscere le cifre che si riferiscono a tale indicatore).
Anche in questo caso, il lavoro principale consiste nella redazione di dizionari specifici, che riportino tutti i termini di interesse, ma anche nella costruzione di specifici metodi di machine learning. Parleremo più in dettaglio della *named entity recognition* nel Capitolo 4.

3.7 Parsing

Il *parsing* consiste nell'analisi delle relazioni tra i token di una frase, e nell'assegnazione di un ruolo semantico. L'output del parsing è solitamente un albero, i cui i nodi rappresentano le relazioni e le cui foglie sono i token. Grazie a tale albero, è possibile ricostruire buona parte dell'analisi logica di una frase ed estrarne informazioni approfondite, che comprendono le relazioni tra i diversi elementi. Avendo già operato la fase di named entity recognition, è perciò possibile individuare le relazioni di interesse tra le named enities ritrovate. In tal modo, si possono costruire sistemi in grado di riconoscere nel testo fatti specifici. Si pensi ad esempio ad un'applicazione che riconosca, analizzando una collezione di articoli di giornali economici, le acquisizioni di società. I nomi delle società rientreranno nella lista delle

named entity di interesse, l'acquisizione sarà indicata dalla presenza di verbi specifici ed i ruoli nell'acquisizione saranno riconosciuti grazie al parsing.
Esistono diverse applicazioni in grado di generare l'albero di parsing, in particolare rivolte a testi in lingua inglese, ma adattabili anche ad altre lingue (tra cui tedesco, cinese e arabo). Citiamo in particolare lo *Stanford Parser*[5].

4 Costruzione del data set

La creazione di vettori di feature a partire dai documenti si può fare in diversi modi. Essa dipende principalmente dal tipo di analisi che si vuole operare sui documenti e dalle fasi di preprocessing eseguite. Immaginiamo di voler condurre un'analisi in cui l'unità osservata è il documento, senza focalizzarci sulle diverse informazioni apportate dalle diverse frasi. Tipicamente, i problemi affrontati riguardano la categorizzazione o classificazione dei documenti, o un'esplorazione di una collezione di documenti tramite cluster analysis. È naturale aspettarsi che, in tal caso, verrà costruito un vettore di feature per ogni documento. Vediamo questa operazione più in dettaglio.

Per analizzare l'argomento trattato dal documento, ci focalizziamo sui termini usati. Le fasi di preprocessing necessarie sono perciò quella di *tokenization* e *stemming*. Più lo stemming sarà operato in profondità, migliori saranno i risultati dell'analisi.

4.1 Scelta delle feature

Come detto, ogni documento corrisponde ad un'osservazione, quindi ad un vettore del data set. Ciò che si osserva nei documenti sono le parole, più precisamente i token individuati nel preprocessing. Un vettore di feature è dunque costruito in modo che ogni componente corrisponda ad un token. In particolare, una componente singola del vettore prenderà valori diversi da zero se il token corrispondente è presente nel documento.

Ogni vettore sarà perciò composto da p componenti, dove p è il numero di token ritrovati in tutta la collezione di testi analizzati. Delle p componenti, solo poche saranno diverse da zero in un singolo vettore, poiché un documento riporta solo un numero limitato di parole, rispetto a tutte quelle ritrovate nella collezione. I vettori saranno perciò sparsi, e sarà necessario,

[5] disponibile sul web su `http://nlp.stanford.edu/software/lex-parser.shtml`

nella fase di implementazione, disporre di buoni algoritmi per la gestione ottimale di vettori sparsi.
La sparsità dei vettori non è il solo problema. In generale, è facile aspettarsi che p sia molto grande. Il linguaggio naturale è infatti tale da richiedere, anche in ambiti specifici, l'impiego di un ampio vocabolario. Se allora n è il numero di documenti osservati (e quindi di vettori costruiti), saremo nel caso delicato in cui

$$p \gg n$$

4.2 Pesi assegnati alle feature

Restano da definire i valori assegnati alle componenti non nulle dei vettori. Introduciamo prima alcune notazioni. Dato un token j, indichiamo con $df(j)$ la *document frequency* relativa a j, ovvero

$$df(j) := \#\text{ documenti contenenti il token } j \tag{1.1}$$

Se invece consideriamo il documento i ed il token j, possiamo contare quante volte j compare all'interno del documento i, definendo allora la *term frequency*, indicata con $tf(i,j)$:

$$tf(i,j) := \#\text{ occorrenze del token } j \text{ nel documento } i$$

Possiamo inoltre definire un coefficiente che indichi la rilevanza di un dato token. Immaginiamo di voler suddividere i documenti di una collezione in base all'argomento, e che si debba scegliere tra l'argomento A e l'argomento B. Se ritroviamo il token j indistintamente in tutti i documenti, esso rappresenterà una parola adottata comunemente sia nella trattazione di A che di B. La presenza di j non darà quindi informazioni utili alla classificazione del documento. Seguendo quest'idea, introduciamo allora l'indice detto *inverse document frequency*, indicato con $idf(j)$ e definito come

$$idf(j) := \log \frac{n}{df(j)}$$

In generale, si assegnano tre tipi di pesi alle componenti dei vettori. Denotiamo w_{ij} la componente j-esima (riferita al token j) del vettore costruito con l'i-esimo documento. Possiamo allora avere:

- **pesi binari:**

$$w_{ij} = \begin{cases} 1 & \text{se } j \text{ compare nel documento } i \\ 0 & \text{altrimenti} \end{cases} \tag{1.2}$$

- **term frequency:** corrisponde al conteggio delle occorrenze del token j nel documento i

$$w_{ij} = tf(i,j) \tag{1.3}$$

- **term frequency normalizzate:** si normalizza la *term frequency* moltiplicandola per il coefficiente, introdotto in precedenza, di *inverse document frequency* riferito al token j:

$$w_{ij} = \text{tf.idf}(i,j) := tf(i,j) \times idf(j) \tag{1.4}$$

La scelta dei pesi (1.4) o (1.3) fornisce maggiori informazioni rispetto ad una descrizione semplicemente binaria (1.2). Va anche ricordato che esistono casi in cui si scelgono pesi ternari, dove w_{ij} assume valore 0, 1 o 2 a seconda che il token j sia rispettivamente assente, presente un'unica volta o presente ripetute volte nel documento i. In ogni caso, le possibilità di assegnazione dei pesi w_{ij} sono moltissime, e si è scelto di citare solamente le più usate in letteratura.

4.3 Riduzione di p

Come già è stato già detto, le analisi di text mining si situano nel caso in cui la quantità di feature osservate è molto maggiore del numero di osservazioni (documenti), cioè $p \gg n$. Risultano perciò di fondamentale importanza metodi in grado di ridurre p.
Un primo passo per ridurre p consiste nella fase di *stemming* durante il preprocessing. Operando un *root stemming*, le parole sono private di prefissi e suffissi e l'insieme dei diversi token osservati diminuisce notevolmente. Al di là del preprocessing, esistono comunque diversi metodi per costruire i vettori di feature, che possono ridurne o meno le dimensioni.

Modello *bag of words* Il più comunemente usato, oltre che il più semplice da implementare, consiste nel considerare tutti i token presenti nel documento (eventualmente i lemmi o i morfemi lessicali se si è operata una fase di inflectional o root stemming). Ogni token osservato corrisponderà dunque ad una feature del vettore. In questo caso la dimensione p dei vettori dipenderà solo dalla fase di preprocessing.

Dizionari di sinonimi Oltre alla fase di stemming, si possono processare ulteriormente i documenti uniformando tutti i termini sinonimi. Questa operazione è possibile solamente in presenza di un dizionario di sinonimi, frutto di un lavoro linguistico specifico per il dominio trattato. Qualora si disponga di buone risorse linguistiche, è possibile quindi ridurre sensibilmente la quantità di diversi token osservati. Tuttavia, raramente si dispone di tali risorse, e risulta necessario adottare altre tecniche per ridurre p.

Uso di indici di rilevanza dei token Indipendentemente dal preprocessing o dall'uso di dizionari, è possibile selezionare solamente i token interessanti rispetto all'analisi che si vuole condurre. Si devono perciò introdurre degli indici che diano un'indicazione sulla rilevanza dei token. Un indice spesso usato nel caso della classificazione supervisionata è l'*information gain*.

4.4 Information gain

Se siamo interessati ad un problema di classificazione supervisionata, è possibile selezionare le feature rilevanti in relazione al training set. La rilevanza della feature j-esima è espressa tramite l'indice *information gain*, indicato come $IG(j)$. Calcolando l'*information gain* di tutte le feature, è possibile selezionare solamente quelle con indice maggiore, riducendo notevolmente p. L'*information gain* esprime quantitativamente il guadagno di informazione che si ha, conoscendo la presenza del termine j in un documento.
Immaginiamo quindi di voler operare una classificazione supervisionata, assegnando ogni documento osservato ad uno tra G gruppi. Supponiamo ora che sia dato un training set, e indichiamo con P_g la proporzione di elementi del training set appartenenti al gruppo g, con $g \in \{1, ..., G\}$. Indichiamo allora L_{label} la misura di entropia definita come:

$$L_{label} := \sum_{g=1}^{G} P_g \log_G \left(\frac{1}{P_g}\right)$$

Consideriamo ora uno specifico token j. Dividiamo il training set nei sottoinsiemi $R_j := \{$ documenti del training set che contengono il token $j \}$ e $S_j := \{$ documenti del training set che non contengono il token $j \}$.
All'interno di questi due insiemi, consideriamo le proporzioni P_j, $P^j_{g,R}$ e $P^j_{g,S}$, che indicano:

- P_j: proporzione degli elementi del training set contenenti il token j

$$P_j := \frac{\#\text{ documenti del training set contenenti } j}{\#\text{ documenti del training set}} \tag{1.5}$$

- $P^j_{g,R}$: proporzione degli elementi di R_j appartenenti al gruppo g

$$P^j_{g,R} := \frac{\#\text{ documenti del training set contenenti } j \text{ nel gruppo } g}{\#\text{ documenti del training set contenenti } j} \quad (1.6)$$

- $P^j_{g,S}$: proporzione degli elementi di S_j appartenenti al gruppo g

$$P^j_{g,S} := \frac{\#\text{ documenti del training set non contenenti } j \text{ nel gruppo } g}{\#\text{ documenti del training set non contenenti } j} \quad (1.7)$$

Definiamo allora L_j come:

$$L_j := P_j \sum_{g=1}^{G} P^j_{g,R} \log_G \left(\frac{1}{P^j_{g,R}}\right) + (1 - P_j) \sum_{g=1}^{G} P^j_{g,S} \log_G \left(\frac{1}{P^j_{g,S}}\right)$$

L_j è tanto più grande quanto più i documenti del training set contenenti j sono distribuiti in modo uguale tra i vari gruppi. Se invece i documenti contenenti j appartengono ad un solo gruppo, L_j sarà minimo. Possiamo allora definire l'indice $IG(j)$ come:

$$IG(j) := L_{label} - L_j$$

In questo modo, più l'*information gain* di j è grande, più j dà indicazioni sull'appartenenza di un documento ad un dato gruppo. Per ridurre p si possono dunque scegliere solo le feature per cui IG è maggiore di un'opportuna soglia.

4.5 Multiword feature

Generalmente, i vettori sono costruiti in modo che ogni feature corrisponda ad una parola o, più precisamente, ad un token. È possibile però avere dei casi in cui una feature corrisponde ad una sequenza di parole. Essa è detta *multiword feature*, ed esistono diversi modi per generarla.

- **Preprocessing:** operando la fase di *named entity reognition*, è possibile raggruppare token separati che si riferiscono alla stessa entità. In generale, in questo modo si prendono in considerazione nomi propri composti di luoghi, eventi o persone, come ad esempio *Stati Uniti*, *Grande Guerra* o *Michael Jackson*.

- **Uso di dizionari specifici:** oltre all'apporto linguistico necessario alla *named entity recognition*, si ricorre a volte all'uso di dizionari supplementari, che riportano elenchi di ulteriori nomi composti o espressioni comuni.
- **Correlazione tra gruppi di parole:** è possibile generare delle *multiword feature* anche senza il supporto di dizionari, calcolando la correlazione tra gruppi di parole. Tale correlazione è espressa dalla *misura di associazione* AM. Si possono allora aggiungere come multiwords feature quei gruppi di parole che hanno misura di associazione superiore ad una certa soglia.

Misura di associazione

Sia data una sequenza di parole, indicata con T, composta da una quantità $size(T)$ di parole. Se $freq(T)$ è il numero di volte che T compare nella collezione di documenti e $freq(w)$ il numero di volte che la parola w compare nella collezione di documenti, allora:

$$AM(T) := \frac{size(T) freq(T) \log_{10}(freq(T))}{\sum_{w_i \in T} freq(w_i)}$$

Più una sequenza di parole T è frequente nel testo, più il suo indice $AM(T)$ è grande. Inoltre, le sequenze più lunghe avranno indice più alto (è meno probabile che sequenze lunghe si ripetano casualmente). Il termine a denominatore riduce invece l'indice nel caso in cui le parole della sequenza siano usate molto frequentemente e quindi apportino poche informazioni (la sequenza *come ad esempio* sarà chiaramente meno importante rispetto a *prodotto interno lordo*).
In genere, per il calcolo di AM si tiene conto di tutte le possibili sequenze composte da un numero compreso tra m_1 ed m_2 parole (sarebbe inutile considerare sequenze di ogni lunghezza). Per quanto opportune scelte di m_1 ed m_2 possano ridurre le sequenze considerate, il calcolo della misura di associazione resta molto pesante. È necessario quindi disporre di algoritmi efficienti e, al tempo stesso, di una collezione di documenti di dimensioni ridotte. L'utilizzo delle multiword feature può comunque apportare buoni miglioramenti nelle performance dei sistemi di text mining.

Capitolo 2

Classificazione di documenti e information retrieval

1 Classificazione supervisionata

La classificazione supervisionata è una delle tecniche più usate nell'ambito del text mining. Se ne possono trovare innumerevoli applicazioni, dai filtri *anti-spam* per la posta elettronica alla categorizzazione di documenti in base all'argomento trattato, fino ad arrivare alla ricerca di documenti contenenti informazioni rilevanti ed alla risposta ad alcune semplici domande relative al contenuto del documento (in tal caso le possibili risposte sono prefissate a l'assegnazione di un documento ad una categoria corrisponde a scegliere una risposta).

Definizione: data la collezione di documenti D ed un insieme di categorie C, possiamo definire un classificatore come un'applicazione $f : D \times C \rightarrow \{0, 1\}$.

Seguendo questa definizione, l'assegnazione delle categorie non è per forza univoca. Un documento potrebbe appartenere infatti contemporaneamente a più di un gruppo. Ciò è naturale nel caso della categorizzazione di documenti: lo stesso testo può infatti trattare più argomenti contemporaneamente. In tal caso, si costruisce per ogni argomento un classificatore binario, che opera indipendentemente dagli altri per l'assegnazione di una singola categoria.
La classificazione supervisionata applicata al text mining richiede particolare attenzione per alcuni aspetti. In primo luogo, i testi scritti analizzati sono quasi sempre in continua evoluzione. Un cambiamento nell'autore o una leggera variazione nel modo di trattare un argomento possono causare una modifica sostanziale dello stile dei documenti. È perciò di fondamentale

importanza mantenere il training set in costante aggiornamento, per evitare il degrado nel tempo delle performances dei classificatori.
In secondo luogo, si deve tenere presente che molti modelli si basano sull'ipotesi di indipendenza delle feature osservate. Ora, nel caso di un testo scritto, le feature osservate sono i token, generalmente parole, che difficilmente possono essere considerate indipendenti. Si osserveranno perciò dei risultati meno buoni di quanto ci si potrebbe aspettare da un'analisi di data mining in cui le feature sono indipendenti.

I metodi più impiegati per la costruzione dei classificatori nell'ambito del text mining sono i metodi deterministici basati sulla definizione di *decision rules*, il *knn* (*k nearest neighbors*), i classificatori detti *bayesiani* (anche se non ricorrono ai modelli propri della statistica bayesiana) ed i classificatori basati su modelli lineari.

1.1 Decision rules

Una *deision rule* consiste in un elenco di condizioni da verificare, per assegnare un documento ad un dato gruppo g. Dato un documento $\mathbf{x}$, la classificazione si fa semplicemente scorrendo tutte le decision rules prestabilite ed assegnando $\mathbf{x}$ ai gruppi per cui le condizioni sono soddisfatte.
Le condizioni si traducono generalmente nella verifica della presenza di opportuni token o combinazioni di token. Una decision rule può assumere ad esempio la forma seguente:

Se il documento $\mathbf{x}$ *contiene i token* w_1, w_2 e w_3 *e non contiene il token* w_4, *allora* $\mathbf{x}$ *è assegnato al gruppo* g.

La difficoltà legata a questo tipo di classificazione non è ovviamente di tipo implementativo. Per stabilire delle decision rules è necessaria una conoscenza linguistica approfonodita delle problematiche trattate dai testi. Perciò, se da un lato non serve disporre di un training set già suddiviso in gruppi, dall'altro si richiede il lavoro di uno o più esperti per scrivere delle regole che permettano di ottenere buoni risultati di classificazione.
È possibile costruire degli algoritmi in grado di affiancare il lavoro dei linguisti nella ricerca di pattern informativi. In tal caso, è necessario disporre di un training set annotato, all'interno del quale verranno ricercate delle successioni di parole ricorrenti associabili a un determinato gruppo. L'apporto di un esperto linguista servirà in seguito per la selezione dei pattern più coerenti.

Un grande vantaggio nell'uso delle decision rules è che le regole sono facilmente leggibili: anche un utente non esperto può ricavare informazioni utili dalla lettura delle regole. Inoltre, se si adottano algoritmi specifici per ritrovare le regole, il lavoro linguistico aggiuntivo per all'individuazione delle regole prive di significato sarebbe minimo. Infine, è sufficiente una rappresentazione dei documenti tramite pesi binari (1.2).
Accanto ai vantaggi, vi sono però notevoli difficoltà legate all'uso delle decision rules. In primo luogo, i metodi si rivelano spesso poco predittivi. In secondo luogo, gli eventuali algoritmi utilizzati per riconoscere automaticamente delle regole sono molto costosi, rendendo spesso inconveniente il loro impiego. Inoltre, qualora le decsion rules siano scritte "a mano" da esperti linguisti, vengono costruite per un determinato problema e non saranno applicabili all'infuori di esso. Il lavoro degli esperti sarebbe quindi difficilmente riadattabile per costruire delle applicazioni in ambiti diversi.

1.2 Similarità tra documenti

Prima di affrontare più in dettaglio la costruzione dei classificatori, è necessario definire una misura di *similarità* tra documenti. Esistono diverse possibilità, e di seguito sono presentate quelle più frequentemente usate. Si noti che la misura di similarità può essere sostituita dal concetto inverso di misura di distanza.

Token comuni Siano dati due documenti, descritti attraverso i vettori di feature $\mathbf{x}_1$ e $\mathbf{x}_2$. Supponiamo di aver operato una selta dei pesi binaria, come definito in (1.2). Il conteggio delle parole comuni si traduce nel calcolo del prodotto scalare tra i due vettori. La similarità s sarà allora definita come

$$s(\mathbf{x}_1, \mathbf{x}_2) \;=\; \mathbf{x}_1 \cdot \mathbf{x}_2 \tag{2.1}$$

dove il simbolo ' $\cdot$ ' indica il prodotto interno di $\mathbb{R}^p$, ovvero, indicando con $\mathbf{x}_i(j)$ la j-esima componente del vettore $\mathbf{x}_i$

$$\mathbf{x}_1 \cdot \mathbf{x}_2 \;:=\; \sum_{j=1}^{p} x_{1j}\, x_{2j}$$

Distanza euclidea Il modo più naturale per definire la distanza tra due vettori-documento è l'uso della distanza euclidea di $\mathbb{R}^p$. Dati quindi i documenti $\mathbf{x}_1$ e $\mathbf{x}_2$, costruiti con una qualsiasi scelta per i pesi, si avrà

$$d(\mathbf{x}_1, \mathbf{x}_2) \;=\; \|\mathbf{x}_1 - \mathbf{x}_2\|_2 \tag{2.2}$$

Similarità del coseno La misura di similarità più usata nell'ambito del text mining è la similarità del coseno. Essa è definita come il prodotto scalare normalizzato di due vettori, e coincide con il coseno dell'angolo di incidenza dei due vettori. Dati quindi due documenti $\mathbf{x}_1$ e $\mathbf{x}_2$, costruiti utilizzando i pesi (1.2), (1.3) o (1.4), definiamo la similarità s_{cos} come

$$s_{cos}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\|\mathbf{x}_1\|\|\mathbf{x}_2\|} \tag{2.3}$$

Passiamo ora alla descrizione dei diversi algoritmi di classificazione supervisionata elencati in precedenza.

1.3 K nearest neighbors

L'idea alla base del metodo *knn* è semplice. Dato un insieme di documenti già classificati (il training set) e dato un nuovo documento $\mathbf{x}$, si ricercano i k documenti del training set più simili a $\mathbf{x}$ (equivalentemente meno distanti da $\mathbf{x}$). Se la maggioranza di essi (o una quota Q prestabilita) appartiene al gruppo g, allora anche $\mathbf{x}$ è assegnato al gruppo g.
La difficoltà principale legata all'algoritmo *knn* consiste nell'individuare dei valori ottimali per i parametri k e Q. Inoltre, è necessario disporre di algoritmi efficienti per l'estrazione dal training set dei k elementi più simili al documento in esame.

1.4 Classificatori bayesiani e modelli lineari

Spesso, i classificatori chiamati *bayesiani* nella letteratura legata al text mining sono altrettanto noti come metodi *naive Bayes*. Non si basano infatti su modelli propriamente bayesiani, ma ricorrono alla formula di Bayes per la probabilità condizionata nella definizione del metodo.

Notazioni

Introduciamo in un primo momento alcune notazioni utili nell'esporre i classificatori bayesiani. L'unità statistica è sempre il documento, descritto tramite un vettore di feature. Modelliamo quindi un documento attraverso un **vettore aleatorio X**. Una singola realizzazione del vettore $\mathbf{X}$ sarà indicata con la lettera minuscola $\mathbf{x}$. Diciamo quindi che per una singola istanza si osservano i valori $\mathbf{x}$, realizzazione del vettore aleatorio $\mathbf{X}$. Supponiamo inoltre di costruire il vettore documento attraverso pesi binari (1.2), quindi $\mathbf{X}$ assume valori in $\{0, 1\}^p$.

Per quanto riguarda la classificazione, a differenza di quanto visto fino ad ora, supponiamo che l'assegnazione di un documento ad uno dei G gruppi sia espressa tramite una variabile aleatoria categorica Y, a valori in $\{1, 2, ..., G\}$.

Supponiamo allora di avere un documento, quindi di conoscere una realizzazione $\mathbf{x}$ del vettore aleatorio $\mathbf{X}$, ma di non conoscere direttamente la categoria cui appartiene. Potremmo allora stimare, per ogni possibile categoria $g \in \{1, 2, ..., G\}$, la probabilità che il documento appartenga ad essa $\mathbb{P}(Y = g|\mathbf{X} = \mathbf{x})$, per $g = 1, ..., G$, ed assegnarlo alla categoria con probabilità più alta. Usando la formula di Bayes, possiamo allora scrivere:

$$\mathbb{P}(Y = g|\mathbf{X} = \mathbf{x}) = \frac{\mathbb{P}(\mathbf{X} = \mathbf{x}|Y = g)\,\mathbb{P}(Y = g)}{\mathbb{P}(\mathbf{X} = \mathbf{x})} \tag{2.4}$$

Il problema sta nel fatto che le distribuzioni $\mathbb{P}(\mathbf{X} = \mathbf{x})$, $\mathbb{P}(Y = g)$ e $\mathbb{P}(\mathbf{X} = \mathbf{x}|Y = g)$ non sono, a priori, note. Servono perciò delle stime per tali distribuzioni, che possiamo ricavare sfruttando il training set.
Innanzi tutto, possiamo stimare la distribuzione delle categorie $\mathbb{P}(Y = g)$ attraverso le proporzioni del training set. Se, come fatto in precedenza, indichiamo con P_g la proporzione di documenti del training set appartenenti al gruppo g, scriviamo allora

$$\mathbb{P}(Y = g) \simeq P_g$$

Le stime per le distribuzioni di $\mathbf{X}$ e $\mathbf{X}|Y$ sono più delicate. Per sfruttare l'informazione fornita dal training set, occorre infatti introdurre l'ipotesi che le componenti del vettore $\mathbf{X}$ siano indipendenti. Questa ipotesi è poco realistica: le parole presenti in un testo scritto possono essere difficilmente considerate indipendenti le une dalle altre. Questa ipotesi è perciò il punto debole del modello.
L'indipendenza tra le p componenti di $\mathbf{X}$, indicate con X_j, $j = 1, ..., p$, permette di scrivere la legge di $\mathbf{X}$ come prodotto delle leggi marginali delle X_j

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) = \prod_{j=1}^{p} \mathbb{P}(X_j = x_j) \tag{2.5}$$

Allo stesso modo si ha

$$\mathbb{P}(\mathbf{X} = \mathbf{x}|Y = g) = \prod_{j=1}^{p} \mathbb{P}(X_j = x_j|Y = g) \tag{2.6}$$

Ora, è possibile stimare le distribuzioni delle X_j e le condizionali $X_j|Y$ attraverso le proporzioni nel training set. Ricordiamo che, scegliendo pesi binari, X_j assume valori in $\{0, 1\}$. Abbiamo allora, richiamando la (1.5):

$$\mathbb{P}(X_j = 1) \simeq P_j := \frac{\#\text{ documenti del training set contenenti } j}{\#\text{ documenti del training set}}$$

In modo analogo, si adotta la stima seguente per $X_j|Y$ e per Y:

$$\begin{aligned} \mathbb{P}(X_j = 1|Y = g) &\simeq \frac{\#\text{ documenti del training set contenenti } j \text{ nel gruppo } g}{\#\text{ documenti del training set nel gruppo g}} \\ \mathbb{P}(Y = g) &\simeq \frac{\#\text{ documenti del training set nel gruppo } g}{\#\text{ documenti del training set}} \end{aligned}$$

A questo punto, abbiamo tutti gli elementi necessari per stimare la (2.4). Possiamo allora utilizzare la (2.6) e, tenendo conto che le X_j assumono valori in $\{0, 1\}$, otteniamo

$$\mathbb{P}(Y = g|\mathbf{X} = \mathbf{x}) = \frac{\mathbb{P}(Y = g)}{\mathbb{P}(\mathbf{X} = \mathbf{x})} \prod_{j=1}^{p} \Big[\Big(\frac{\mathbb{P}(X_j = 1|Y = g)}{\mathbb{P}(X_j = 0|Y = g)} \Big)^{x_j} \mathbb{P}(X_j = 0|Y = g) \Big] \tag{2.7}$$

Modello Bernoulli multivariato

Dalla scrittura (2.7) si ricava la forma generica dei modelli lineari. Operiamo quindi una trasformazione, introducendo il vettore $\mathbf{w}^{(g)} \in \mathbb{R}^p$ ed il termine noto $b^{(g)}$ definiti come

$$w_j^{(g)} := \text{logit}\big(\mathbb{P}(X_j = 1|Y = g)\big) := \log\Big(\frac{\mathbb{P}(X_j = 1|Y = g)}{\mathbb{P}(X_j = 0|Y = g)}\Big) \tag{2.8}$$

$$b^{(g)} := log(\mathbb{P}(Y = g)) + \sum_j \log(\mathbb{P}(X_j = 0|Y = g)) \tag{2.9}$$

Possiamo allora riscrivere la (2.7) come

$$\mathbb{P}(Y = g|\mathbf{X} = \mathbf{x}) \;=\; \frac{1}{\mathbb{P}(\mathbf{X} = \mathbf{x})} \exp\{ \sum_j w_j^{(g)} X_j \,+\, b^{(g)} \} \tag{2.10}$$

Ora, la formula (2.10) può riferirsi a più modelli lineari, che differenziano per una diversa espressione di $\mathbf{w}^{(g)}$ e di $b^{(g)}$. Definendo $\mathbf{w}^{(g)}$ e $b^{(g)}$ come in (2.8) e (2.9), si ottiene il cosiddetto *modello Bernoulli multivariato*.

Modello multinomiale

Un altro modello molto diffuso è il *multinomiale*, che si ottiene a partire dalla (2.10) scegliendo le seguenti espressioni per $\mathbf{w}^{(g)}$ e $b^{(g)}$:

$$w_j^{(g)} \;=\; \log\big[\, \frac{\lambda + \mathbb{P}(X_j = 1|Y = g)}{\lambda p + \sum_{k=1}^{p} \mathbb{P}(X_k = 1|Y = g)} \,\big] \tag{2.11}$$

$$b^{(g)} \;=\; \log(\,\mathbb{P}(Y = g)\,) \tag{2.12}$$

Il parametro $\lambda > 0$ può essere modificato per regolarizzare il modello. Solitamente in letteratura si usa impostare $\lambda = 1$.

1.5 Modelli lineari generali

È possibile generalizzare le espressioni viste in precedenza. Supponiamo per semplicità che si voglia operare una classificazione binaria e che Y assuma valori in $\{-1, 1\}$, dove -1 e 1 indicano i due gruppi. Un classificatore lineare è una funzione lineare f dell'osservazione $\mathbf{x}$:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{2.13}$$

Nel caso binario, possiamo costruirla in modo da assegnare la categoria in base al segno di f:

$$\begin{cases} f(\mathbf{x}) > 0 \longrightarrow y = 1 \\ f(\mathbf{x}) < 0 \longrightarrow y = -1 \end{cases} \tag{2.14}$$

Il problema è dunque quello di determinare il vettore dei pesi $\mathbf{w}$ ed il coefficiente di *bias* b in modo che, applicando f ai documenti del training set, i

valori di y ottenuti siano il più possibile coincidenti con quelli effettivamente osservati.
Definiamo allora l'errore di misclassificazione come

$$I(f(\mathbf{x}), y) = \begin{cases} 1\,, & \text{se } \; y \times f(\mathbf{x}) \leq 0 \\ 0\,, & \text{se } \; y \times f(\mathbf{x}) > 0 \end{cases} \tag{2.15}$$

La determinazione di $\mathbf{w}$ e b si fa quindi minimizzando gli errori di misclassificazione. Formalmente, se $\mathbf{x}_1, ..., \mathbf{x}_m$ sono gli elementi del training set, si ha

$$(\hat{\mathbf{w}}, \hat{b}) = \arg\min_{\mathbf{w},b} \sum_{i=1}^{m} I(\, \mathbf{w} \cdot \mathbf{x}_i + b\,,\; y_i\,) \tag{2.16}$$

Minimizzazione di upper bound

Il problema formulato sopra è NP-difficile. In generale, si adottano quindi metodi propri della teoria dell'ottimizzazione per determinare $\mathbf{w}$ e b. Per risolvere il problema, è possibile sostituire la funzione $I(f(\mathbf{x}), y)$ con un *upper bound* $\tilde{I}(f(\mathbf{x}), y)$.
Ad esempio, si definisce la *Hinge loss function* come:

$$\tilde{I}(f(\mathbf{x}), y) = \begin{cases} 1 - y \times f(\mathbf{x}) & \text{se } \; y \times f(\mathbf{x}) \leq 1 \\ 0 & \text{se } \; y \times f(\mathbf{x}) > 1 \end{cases} \tag{2.17}$$

Si cerca allora il minimo di

$$(\hat{\mathbf{w}}, \hat{b}) = \arg\min_{\mathbf{w},b} \sum_{i=1}^{m} \tilde{I}(\mathbf{w} \cdot \mathbf{x}_i + b\,,\; y_i) \tag{2.18}$$

Un'altra possibile scelta per l'upper bound è la *robust classification loss function*, definita come:

$$\tilde{I}(f(\mathbf{x}), y) = \begin{cases} -2y \times f(\mathbf{x}) & \text{se } \; y \times f(\mathbf{x}) < -1 \\ \frac{1}{2}(y \times f(\mathbf{x}) \; - \; 1)^2 & \text{se } \; y \times f(\mathbf{x}) \in [-1, 1] \\ 0 & \text{se } \; y \times f(\mathbf{x}) > 1 \end{cases} \tag{2.19}$$

Formulazione *robust risk minimization*

Un ulteriore metodo per calcolare $\mathbf{w}$ e b è attraverso la formulazione *robust risk minimization*. Essa consiste nell'adottare la *robust classification loss function*, e di aggiungere un vincolo al problema di ottimizzazione:

$$\begin{cases} (\hat{\mathbf{w}}, \hat{b}) = \arg\min_{\mathbf{w},b} \sum_{i=1}^{m} \tilde{I}(\mathbf{w} \cdot \mathbf{x}_i + b\,,\, y_i) \\ \\ \text{t.c. } \|\mathbf{w}\|^2 + b^2 \leq A \end{cases} \tag{2.20}$$

Il problema di ottimizzazione vincolata è poi risolto tramite metodo della lagrangiana. Introduciamo quindi $\lambda > 0$, ottenendo la formulazione *robust risk minimization* definitiva:

$$(\hat{\mathbf{w}}, \hat{b}) = \arg\min_{\mathbf{w},b} \Big[\sum_{i=1}^{m} \tilde{I}(\mathbf{w} \cdot \mathbf{x}_i + b\,,\, y_i) + \frac{\lambda}{2}(\|\mathbf{w}\|^2 + b^2) \Big] \tag{2.21}$$

Non andiamo oltre nella descrizione dei metodi per ricavare dei classificatori lineari. Spostiamo invece la nostra attenzione sulla valutazione dei classificatori ottenuti.

1.6 Valutazione di classificatori

È necessario introdurre degli indici per valutare la bontà dei classificatori introdotti. Supponiamo quindi di avere, a fianco del training set, un *test set* composto da documenti la cui categoria è nota, cui applicare il classificatore per valutarne le performances. Attraverso i risultati ottenuti, possiamo calcolare allora gli indici più usati:

- **error rate:**

$$e_{rate} := \frac{\#\text{errori}}{\#\text{documenti}} \tag{2.22}$$

- **standard error**:

$$SE := \sqrt{\frac{e_{rate}(1 - e_{rate})}{\#\text{documenti}}} \tag{2.23}$$

Ciò che accade spesso nella classificazione binaria di testi, è che i casi di testi classificati positivi sono molto più rari rispetto a quelli classificati negativi. Ad esempio, se stiamo cercando, tra un grande numero di articoli di

giornale, gli articoli relativi allo sport (classificati positivi), nella collezione di documenti saranno molto maggiori gli articoli che non trattano di sport, quindi i classificati negativi.
In tal caso, gli indici appena introdotti sono poco indicativi. Se costruissimo infatti un classificatore che associa $y = -1$ ad ogni $\mathbf{x}$, esso classificherebbe correttamente la maggior parte dei documenti, e si avrebbe $e_{rate} \ll 1$.
Per valutare meglio i classificatori in casi simili, si introducono allora altri indici:

- **precision:**

$$\frac{\#\text{classificati positivi corretti}}{\#\text{classificati positivi}} \tag{2.24}$$

- **recall:**

$$\frac{\#\text{classificati positivi corretti}}{\#\text{documenti positivi}} \tag{2.25}$$

- **F-measure:**

$$\left[\, \frac{1}{2}\left(\frac{1}{\text{precision}} + \frac{1}{\text{recall}}\right) \right]^{-1} \tag{2.26}$$

A seconda dello scopo della classificazione, sarà più utile adottare un indice o l'altro. Ad esempio, per un filtro anti-spam sarà molto importante che le mail classificate come *spam* (classificazione positiva) non siano in realtà interessanti. Sarà dunque più indicativa la *precision* rispetto al *recall*. Al contrario, se stiamo cercando dei documenti che contengono una certa informazione, classificando positivi i documenti rilevanti, sarà fondamentale che tutti i documenti effettivamente rilevanti siano ritrovati, poco importa se è stato individuato qualche documento fuori luogo. In tal caso sarà dunque più interessante il *recall*.
Infine, è importante sottolineare un dettaglio sulla costruzione del test set e del training set. Una delle difficoltà principali per il text mining è legata all'evoluzione temporale dei documenti, che rendono meno efficaci gli algoritmi impiegati. Per avere una valutazione sulla predittività dei classificatori, è importante scegliere il test set in modo che sia composto da doumenti più recenti del data set. Ciò permette di verificare le performances di un classificatore tenendo conto degli effetti dell'evoluzione temporale nella collezione di documenti.

2 Recupero di informazioni (*information retrieval*)

Il problema del *recupero di informazioni*, noto in letteratura come *information retrieval*, consiste nella ricerca, in una vasta collezione di documenti, dei testi che contengono le informazioni ricercate da un utente. L'utente presenta quindi al sistema una *query* di ricerca, che viene confrontata con i documenti per estrarre solo quelli rilevanti rispetto alla richiesta. L'*information retrieval* può essere inteso come un problema di classificazione binaria (documenti interessanti/irrilevanti), ma presenta delle particolarità, in primis la presenza di una stringa di testo di riferimento, la query.

2.1 Ricerca tramite misura di similarità

È possibile considerare la query di ricerca come un piccolo documento, e calcolarne la similarità con gli elementi della collezione. In tal caso, saranno presentati all'utente solo i documenti più simili alla query. Va ricordato che in letteratura, oltre alle definizioni di similarità già viste (2.1) e (2.3), si usa spesso una variante della (2.1). Tale variante aggiunge al conteggio dei token comuni a due documenti un termine che tiene conto della rilevanza dei token. Siano quindi $\mathbf{u}$ e $\mathbf{v}$ due vettori documento costruiti con pesi binari. Sommando sulle diverse componenti j dei vettori otteniamo la misura di similarità:

$$s(\mathbf{u}, \mathbf{v}) = \sum_{j=1}^{p} u_j v_j + \frac{1}{df(j)} \tag{2.27}$$

dove $df(j)$ è definito in (1.1). Il termine correttivo dà più rilevanza ai token rari, in modo che due documenti risultino più simili fra di loro nel caso la similarità sia data da parole rare.

Il metodo del calcolo di similarità è semplice e piuttosto intuitivo, ma presenta gravi carenze che lo rendono inapplicabile senza opportune modifiche. Data una query composta da poche parole, se la collezione di documenti è molto vasta, è probabile che vi siano moltissimi documenti molto simili alla query. Non solo, è probabile che molti documenti contengano tutti i token presenti nella query. L'utente si troverebbe così davanti ad un numero troppo grande di documenti estratti, tutti molto simili alla query e dunque ugualmente importanti. Si rende perciò necessario saper distinguere, tra i documenti aventi massima similarità con la query, quelli più o meno rilevanti.

2.2 Ordinamento basato sulla rilevanza

In linea generale, si può definire un ordinamento di rilevanza per i documenti in due modi:

1. basandosi sulla query posta dall'utente, e quindi ricalcolato ad ogni ricerca;
2. indipendentemente dalla ricerca specifica, in modo che i documenti siano ordinati una volta per tutte in base alla loro importanza.

Il secondo metodo si è rivelato estremamente efficiente. La principale applicazione dell'*information retrieval* consiste certamente nei motori di ricerca. Il web è la più grande fonte di documenti scritti, ed è indispensabile disporre di strumenti di ricerca efficaci. L'algoritmo che ha rivoluzionato il web searching, il *Google Rank Page*, è basato proprio su un ordinamento dei documenti in base alla loro rilevanza, indipendentemente dal contenuto semantico dei documenti. Ricordiamo brevemente l'idea alla base dell'algoritmo.

2.3 Analisi dei link per ordinamento di pagine web

Una classe di elementi molto informativi presenti in una pagina web sono i collegamenti verso altre pagine. Grazie a tali collegamenti, è possibile descrivere internet attraverso un grafo, i cui nodi sono le pagine ed i lati i link. Sfruttando un'idea legata al mondo della ricerca, in cui l'importanza di una pubblicazione è valutata sulla base delle citazioni autorevoli ad essa, si può allora pensare di sfruttare i link per costruire un ordinamento di rilevanza. Più precisamente, si vuole definire un *indice* di rilevanza r.

Sia quindi A la pagina web di cui vogliamo calcolare l'indice di rilevanza, indicato con $r(A)$. Indichiamo con $T_1, ..., T_n$ le n pagine che contengono un link verso A. Per ognuna delle pagine T_j, $C(T_j)$ indica il numero di link presenti in T_j verso altre pagine. L'indice $r(A)$ può essere definito ricorsivamente come:

$$r(A) \; = \; \sum_{j=1}^{n} \frac{r(T_j)}{C(T_j)} \tag{2.28}$$

Non andiamo oltre nell'illustrazione dell'algoritmo *Page Rank*. Vanno però fatte alcune considerazioni sull'analisi dei link.
Così come il titolo di una pagina web contiene parole chiave per la comprensione del contenuto, così l'*anchor text*, ovvero il testo su cui cliccare per raggiungere la pagina cui punta il link, può contenere informazioni molto

indicative sul contenuto della pagina di arrivo. Spesso infatti l'anchor text consiste in una brevissima descrizione della pagina, e contiene parole chiave. Anche per problemi di classificazione applicata a pagine web, l'informazione proveniente dai link può essere utile: è probabile che pagine linkate tra loro siano considerabili più simili, e quindi appartengano più probabilmente alla stessa categoria.

2.4 Document matching

Un caso particolare di information retrieval si ha quando, al posto di una query di ricerca scritta dall'utente, si utilizza per la comparazione un intero documento. Questo caso è noto come *document matching*.
Si può avere un esempio di applicazione nell'ambito di un servizio clienti. Supponiamo che l'help desk riceva una mail contenente la descrizione dettagliata di un problema. Sarebbe interessante ricercare, tra tutte le richieste già effettuate dai clienti in passato, quelle che sono più simili alla presente. In tal modo si potrebbe avere un rapido accesso alle soluzioni già proposte, riducendo notevolmente i tempi di risposta. Allo stesso modo, si possono pensare sistemi online che, disponendo di ampie basi di dati, forniscano automaticamente risposte possibili alle richieste di utenti.
Le tecniche usate sono le stesse coinvolte nell'information retrieval, con la differenza che, in questo caso, non è necessario un ordinamento dei documenti sulla base della rilevanza.

Capitolo 3

Analisi cluster

Non sempre si conoscono a priori le categorie in cui possono essere suddivisi i documenti di una collezione, e non sempre si dispone di training set annotati. Spesso non si ha alcuna informazione sui testi. Risulta perciò interessante esplorare la collezione suddividendola in gruppi di documenti simili tra loro, effettuare cioè una classificazione non supervisionata (o analisi cluster).
Il contenuto dei gruppi non è però noto. Sarà necessario, dopo aver effettuato l'analisi cluster, indagare il significato dei gruppi. Si può allora pensare di ricercare le parole chiave o estrarre i documenti più rappresentativi. Vedremo in seguito come mettere in pratica queste idee.
Per dare un'idea di una possibile applicazione, riprendiamo l'esempio del servizio clienti. Per migliorare il servizio, è utile avere un'immagine chiara del tipo di problemi che i clienti sottopongono. Attraverso un'analisi cluster delle mail ricevute, sarà quindi possibile suddividere le richieste in gruppi, che corrisponderanno a tipologie diverse di problemi. Vedremo nel seguito altre applicazione dell'analisi cluster nel contesto del text mining. Ci concentriamo ora sulla descrizione degli algoritmi maggiormente impiegati nella letteratura riguardante il text mining.

1 Algoritmi di analisi cluster

Come per il data mining, nel text mining si usano sia algoritmi di tipo gerarchico che algoritmi basati sui centroidi dei gruppi. Iniziamo descrivendo il metodo delle *K-medie*.

1.1 K-medie

L'idea alla base dell'algoritmo è semplice. In un primo momento si fissa il numero k di cluster che si vogliono individuare. In seguito, si ricerca una

suddivisione dei documenti ottimale per k gruppi. In particolare, l'algoritmo cerca una disposizione in modo che sia massimizzata la similarità dei documenti con il centroide del gruppo cui appartengono, in generale il baricentro. Ne risulta un problema di ottimizzazione, la cui funzione obiettivo è la somma di tutte le similarità dei documenti con il baricentro del loro gruppo.
L'algoritmo generico per risolvere il problema può essere espresso nel modo seguente:

1. si distribuiscono tutti i documenti in k gruppi (ripartizione casuale)
2. si calcola il baricentro - la media - di ogni gruppo come media aritmetica dei vettori-documento che lo compongono, ottenenedo le medie $\mathbf{m}_1, ..., \mathbf{m}_k$
3. per ogni documento $\mathbf{x}_i$, si calcola la similarità con ognuna delle k medie $sim(\mathbf{x}_i, \mathbf{m}_1), ..., sim(\mathbf{x}_i, \mathbf{m}_k)$
4. si sposta ogni documento nel gruppo con la media più simile: $\mathbf{x}_i$ assegnato al gruppo $c := \arg\max_j sim(\mathbf{x}_i, \mathbf{m}_j)$
5. - SE non si è operato nessuno spostamento: FINE
   - ALTRIMENTI: si torna al punto 2.

È possibile mostrare che lo stato finale dell'algoritmo è un ottimo locale del problema di massimizzazione. Tuttavia, non è detto che l'ottimo locale sia unico. In generale, si possono ottenere risultati diversi a seconda di come venga scelto lo stato iniziale al punto 1.

Similarità, dissimilarità e distanza

La definizione del metodo delle k-medie è basata sull'introduzione di una misura di *similarità*. Analogamente, una definizione può essere data attraverso una misura di *dissimilarità* (a patto di sostituire il problema di massimizzazione con un analogo problema di minimizzazione). In questo caso, si cercherà di minimizzare la dissimilarità degli elementi con i centri dei cluster.
Formalmente, dati due elementi $\mathbf{x}_1$ e $\mathbf{x}_2$, una misura di dissimilarità d deve soddisfare le seguenti proprietà:

1. $d(\mathbf{x}_1, \mathbf{x}_2) = d(\mathbf{x}_2, \mathbf{x}_1)$
2. $d(\mathbf{x}_1, \mathbf{x}_2) = 0 \quad \text{se} \quad \mathbf{x}_1 = \mathbf{x}_2$
3. $d(\mathbf{x}_1, \mathbf{x}_2) > 0 \quad \text{se} \quad \mathbf{x}_1 \neq \mathbf{x}_2$

Possiamo utilizzare come misura di dissimilarità una qualsiasi distanza (una distanza verifica le proprietà 1 - 3 ed in più la disuguaglianza triangolare). Ricordiamo inoltre che, data una misura di dissimilarità d, è sempre possibile ricavare una misura di similarità sim, definita in funzione di d nel modo seguente:

$$sim(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{1 + d(\mathbf{x}_1, \mathbf{x}_2)}$$

1.2 Inizializzazione per il metodo delle k-medie

Si possono trovare molti studi in letteratura che propongono metodi diversi di inizializzazione (si veda ad esempio [20]). Nel seguito, ne riportiamo in particolare due: un semplice algoritmo *greedy* citato in [29] ed un metodo molto diffuso, noto come *k-means++*, descritto in [1]. Facciamo presente che in generale (e questo è il caso del metodo *k-means++*), più che determinare una partizione dei dati, l'inizializzazione consiste nel definire k centri iniziali.

Inizializzazione *greedy*

Esistono molti metodi per definire una suddivisione iniziale in k gruppi. La più diffusa inizializzazione *greedy* consiste nel calcolare il baricentro di tutta la collezione ed ordinare i documenti in base alla similarità con tale baricentro. Gli n documenti ordinati saranno poi suddivisi in k gruppi nel modo seguente (supponiamo che n/k sia intero): i primi n/k documenti più simili al baricentro nel primo gruppo, i secondi n/k più simili nel secondo gruppo e così via. Questa inizializzazione permette di ottenere una convergenza rapida dell'algoritmo ma, se i vettori documento sono distribuiti uniformemente in $\mathbb{R}^p$, darà facilmente luogo a dei gruppi disposti in forma di corone circolari concentriche, come illustrato in figura (3.1).

k-means++

L'algoritmo di inizializzazione *k-means++*, presentato per la prima volta nel 2006, prevede la selezione di k elementi del set di dati come centri iniziali. Nella sua formulazione, l'algoritmo prevede l'uso di una misura di dissimilarità.

Sia allora $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ il set di dati, con $\mathcal{X} \subset \mathbb{R}^p$. Supponiamo che, all'inizio del passo h-esimo dell'algoritmo, si siano individuati $h - 1$ centri $\mathbf{m}_1, ..., \mathbf{m}_{h-1} \in \mathcal{X}$. Indichiamo con $\hat{\mathcal{X}} \subseteq \mathcal{X}$ l'insieme dei punti del data set non ancora utilizzati come centri iniziali, ovvero

$$\hat{\mathcal{X}} = \mathcal{X} \setminus \{\mathbf{m}_1, ..., \mathbf{m}_{\mathbf{h-1}}\}$$

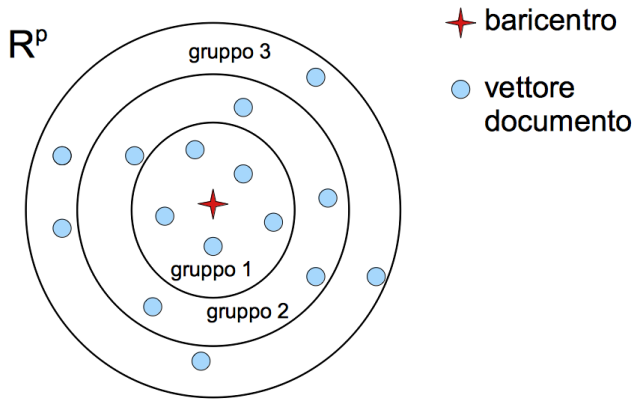


Figura 3.1: Formazione di gruppi concentrici per inizializzazione greedy, caso $k = 3$, $n/k = 5$

Dato allora un punto $\mathbf{x} \in \hat{\mathcal{X}}$, indichiamo con $D(\mathbf{x})$ la dissimilarità minima di $\mathbf{x}$ dai centri $\mathbf{m}_1, ..., \mathbf{m}_{h-1}$:

$$D(\mathbf{x}) \ := \min_{j=1,...,h-1} d(\mathbf{x}, \mathbf{m}_j)$$

L'algoritmo può essere quindi schematizzato nel modo seguente.

1. (**Passo iniziale**) Si sceglie il primo centro $\mathbf{m}_1$ tramite un campionamento casuale in $\mathcal{X}$, seguendo una distribuzione uniforme tra i dati ($\mathbb{P}(\mathbf{x}_i) = 1/n \quad$ per ogni $\ i = 1, ..., n$).
2. (**Passo h-esimo**) Si sceglie il centro $\mathbf{m}_h$ tramite un campionamento casuale in $\hat{\mathcal{X}}$, seguendo una distribuzione di probabilità definita come:

$$\mathbb{P}(\mathbf{x}_i) \ = \ \frac{D(\mathbf{x}_i)^2}{\sum_{\mathbf{x}_j \in \hat{\mathcal{X}}} D(\mathbf{x}_j)^2}$$

3. - SE $h < k$ ripetere dal punto 2
   - ALTRIMENTI: FINE

La distribuzione di probabilità definita al punto 2 assegna peso maggiore agli elementi più dissimili dai centri già individuati, in modo quadratico. Questo permette di evitare centri iniziali troppo vicini. Si può mostrare (si veda [1]) che l'algoritmo, pur non richiedendo importanti risorse di calcolo, permette di ottenere una buona struttura finale dei dati clusterizzati e di migliorare la rapidità di convergenza del metodo delle k-medie.

1.3 Scelta di k

In generale, l'algoritmo descritto per il metodo delle k medie è molto efficiente. Tuttavia, al crescere di k il problema di ottimizzazione diventa molto complesso, e la risoluzione richiede molto più sforzo computazionale.

La scelta del parametro k, oltre ad influenzare le prestazioni dell'algoritmo, è di fondamentale importanza per ottenere risultati informativi dall'analisi cluster. È evidente che scegliendo k molto piccolo, si rischia di descrivere i dati con troppa superficialità, ricavandone una struttura troppo grossolana. Al contrario, un valore di k troppo alto forzerà l'analisi a creare dei gruppi di pochi elementi, descrivendo una struttura poco generale e poco informativa.

Indice di dispersione

Possiamo introdurre un indice per valutare la bontà della struttura riconosciuta dall'analisi cluster. Tale indice terrà conto della dispersione dei documenti rispetto ai centri dei gruppi cui sono stati assegnati. Siano allora $\mathbf{x}_1, ..., \mathbf{x}_n$ i vettori-documento della collezione. Per ogni documento $\mathbf{x}_i$, c_i indica il cluster cui è stato assegnato, avente centro $\mathbf{m}_i$. L'indice di dispersione calcolato in seguito all'individuazione di k gruppi, $E(k)$, è espresso come

$$E(k) = \sum_{i=1}^{n} \frac{|\mathbf{x}_i - \mathbf{m}_i|^2}{n} \tag{3.1}$$

Più è bassa la dispersione, meglio è descritto il set di dati attraverso la suddivisione in cluster. Tuttavia, fissato il set di dati, $E(k)$ è decrescente con k, poiché con più gruppi si individuano, più i vettori si avvicinano ai centri. Non ha dunque senso cercare il minimo di $E(k)$. Tuttavia, tenendo presente quanto detto in precedenza sulla scelta di k (cioè che il numero di gruppi non deve essere né troppo piccolo, né troppo grande), è possibile trovare un valore ottimale. Esso corrisponde al valore di k a partire dal quale un incremento nel numero dei gruppi corrisponde ad un decremento molto piccolo di $E(k)$. In altri termini, tracciando l'andamento di $E(k)$ in funzione di k in un grafico, il valore ottimale corrisponde ad un *gomito* del grafico. Si riporta un esempio in figura (3.2).

Va sottolineato come il calcolo di $E(k)$ implichi la risoluzione del problema di clustering per ogni valore di k, quindi richiede un notevole sforzo computazionale.

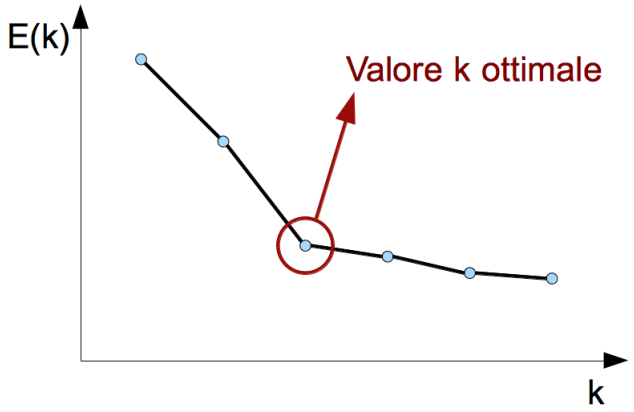


Figura 3.2: Esempio di valore ottimale per k ottenuto valutando $E(k)$

Funzione di silhouette

Un altro strumento utile per scegliere il numero k di gruppi è la cosiddetta *funzione di silhouette* (si vedano gli articoli [24] e [28]). Essa fornisce una valutazione sulla bontà del risultato dell'analisi cluster. La costruzione della funzione di silhouette si basa sul calcolo di opportuni indici, che necessitano della definizione di una misura di dissimilarità (si veda la sezione precedente per la definizione). Immaginiamo quindi di aver ottenuto k gruppi da un'analisi cluster. Consideriamo l'elemento $\mathbf{x}_i$, classificato nel cluster c_i. Definiamo allora la media della dissimilarità di $\mathbf{x}_i$ con gli elementi di c_i come

$$d(\mathbf{x}_i, c_i) \;:=\; \frac{1}{|c_i| - 1} \sum_{\mathbf{x}_j \in c_i} d(\mathbf{x}_i, \mathbf{x}_j) \tag{3.2}$$

dove $|c_i|$ indica il numero di documenti contenuti nel cluster c_i. Per alleggerire la notazione, indichiamo la quantità (3.2) come $a(\mathbf{x}_i)$:

$$a(\mathbf{x}_i) \;:=\; d(\mathbf{x}_i, c_i)$$

Allo stesso modo, possiamo definire la dissimilarità media di $\mathbf{x}_i$ con gli elementi di un qualsiasi altro cluster c, definita come

$$d(\mathbf{x}_i, c) \;:=\; \frac{1}{|c|} \sum_{\mathbf{x}_j \in c} d(\mathbf{x}_i, \mathbf{x}_j)$$

Ora, cerchiamo il cluster diverso da c_i più *vicino* a $\mathbf{x}_i$. La dissimilarità media di tale cluster con $\mathbf{x}_i$ sarà indicata come $b(\mathbf{x}_i)$:

$$b(\mathbf{x}_i) \;:=\; \min_{c \neq c_i} d(\mathbf{x}_i, c)$$

Possiamo allora definire la **funzione di silhouette** per $\mathbf{x}_i$ come

$$\mathcal{S}(\mathbf{x}_i) := \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max\{a(\mathbf{x}_i), b(\mathbf{x}_i)\}} \tag{3.3}$$

La (3.3) assume valori in $[-1, 1]$, che indicano la bontà o meno dell'assegnazione di $\mathbf{x}_i$ al cluster c_i.

- $\mathcal{S}(\mathbf{x}_i) \approx 1 \implies$ l'assegnazione di $\mathbf{x}_i$ in c_i è buona;
- $\mathcal{S}(\mathbf{x}_i) \approx 0 \implies$ $\mathbf{x}_i$ si trova a metà strada tra due cluster;
- $\mathcal{S}(\mathbf{x}_i) \approx -1 \implies$ l'assegnazione di $\mathbf{x}_i$ in c_i non è buona: $\mathbf{x}_i$ è più *vicino* agli elementi di un altro cluster.

È possibile calcolare un indice di bontà dell'analisi cluster attraverso la media della funzione di silhouette per tutti gli n elementi del set di dati $\mathcal{S}_{av}$:

$$\mathcal{S}_{av} := \frac{1}{n} \sum_{i=1}^{n} \mathcal{S}(\mathbf{x}_i)$$

Per valutare il valore ottenuto per $\mathcal{S}_{av}$, si fa solitamente riferimento ad una tabella simile a quella riportata di seguito:

Valore di $\mathcal{S}_{av}$	**Possibile interpretazione**
0.71 - 1.00	È stata trovata una struttura evidente
0.51 - 0.70	È stata trovata una struttura ragionevole
0.26 - 0.50	La struttura trovata è poco evidente
≤ 0.25	Non è stata trovata alcuna struttura sostanziale

In genere, quando la struttura trovata è poco evidente, è utile provare ad utilizzare un altro metodo di cluster analysis.
Per scegliere k, è possibile valutare la $\mathcal{S}_{av}$ ottenuta per diversi valori di k, e scegliere quello che dà il valore più alto.

La funzione di silhouette permette inoltre di costruire uno strumento grafico per valutare la bontà dell'analisi cluster: il *silhouette plot*. Dopo aver calcolato la funzione di silhouette (3.3) per tutti gli elementi del set di dati, i valori ottenuti saranno raggruppati per cluster, possibilmente ordinati in ordine decrescente, ed in seguito rappresentati su un grafico. La curva ottenuta per un singolo cluster è detta *silhouette* del cluster. Un esempio simulato di Silhouette Plot è riportato in Figura 3.3. In esso vediamo un

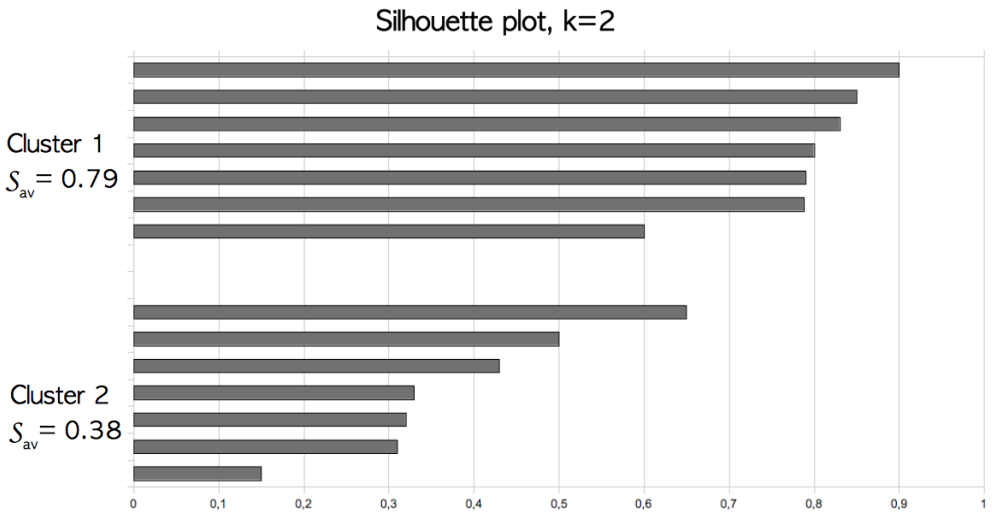


Figura 3.3: Esempio di silhouette plot per un'analisi cluster con $k = 2$ gruppi. Il secondo cluster sembra poco ragionevole, forse è opportuno aumentare k.

cluster la cui silhouette è *piena*, ovvero con valori vicini ad 1, che indica una struttura evidente. Il secondo cluster invece, contraddistinto da una silhouette più scarna, non corrisponde ad una struttura evidente nei dati.

1.4 Algoritmi gerarchici

A differenza del metodo delle k-medie, gli algoritmi gerarchici non prevedono la scelta iniziale di k. Al contrario, forniscono una descrizione completa della struttura dei dati, rimandando ad un secondo momento la scelta del numero di gruppi. L'output di un algoritmo gerarchico consiste in genere in un albero, che descrive le varie fasi di aggregazione dei dati, ovvero la formazione dei cluster. Nello stato iniziale dell'algoritmo, tutti i vettori costituiscono un cluster a sè. Durante le iterazioni successive, i cluster più simili tra loro vengono uniti per formare dei gruppi più grandi. L'algoritmo procede fin quando tutti i cluster sono stati uniti, ed è risultato un unico grande gruppo contenente tutti i dati.

Similarità tra gruppi

Resta da definire un metodo per scegliere i cluster più *simili* ed unirli durante le iterazioni. Nel capitolo precedente, sono state introdotte diverse misure di similarità tra documenti. In questo caso, il concetto viene esteso definendo

delle misure di similarità tra cluster. Si possono dare diverse definizioni. Le più usate sono la *single link*, la *complete link* e l'*average link*. Siano allora A e B due cluster, le diverse misure di similarità si definiscono nei modi seguenti:

- **single link**:

$$sim(A, B) = \max_{a \in A, b \in B} sim(a, b)$$

- **complete link**:

$$sim(A, B) = \min_{a \in A, b \in B} sim(a, b)$$

- **average link**:

$$sim(A, B) = \sum_{a \in A, b \in B} \frac{sim(a, b)}{(\#A \#B)}$$

  dove $\#A$ e $\#B$ indicano, rispettivamente, il numero di elementi contenuti in A e in B.

Schema generale di algoritmo gerarchico

Lo schema generale di un algoritmo gerarchico resta invariato cambiando la scelta della similarità tra cluster. Esso può essere definito tramite i seguenti passi:

1. Ogni documento rappresenta un cluster
2. Si trovano i due cluster con similarità maggiore
3. Si uniscono i due cluster trovati in un unico cluster
4. - SE resta un unico cluster: fine
   - ALTRIMENTI: si torna al punto 2.

L'albero con cui si rappresenta l'analisi cluster, effettuata tramite un algoritmo gerarchico, sarà dunque binario. La scelta del numero di gruppi equivale alla scelta di un livello dell'albero, corrispondente ad una data configurazione dei cluster.
In generale, gli algoritmi gerarchici sono piuttosto costosi da un punto di vista computazionale, in particolare quando si impiega la similarità tra cluster di tipo *complete link*.

2 Soft clustering

Fino ad ora, abbiamo sempre descritto algoritmi di analisi cluster che *assegnano* ogni documento ad un dato gruppo. Tali algoritmi seguono modelli detti di *hard clustering*. Le tecniche di *soft clustering* prevedono invece una modellazione più fine: l'assegnazione di un elemento ad un cluster avviene attraverso l'introduzione di una variabile aleatoria.

Immaginiamo di aver fissato a priori il numero k di cluster, e di poterli ordinare indicandoli ciascuno con un intero $1, ..., k$. Come per i modelli visti in precedenza nel caso della classificazione supervisionata, ad ogni elemento $\mathbf{x}_i$ del set di dati possiamo associare una realizzazione y_i della variabile aleatoria Y. La variabile aleatoria Y assume valori in $\{1, ..., k\}$. Un metodo di *soft clustering* si occupa perciò di trovare una stima della distribuzione di probabilità di $Y|\mathbf{X} = \mathbf{x}_i$.

Per riassumere, la differenza tra un metodo di *hard clustering* e *soft clustering* è la seguente:

- **Hard clustering:** il label y_i (relativo al documento $\mathbf{x}_i$) è assegnato ad un cluster fissato.
- **Soft clustering:** y_i è determinato assegnando una distribuzione di probabilità su $\{1, 2, ..., k\}$ per $Y|\mathbf{X} = \mathbf{x}_i$.

2.1 Modello bayesiano

Spesso, per il *soft clustering* si impiegano modelli bayesiani. Definiamo allora formalmente la distribuzione di probabilità della variabile Y attraverso k valori reali $\{\mu_1, ..., \mu_k\}$ tali che $\mu_c \leq 1$ e $\sum_c \mu_c = 1$:

$$\mathbb{P}(Y = c) \;=\; \mu_c$$

Supponiamo ora di conoscere la distribuzione di $\mathbf{X} \,|\, Y = c$. Per esprimere tale distribuzione, introduciamo formalmente un vettore di parametri θ_c legati al cluster c e scriviamo

$$\mathbf{X} \,|\, Y = c \;\sim\; \mathbf{X} \,|\, \theta_c \tag{3.4}$$

Se $\mathbb{P}(\mathbf{X} = \mathbf{x} \,|\, \theta_c)$ è nota, possiamo esprimere la distribuzione di $\mathbf{X}$ attraverso la formula della probabilità totale:

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) \;=\; \sum_{c=1}^{k} \mathbb{P}(Y = c)\, \mathbb{P}(\mathbf{X} = \mathbf{x} \,|\, Y = c) \;=\; \sum_{c=1}^{k} \mu_c\, \mathbb{P}(\mathbf{X} = \mathbf{x} \,|\, \theta_c) \tag{3.5}$$

Possiamo infine applicare la formula di Bayes, per esprimere la distribuzione di probabilità di $Y|\mathbf{X} = \mathbf{x}_i$:

$$\mathbb{P}(Y = \tilde{c}\,|\,\mathbf{X} = \mathbf{x}) \;=\; \frac{\mu_{\tilde{c}}\,\mathbb{P}(\mathbf{X} = \mathbf{x}\,|\,\theta_{\tilde{c}})}{\sum_{c=1}^{k}\mu_c\,\mathbb{P}(\mathbf{X} = \mathbf{x}\,|\,\theta_c)} \tag{3.6}$$

Introduciamo ora delle variabili supplementari $q_1, ..., q_k$, tali che $q_c \geq 0$ e $\sum_c q_c = 1$. Esse serviranno per stimare le probabilità ricercate $\mathbb{P}(Y = \tilde{c}\,|\,\mathbf{X} = \mathbf{x})$. Vediamo in che modo.
Introduciamo innanzi tutto una disuguaglianza, detta di *entropia mutuale*, valida tra due qualsiasi distribuzioni di probabilità discrete $\{p_1, ..., p_k\}$ e $\{s_1, ..., s_k\}$:

$$\sum_{c=1}^{k} s_c \log\big(\frac{s_c}{p_c}\big) \;\geq\; 0 \tag{3.7}$$

Si noti che l'uguaglianza è soddisfatta se vale $p_c = s_c\ \forall c$.
Applicando la (3.7) alle distribuzioni q_c e $Y\,|\,\mathbf{X} = \mathbf{x}$, otteniamo allora

$$\sum_{c=1}^{k} q_c \log\big(\frac{q_c}{\mathbb{P}(Y = c\,|\,\mathbf{X} = \mathbf{x})}\big) \;\geq\; 0 \tag{3.8}$$

e vale l'uguaglianza se $q_c \;=\; \mathbb{P}(Y = c\,|\,\mathbf{X} = \mathbf{x})$.
Possiamo riscrivere la (3.8) usando l'espressione (3.6), ottenendo:

$$\sum_{c=1}^{k} q_c\ \log\Big(\frac{q_c\ \sum_{l=1}^{k}\mu_l\mathbb{P}(\mathbf{X} = \mathbf{x}\,|\,\theta_l)}{\mu_c\mathbb{P}(\mathbf{X} = \mathbf{x}\,|\,\theta_c)}\Big) \;\geq\; 0 \tag{3.9}$$

Riorganizzando i termini, si ha:

$$\sum_{c=1}^{k} q_c\ \log\Big[\ \sum_{l=1}^{k}\mu_l\mathbb{P}(\mathbf{X} = \mathbf{x}\,|\,\theta_l)\ \Big] \;\geq\; \sum_{c=1}^{k} q_c \log\Big[\ \frac{\mu_c\mathbb{P}(\mathbf{X} = \mathbf{x}\,|\,\theta_c)}{q_c}\ \Big] \tag{3.10}$$

Ricordiamo poi che $\sum_c q_c \;=\; 1$ ed otteniamo infine l'espressione:

$$\log\Big(\ \sum_{c=1}^{k}\mu_c\,\mathbb{P}(\mathbf{X} = \mathbf{x}\,|\,\theta_c)\ \Big) \geq \sum_{c=1}^{k} q_c \log\Big(\ \frac{\mu_c\mathbb{P}(\mathbf{X} = \mathbf{x}\,|\,\theta_c)}{q_c}\ \Big) \tag{3.11}$$

Si noti che l'uguaglianza vale sempre nel caso in cui $q_c \;=\; \mathbb{P}(Y = c\,|\,\mathbf{X} = \mathbf{x})$. Ora, il termine di sinistra della disuguaglianza non dipende dai valori q_c. Se consideriamo μ_c e $\mathbb{P}(\mathbf{X} = \mathbf{x}\,|\,\theta_c)$ fissati, il termine di sinistra è un maggiorante del termine di destra. Possiamo allora sfruttare la (3.11) per stimare $\mathbb{P}(Y = c\,|\,\mathbf{X} = \mathbf{x})$ con q_c, massimizzando il termine di destra della disuguaglianza.

Scriviamo allora:

$$\mathbb{P}(Y = c \,|\, \mathbf{X} = \mathbf{x}) = \arg\max_{q_c} \sum_{c=1}^{k} q_c \log\big(\frac{\mu_c \mathbb{P}(\mathbf{X} = \mathbf{x} \,|\, \theta_c)}{q_c}\big) \tag{3.12}$$

Restano ora da stimare i parametri θ_c. Si impiega allora una stima di massima verosimiglianza, massimizzando quindi la probabilità di osservare il dato $\mathbf{x}$:

$$\hat{\theta} \;=\; \arg\max_{\theta} \prod_{i=1}^{n} \big(\sum_{c=1}^{k} \mu_c \mathbb{P}(\mathbf{X} = \mathbf{x}_i \,|\, \theta_c)\big) \tag{3.13}$$

Poiché il logaritmo è una funzione monotona, possiamo scrivere equivalentemente:

$$\hat{\theta} \;=\; \arg\max_{\theta} \sum_{i=1}^{n} \log\big(\sum_{c=1}^{k} \mu_c \mathbb{P}(\mathbf{X} = \mathbf{x}_i \,|\, \theta_c)\big) \tag{3.14}$$

Ci troviamo allora di fronte ad un problema di massimizzazione, al variare contemporaneamente dei possibili valori θ_c e q_c. Per risolvere questo problema, si adotta un algoritmo di tipo *expectation-maximization*. Vediamo dunque più in dettaglio in cosa consiste questo algoritmo.

2.2 Algoritmo expectation-maximization

Per ogni dato $\mathbf{x}_i$, vogliamo calcolare il valore ottimale di q_c, che indichiamo $q_{i,c}$. Sia $\mathbf{q}$ il vettore contenente tutti i valori $q_{i,c}$, (con $i = 1, ..., n$ e $c = 1, ..., k$), e θ il vettore contenente tutti i parametri θ_c ($c = 1, ..., k$). Vogliamo allora massimizzare la (3.14) al variare di θ e la (3.12) al variare di $\mathbf{q}$. Il problema di ottimizzazione si scrive:

$$\max_{\theta} \sum_{i=1}^{n} \max_{\mathbf{q}} \sum_{c=1}^{k} q_{i,c} \log\big(\frac{\mu_c \,\mathbb{P}(\mathbf{X} = \mathbf{x}_i \,|\, \theta_c)}{q_{i,c}}\big) \tag{3.15}$$

L'algoritmo *expectation-maximization* risolve il problema (3.15) attraverso la scomposizione dell'ottimizzazione in due step. È un metodo iterativo, la cui velocità di convergenza teorica è piuttosto ridotta. In realtà, nella pratica si osserva una buona rapidità di convergenza.
Lo schema iterativo è il seguente:

- **Expectation:** fissati i θ_c, si risolve (3.12) per i $q_{i,c}$. La soluzione è calcolata attraverso la formula di Bayes (3.6).
- **Maximization:** fissati i $q_{i,c}$, si risolve (3.14) per i θ_c.

2.3 Modello gaussiano

Si possono avere diversi modelli, a seconda di come si definisce la distribuzione di probabilità per $\mathbf{X} \,|\, \theta_c$. Il più usato è il modello gaussiano, che prevede una densità normale:

$$\mathbf{X} \,|\, \theta_c \;\sim\; \mathcal{N}(\theta_c \,,\, \Sigma) \tag{3.16}$$

Supponiamo che la matrice di covarianza Σ sia nota e uguale per tutti i gruppi $c = 1, ..., k$. Il vettore $\theta_c \in \mathbb{R}^p$ è allora la media degli elementi appartenenti al gruppo c. Più precisamente, θ_c è il centro del gruppo c. Abbiamo perciò una generalizzazione del metodo delle k-medie visto in precedenza. La distribuzione per $\mathbf{X} \,|\, \theta_c$ è la seguente:

$$\mathbb{P}(\mathbf{X} = \mathbf{x}|\theta_c) \;\propto\; \exp\{-\frac{1}{2}\,(\theta_c - \mathbf{x})^t\,\Sigma^{-1}\,(\theta_c - \mathbf{x})\}$$

Si possono allora calcolare i risultati degli step di *expectation* e *maximization*. Si ottengono i valori riportati di seguito.

- *Expectation*:

$$q_{i,c} \;=\; \frac{\exp\{\frac{1}{2}(\mathbf{x}_i - \theta_c)^t\,\Sigma^{-1}\,(\mathbf{x}_i - \theta_c)\}}{\sum_l \exp\{\frac{1}{2}(\mathbf{x}_i - \theta_l)^t\,\Sigma^{-1}\,(\mathbf{x}_i - \theta_l)\}}$$

- *Maximization*:

$$\theta_c \;=\; \frac{1}{\sum_{i=1}^n q_{i,c}} \sum_{i=1}^n q_{i,c}\mathbf{x}_i$$

3 Dare significato ai cluster

Come accennato in precedenza, è utile disporre di mezzi per descrivere in modo sintetico il significato dei cluster trovati. In tal modo, si avrebbe una visione immediata della struttura del set di dati. Lavorando con documenti scritti, questa operazione è ancora più importante, e può dare risultati molto informativi, comprensibili per ogni utente. Per la facile leggibilità dei risultati, è importante prevedere nelle applicazioni la possibilità da parte dell'utente di intervenire, se necessario, modificando l'analisi, soprattutto per quanto riguarda la descrizione dei cluster.
Solitamente, la descrizione automatica dei cluster avviene in due modi: con l'estrazione di parole chiave, oppure l'individuazione dei documenti più rappresentativi.

Ricerca di parole chiave

Le parole chiave possono essere estratte valutando degli opportuni indici associati ai token. Sia w_{ij} il peso associato al token j per l'i-esimo documento della collezione (ovvero la componente j-esima del vettore $\mathbf{x}_i$). Come mostrato nel capitolo 1, una scelta possibile per w_{ij} è quella di utilizzare l'indice tf.idf(i, j), definito nella (1.4). Ora, possiamo calcolare l'indice tf.idf non più in relazione ad un documento, ma in relazione ad un cluster c. Adottiamo allora la seguente definizione per l'indice $tf(c, j)$, definito a partire dalla (1.3):

$$tf(c, j) \ := \ \sum_{\mathbf{x}_i \in c} tf(i, j)$$

che equivale a

$$tf(c, j) \ := \ \#\text{ occorrenze del token } j \text{ nel cluster } c$$

Abbiamo allora

$$\text{tf.idf}(c, j) \ := \ tf(c, j) \times idf(j)$$

Possiamo allora selezionare come parole chiave per il cluster c i token j con maggiore indice tf.idf(c, j).
Più semplicemente, spesso si usa scegliere come parole chiave i token che compaiono più volte nei documenti del cluster, escludendo le cosiddette *stopwords* (in genere le congiunzioni, ma anche alcuni verbi e avverbi usati frequentemente senza essere riferiti ad un argomento particolare).

Ricerca di documenti rappresentativi

La seconda strategia per fornire una descrizione del cluster consiste nell'individuare i documenti più rappresentativi. Si scelgono allora i documenti più simili al centro del cluster, che rappresentano "in media" il gruppo. Ma non solo. Per tener conto dell'estensione del cluster, spesso si considerano anche i documenti a coppie. Si scelgono allora le coppie di documenti più dissimili fra loro.

4 Valutazione dei risultati

Come nel caso dell'algoritmo delle k-medie, è possibile valutare la suddivisione in cluster calcolando l'indice di dispersione E definito in (3.1).
È inoltre possibile definire altri indici di dispersione. Dato allora il documento $\mathbf{x}_i$, ne definiamo la similarità con il cluster cui è stato assegnato

indicandola $S(\mathbf{x}_i)$. Tale similarità può essere calcolata come similarità con il centro del cluster, oppure attraverso le definizioni, introdotte per gli algoritmi gerarchici, di similarità tra cluster *single link*, *complete link* o *average link*. Se poi c_i indica il cluster contenente il documento $\mathbf{x}_i$, indichiamo con $MS(\mathbf{x}_i)$ la media delle similarità di tutti i documenti in c_i con il loro cluster, ovvero:

$$MS(\mathbf{x}_i) \ := \ \frac{1}{\#\text{ documenti contenuti in } c_i} \sum_{\mathbf{x}_k \in c_i} S(\mathbf{x}_k)$$

Un altro indice D di dispersione con cui si può valutare la bontà della suddivisione in cluster è allora definito come:

$$D \ := \ \frac{\sum_{i=1}^{n} (S(\mathbf{x}_i) - MS(\mathbf{x}_i))^2}{n}$$

dove n indica, come sempre, il numero di documenti presenti nella collezione.

Mentre E cresce al crescere della distanza dei vettori rispetto ai centri dei cluster, D fornisce una misura dell'uniformità del cluster. Se i documenti di un dato cluster stanno tutti alla stessa distanza dal centro (o meglio, hanno tutti la stessa similarità con il centro), allora l'uniformità è massima e D è minimo. Se invece un cluster contiene sia documenti molto simili che molto dissimili dal centro, allora D sarà grande.

Più gli indici di dispersione D ed E sono piccoli, migliore è il risultato dell'analisi cluster. In generale, per avere una valutazione che tenga conto della composizione originale del set di dati, D ed E sono confrontati con D_0 ed E_0, gli stessi indici calcolati come se tutti i documenti appartenessero ad un unico grande cluster. Si terrà perciò conto dei valori $D_0 - D$ e $E_0 - E$. Come già sottolineato in precedenza, il crescere del numero k di gruppi causa una diminuzione degli indici di dispersione. Tuttavia, valori di k troppo grandi danno origine ad una suddivisione in gruppi poco generale e spesso insensata. Questo aspetto va tenuto presente nella valutazione dei risultati.

5 Applicazioni

Si possono trovare numerosissime applicazioni delle tecniche descritte in questo capitolo.

In primo luogo, l'analisi cluster risulta molto utile per supportare e migliorare i risultati di sistemi di information retrieval. Si pensi ad esempio a quanti risultati si possano ottenere ricercando i documenti che contengono due o tre parole specificate dall'utente in una query. Oltre ad ordinare i documenti trovati per rilevanza, per rendere più agevole l'esplorazione dei risultati può essere utile presentare i documenti raggruppati in base al contenuto.

Non solo, qualora si volesse esplorare una collezione di testi senza effettuare ricerche di termini specifici, è possibile impiegare l'analisi cluster, unita alle tecniche per la descrizione automatica dei gruppi, per realizzare un'*auto-indicizzazione* a più livelli. In parole povere, l'utente potrebbe scegliere in un primo momento tra pochi grandi gruppi di documenti risultato di una prima suddivisione; dopo aver selezionato il gruppo più interessante, l'esplorazione continuerebbe attraverso un'ulteriore suddivisione in cluster e così via. Il termine *auto-indicizzazione* deriva dal fatto che i vari gruppi sono resi riconoscibili grazie a delle "etichette" (dei "titoli") redatte tramite l'estrazione di parole chiave. L'operazione assomiglia quindi alla creazione automatica di un indice che descriva la collezione di documenti.

Un'altra applicazione, già accennata all'inizio del capitolo, consiste nell'individuazione delle tipologie di problemi presentati ad un servizio clienti. Analizzando tutte le richieste, è possibile individuare le problematiche principali per razionalizzare l'organizzazione dell'assistenza ai clienti.

Capitolo 4

Estrazione automatica di informazioni

L'estrazione automatica di informazioni è l'applicazione del text mining che richiede le tecniche più complesse. Essa è strettamente legata alla linguistica, e spesso ricopre ambiti di studio propri del Natural Language Processing. Il suo scopo è quello di individuare e strutturare informazioni precise contenute in un testo.
Le tecniche di estrazione automatica dell'informazione hanno come scopo ultimo la *comprensione* del linguaggio naturale scritto, facilitando la comunicazione uomo-macchina e favorendo il passaggio di dati tra diversi sistemi di informazione.

Esempio introduttivo

Supponiamo di avere a disposizione una collezione di testi che contengono notizie in ambito economico. In tali articoli, tra l'altro, si riportano i cambiamenti nelle posizioni manageriali di spicco. Siamo allora interessati a registrare questi cambiamenti, trascrivendo le informazioni in una base di dati strutturata. Un sistema che compie automaticamente queste operazioni deve perciò essere in grado, dato un testo, di riconoscere l'azienda in questione, il ruolo manageriale, il nome della persona uscente e di quella entrante, per poi registrare i dati in modo strutturato.
Ad esempio, si potrebbe avere il testo seguente:

La Vodafone Italia ha annunciato in una conferenza stampa tenuta lo scorso giovedì che il dott. Mario Rossi, direttore marketing, ha lasciato la sua posizione. Al suo posto è subentrato il l'ing. Sergio Bianchi.

Da questo testo, il sistema deve essere in grado di strutturare le informazioni ricercate, ad esempio compilando la seguente tabella:

Impresa	Voafone Italia
Ruolo	direttore marketing
Uscente	Mario Rossi
Entrante	Sergio Bianchi

A questo punto, è possibile creare delle basi di dati strutturate, che consentano un'esplorazione agevole e che rendano possibile l'applicazione di metodi statistici. Le informazioni, in quanto strutturate, diventano così accessibili ad altri sistemi per ulteriori analisi.

Fasi di estrazione di informazioni

Il campo di studio dell'estrazione automatica di informazioni è strettamente legato all'intelligenza artificiale, in quanto finalizzato alla comprensione del linguaggio naturale. Per comprendere il linguaggio naturale, è necessario rintracciare gli elementi di interesse nel testo e capire le relazioni descritte tra di essi.
In questo lavoro, ci interessiamo a tre diverse operazioni di estrazione di informazioni, alcune delle quali corrispondono ai trattamenti catalogati nel capitolo 1 come *preprocessing*. Le tre fasi sono la ***Named Etities recognition*** (già accennata nella parte riguardante il *preprocessing*), la ***risoluzione di coreferenze*** e l'***estrazione di relazioni***. Vediamo dunque in dettaglio le tre operazioni.

1 Named entity recognition

Come abbiamo già accennato nel Capitolo 1, l'operazione di *named entity recognition* consiste nell'individuare nel testo elementi particolarmente informativi come nomi propri, luoghi, date, organizzazioni e via dicendo. È possibile sfruttare delle informazioni linguistiche a priori, solitamente dizionari contenenti sequenze di parole associate alla loro rispettiva funzione. Tuttavia, in questo lavoro ci occupiamo soprattutto dell'applicazione, in questo ambito, di classificatori supervisionati.

Rispetto alle problematiche affrontate nei capitoli 2 e 3, il punto di partenza dell'analisi cambia. L'unità statistica considerata non sarà più un documento, ma un singolo token del testo. Supponiamo allora di voler riconoscere luoghi, date e nomi di organizzazioni (aziende, enti pubblici, ecc...). Dobbiamo trovare un sistema in grado di classificare ogni token in una categoria

(solitamente detta *type* - tipo) scelta tra *luogo*, *data*, *organizzazione* e *irrilevante*. Spesso, nella letteratura si riportano sistemi in cui si costruisce un classificatore binario indipendente per ogni tipo. In tal caso, il problema della possibile individuazione contemporanea di più tipi è risolto in un secondo momento.

1.1 Chunk recognition

L'operazione descritta è nota in letteratura anche con il nome *chunk recognition* (in inglese *chunk* significa *pezzo*). Il suo scopo è infatti quello di riconoscere in un documento le sequenze di parole (i *pezzi* di testo) che si riferiscono alle entità ricercate.

Solitamente, si richiede che il risultato del processo di *chunk recognition* annoti ogni token con un'etichetta (o meglio, una *tag*) scelta fra quattro possibilità. Citando le annotazioni standard spesso usate dai sistemi esistenti, dato un tipo X, si avrà allora:

- B-X: token iniziale di una sequenza che si riferisce ad un'entità di tipo X
- I-X: token intermedio di una sequenza relativa ad un'entità di tipo X
- F-X: token finale di una sequenza riferita ad un'entità di tipo X
- O: token che non si riferisce ad un'entità di tipo X

Non tutti i sistemi classificano ogni token semplicemente scegliendo tra i tipi a disposizione. In molti casi infatti la classificazione avviene sul token iniziale e quello finale. Si cerca di riconoscere non tanto se un token appartenga ad una named entity di un certo tipo, ma se stia all'inizio o alla fine di una sequenza interessante. Considerando solo il tipo X e seguendo le notazioni riportate sopra, le possibili classi saranno allora B-X, F-X e O.

Si pongono ora i principali problemi per la costruzione dei classificatori. In che modo si può costruire un vettore di feature per ogni token? Quali algoritmi di classificazione supervisionata si adattano meglio al problema? Si affronteranno questi temi nelle prossime sezioni, tralasciando i metodi di classificazione più usati, le Support Vector Machines, che saranno spiegati nel Capitolo 6.

1.2 Scelta delle feature

In un'analisi statistica, la fase più importante e delicata è quella della scelta delle quantità da osservare, che descrivano il fenomeno cui si è interessati nel modo più esaustivo possibile. Quali sono allora le caratteristiche importanti che permettono l'identificazione di una *named entity* a partire da un dato token? Bisognerà tenere conto dei token circostanti, dell'informazione proveniente da eventuali fasi di preprocessing (ad esempio il ruolo grammaticale di una parola), della composizione del token stesso e di alcune informazioni derivanti da conoscenze linguistiche dell'ambito di studio.
Generalmente, le feature osservate sono quelle riportate nel seguente elenco.

- Il token stesso.
- Gli elementi che circondano il token in esame, presenti in una finestra di dimensione fissata. Ad esempio, si possono scegliere i quattro token più vicini (i due prima ed i due dopo). A seconda degli studi, si può considerare o meno l'ordine dei token.
- Poiché spesso gli algoritmi operano processando sequenzialmente un token dopo l'altro, le classificazioni precedenti sono usate per riassumere l'informazione contenuta nei token non direttamente considerati. I tipi precedentemente riconosciuti possono quindi diventare delle feature per il nuovo token.
- Le annotazioni di preprocessing. In generale, le più importanti riguardano il ruolo grammaticale delle parole (*POS tagging*) ed il confronto con dizionari specifici (qualora presenti).
- La presenza di lettere maiuscole, nel token in esame ed in quelli circostanti.
- I prefissi e suffissi delle parole. Fissato un intero m, si possono scegliere come feature i prefissi e suffissi del token da 1 ad m lettere.
- Delle conoscenze linguistiche più approfondite possono essere affiancate all'approccio statistico. Se sono state determinate delle regole per individuare alcune *named entity* e codificate in forma di *decision rules*, è possibile utilizzarle come delle feature binarie (valore 1 se la regola è soddisfatta, 0 altrimenti). Esse rivestiranno particolare importanza nella classificazione.

Tutte le feature descritte corrispondono a variabili categoriche. Assumono perciò valore all'interno di insiemi finiti. Per meglio visualizzare la creazione di feature, facciamo un esempio. Analizziamo la seguente frase in inglese:

EU rejects ***German*** *call to boycott British lamb.*

Costruiamo il vettore di feature corrispondente al token *German*. Il termine *posizione* si riferisce alla posizione relativa rispetto al token considerato (ad esempio, *posizione -1* si riferisce al token precedente, in questo caso *rejects*).

Feature	**Valore**
Token	German
Token in posizione -1	rejects
Token in posizione -2	EU
Token in posizione +1	call
Token in posizione +2	to
Tipo in posizione -1	ininfluente
Tipo in posizione -2	Organizzazione
Iniziale maiuscola	Sì
Tutte maiuscole	No
Iniziale maiuscola in posizione -1	No
Iniziale maiuscola in posizione +1	No
Prefisso 1 carattere	G
Prefisso 2 caratteri	Ge
Prefisso 3 caratteri	Ger
Suffisso 1 carattere	n
Suffisso 2 caratteri	an
Suffisso 3 caratteri	man
POS	aggettivo

Rappresentazione numerica per vettori di feature categoriche

Come detto in precedenza, le componenti del vettore di feature sono, sostanzialmente, delle variabili categoriche o binarie. È possibile perciò trovare una codifica di tale vettore in termini di variabili binarie. Definiamo $\mathbf{x}_{bin}$ il nuovo vettore costruito con componenti in $\{0, 1\}$. Si avrà allora $\mathbf{x}_{bin} \in \{0, 1\}^p$, con p molto grande, maggiore rispetto alla dimensione che si ottiene costruendo i vettori-documento trattati nei capitoli precedenti.

I pesi binari possono eventualmente essere modificati, nel caso di feature basate su token in posizioni relative. In tal caso, più la posizione relativa è

distante, più il peso assegnato alla feature sarà ridotto. Facciamo un esempio. Nella tabella precedente, si considerano i token in posizione ± 1 (*call*, *rejects*) e ± 2 (*to*, *EU*). Grazie alla codifica binaria, questi token osservati daranno luogo a delle componenti non nulle nel vettore $\mathbf{x}_{bin}$. Possiamo allora moltiplicare le componenti non nulle relative a *to* ed *EU* per un coefficiente μ_2 e, in modo analogo, le componenti non nulle relative a *call* e *rejects* per un coefficiente μ_1. Lo stesso vale per tutte le componenti date da feature osservate in posizioni relative (rispetto all'esempio precedente, il tipo nelle posizioni -1 e -2 e l'iniziale maiuscola nelle posizioni ± 1).
In generale, se si hanno feature osservate nelle posizioni relative $\pm 1, ..., \pm k$, si avranno k coefficienti $0 < \mu_k < \mu_{k-1} < ... < \mu_1 \leq 1$. In questo modo, il vettore $\mathbf{x}_{bin}$ sarà trasformato nel nuovo vettore $\mathbf{x} \in [0,1]^p$.
Una scelta molto diffusa è quella di porre $\mu_i = \frac{1}{i}$, per $i = 1, ..., k$.

Passiamo ora alla descrizione di un algoritmo di classificazione molto usato nell'ambito della *named entity recognition* e, più in generale, per l'estrazione di informazioni: il metodo di *massima entropia*.

1.3 Metodo di massima entropia

Riprendendo quanto descritto nei capitoli 2 e 3, ricordiamo le notazioni per definire un modello di classificazione supervisionata. Sia allora $\mathbf{X}$ il vettore aleatorio relativo ad un token ed $\mathbf{x}$ una sua realizzazione. Sia k il numero di categorie (in questo caso i tipi di *named entity*) in cui classifichiamo i token. Sia Y la variabile aleatoria a valori in $\{1, ..., k\}$ che indica il tipo associato ad un token. Una sua realizzazione sarà indicata con y. Come per ogni problema di classificazione supervisionata, supponiamo di disporre di un insieme di testi annotati a mano, che compongono il training set. Ogni elemento $\mathbf{x}_i$ del training set sarà allora annotato con il suo tipo y_i.

Lo scopo del modello è quello di trovare la categoria $g \in \{1, ..., k\}$ per cui, avendo osservato le feature $\mathbf{x}$, la probabilità $\mathbb{P}(Y = g | \mathbf{X} = \mathbf{x})$ sia massima. Esistono due tipi di modelli:

- **Generative Model** - prevedono il calcolo di $\mathbb{P}(\mathbf{X} = \mathbf{x} | Y = g)$, e ricavano la probabilità ricercata attraverso la formula di Bayes.
- **Discriminative Model** - formulano direttamente una stima per $\mathbb{P}(Y = g | \mathbf{X} = \mathbf{x})$, senza bisogno del calcolo esplicito di $\mathbb{P}(\mathbf{X} = \mathbf{x} | Y = g)$.

Una delle maggiori limitazioni dei modelli visti nel capitolo 2 era l'ipotesi di indipendenza delle componenti di $\mathbf{X}$. Tale ipotesi, nonostante spesso non

soddisfatta, era necessaria per trovare una stima di $\mathbb{P}(\mathbf{X} = \mathbf{x}|Y = g)$. Nel caso di un modello di tipo *discriminative*, sebbene si possa utilizzare formalmente l'ipotesi di indipendenza per ricavarlo, non è richiesto il calcolo esplicito della probabilità congiunta tramite prodotto delle marginali (che andrebbero stimate grazie al training set). Sarà dunque interessante impiegare modelli di questo tipo, uno dei quali è proprio il *metodo di massima entropia*, per limitare l'inaccuratezza dovuta a tale calcolo. Ribadiamo che, sebbene il modello sia ricavato a partire dall'assunzione di indipendenza, esso non richiede il calcolo esplicito di $\mathbb{P}(\mathbf{X} = \mathbf{x}|Y = g)$ come prodotto delle $\mathbb{P}(X_j = x_j|Y = g)$.

Per definire il *metodo di massima entropia*, riprendiamo le definizioni introdotte nel capitolo 2 per i vettori $\mathbf{w}^{(g)}$ (2.8) ed i valori reali $b^{(g)}$ (2.9). Nel caso di indipendenza delle componenti di $\mathbf{X}$, poiché abbiamo solo valori in $\{0, 1\}$, possiamo scrivere:

$$\mathbb{P}(\mathbf{X} = \mathbf{x}|Y = g) \;=\; \prod_{j=1}^{p} \Big(\frac{\mathbb{P}(X_j = 1|Y = g)}{\mathbb{P}(X_j = 0|Y = g)} \Big)^{x_j} \mathbb{P}(X_j = 0|Y = g) \quad (4.1)$$

Sfruttando allora la definizione di $\mathbf{w}^{(g)}$, riscriviamo la precedente espressione e svolgiamo qualche calcolo:

$$\mathbb{P}(\mathbf{X} = \mathbf{x}|Y = g) \;=\; \prod_{j=1}^{p} \Big(\exp\{w_j^{(g)} x_j\} \, \mathbb{P}(X_j = 0|Y = g) \Big) \;=$$

$$= \prod_{j=1}^{p} \frac{\exp\{w_j^{(g)} x_j\}}{\Big(\dfrac{1}{\mathbb{P}(x_j = 0|Y = g)} \Big)} = \prod_{j=1}^{p} \frac{\exp\{w_j^{(g)} x_j\}}{\Big(\dfrac{\mathbb{P}(x_j = 0|Y = g) + \mathbb{P}(x_j = 1|Y = g)}{\mathbb{P}(x_j = 0|Y = g)} \Big)} =$$

$$= \prod_{j=1}^{p} \frac{\exp\{w_j^{(g)} x_j\}}{1 + \exp\{w_j^{(g)}\}} \;=\; \frac{\exp\{\mathbf{w}^{(g)} \cdot \mathbf{x}\}}{\prod_{j=1}^{p} \big(1 + \exp\{w_j^{(g)}\} \big)} \quad (4.2)$$

Introduciamo ora il termine $t(\mathbf{w}^{(g)})$, definito come

$$t(\mathbf{w}^{(g)}) \;:=\; \prod_{j=1}^{p} \big(1 + \exp\{w_j^{(g)}\} \big) \quad (4.3)$$

Possiamo allora esprimere $b^{(g)}$ attraverso la (4.3), ottenendo

$$b^{(g)} = \log(\mathbb{P}(Y = g)) - \log(t(\mathbf{w}^{(g)})) \quad (4.4)$$

Per la formula delle probabilità totali, vale

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) \;=\; \sum_{g=1}^{k} \mathbb{P}(Y = g)(\mathbf{X} = \mathbf{x}|Y = g)$$

da cui, inserendo i termini calcolati in precedenza, si ottiene

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) \;=\; \sum_{g=1}^{k} \exp\{\mathbf{w}^{(g)} \cdot \mathbf{x} + b^{(g)}\}$$

Infine, dalla formula di Bayes vale

$$\mathbb{P}(Y = g|\mathbf{X} = \mathbf{x}) \;=\; \frac{\mathbb{P}(Y = g)\mathbb{P}(\mathbf{X} = \mathbf{x}|Y = g)}{\mathbb{P}(\mathbf{X} = \mathbf{x})} \;=$$

$$= \frac{\exp(\mathbf{w}^{(g)} \cdot \mathbf{x} + b^{(g)})}{\sum_{s=1}^{k} \exp(\mathbf{w}^{(s)} \cdot \mathbf{x} + b^{(s)})} \tag{4.5}$$

Abbiamo ottenuto quindi un espressione di $\mathbb{P}(Y = g|\mathbf{X} = \mathbf{x})$ in funzione dei vettori $\mathbf{w}^{(g)}$ e dei valori $b^{(g)}$. La classificazione non richiede quindi la stima esplicita delle probabilità $\mathbb{P}(X_j = x_j|Y = g)$.
Restano da calcolare i $\mathbf{w}^{(g)}$ ed i $b^{(g)}$. Per farlo, si può impiegare una stima di massima verosimiglianza con i dati osservati nel training set. Siano allora $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$ i dati del training set. Le stime $(\hat{\mathbf{w}}^{(g)}, \hat{b}^{(g)})$ saranno allora calcolate risolvendo:

$$(\hat{\mathbf{w}}^{(1)}, \hat{b}^{(1)}), ..., (\hat{\mathbf{w}}^{(k)}, \hat{b}^{(k)}) \;=$$

$$= \arg \max_{(\mathbf{w}^{(1)}, b^{(1)}), ..., (\mathbf{w}^{(k)}, b^{(k)})} \;\prod_{i=1}^{n} \frac{\exp\{\mathbf{w}^{(y_i)} \cdot \mathbf{x}_i + b^{(y_i)}\}}{\sum_{s=1}^{k} \exp(\mathbf{w}^{(s)} \cdot \mathbf{x}_i + b^{(s)})}$$

Solitamente, si preferisce riscrivere il problema di ottimizzazione applicando il logaritmo, ottenendo il problema di minimo seguente:

$$(\hat{\mathbf{w}}^{(1)}, \hat{b}^{(1)}), ..., (\hat{\mathbf{w}}^{(k)}, \hat{b}^{(k)}) \;=$$

$$= \arg \min_{(\mathbf{w}^{(1)}, b^{(1)}), ..., (\mathbf{w}^{(k)}, b^{(k)})} \sum_{i=1}^{n} \left[-(\mathbf{w}^{(y_i)} \cdot \mathbf{x}_i + b^{(y_i)}) + \log(\sum_{s=1}^{k} \exp\{\mathbf{w}^{(s)} \cdot \mathbf{x}_i + b^{(s)}\}) \right]$$

Ora, la soluzione del problema è definita a meno di costanti. Possiamo allora imporre un vincolo ulteriore, ad esempio

$$\sum_{g=1}^{k} \mathbf{w}^{(g)} \;=\; \mathbf{0} \qquad \sum_{g=1}^{k} b^{(g)} \;=\; 0$$

Per scrivere in forma più compatta il problema di ottimizzazione, possiamo introdurre un vettore generale $\mathbf{v}$, contenente tutti i $\mathbf{w}^{(g)}$ ed i $b^{(g)}$:

$$\mathbf{v} = \begin{bmatrix} \mathbf{w}^{(1)} \\ b^{(1)} \\ \mathbf{w}^{(2)} \\ b^{(2)} \\ \vdots \\ \mathbf{w}^{(k)} \\ b^{(k)} \end{bmatrix}$$

Analogamente, riscriviamo i dati attraverso i vettori $\{ \mathbf{z}_{1,1}, \mathbf{z}_{1,2}, ..., \mathbf{z}_{1,k}, \mathbf{z}_{2,1}, ..., \mathbf{z}_{n,k} \}$, con $\mathbf{z}_{i,j} \in \mathbb{R}^{(p+1)k}$, definiti come

$$\mathbf{z}_{i,1} = \begin{bmatrix} \mathbf{x}_i \\ 1 \\ \left.\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right\} (p+1)(k-1) \text{ zeri} \end{bmatrix} \qquad \mathbf{z}_{i,2} = \begin{bmatrix} \left.\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right\} p+1 \text{ zeri} \\ \mathbf{x}_i \\ 1 \\ \left.\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right\} (p+1)(k-2) \text{ zeri} \end{bmatrix}$$

$$\mathbf{z}_{i,g} = \begin{bmatrix} \left.\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right\} (p+1)(g-1) \text{ zeri} \\ \mathbf{x}_i \\ 1 \\ \left.\begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}\right\} (p+1)(k-g) \text{ zeri} \end{bmatrix}$$

Riscriviamo allora il problema di ottimizzazione in funzione di $\mathbf{v}$ e dei dati $\mathbf{z}_{i,g}$, cercando la stima $\hat{\mathbf{v}}$ definita come:

$$\hat{\mathbf{v}} = \arg\min_{\mathbf{v}} \sum_{i=1}^{n} \left[-\mathbf{v} \cdot \mathbf{z}_{i,y_i} + \log\left(\sum_{s=1}^{k} \exp(\mathbf{v} \cdot \mathbf{z}_{i,s}) \right) \right] \tag{4.6}$$

Esistono diverse tecniche per risolvere (4.6). Un algoritmo spesso usato è il GIS (*Generalized Iterative Scaling*). Per una descrizione dell'algoritmo si veda ad esempio [4]. Per maggiori approfondimenti sul metodo di massima entropia applicato a problemi di text mining, si rimanda a [2].

1.4 Modelli a probabilità sequenziale

Fino ad ora, abbiamo presentato un metodo che, dato un token $\mathbf{x}$, assegna un tipo g stimando $\mathbf{P}(Y = g|\mathbf{X} = \mathbf{x})$. I token sono processati sequenzialmente, uno dopo l'altro, nell'ordine in cui si presentano nel testo. Abbiamo visto che, nel costruire il vettore di feature, si può tenere conto delle assegnazioni precedenti. È infatti ragionevole ipotizzare che l'ordine con sui si presentano i tipi sia importante. Ad esempio, è molto improbabile che tre date diverse si trovino una di seguito all'altra, senza essere separate da token di tipo diverso.
Un miglioramento al modello presentato in precedenza per il riconoscimento delle *named entity* consiste allora nel considerare non più un singolo token alla volta, ma una sequenza di t token $[\mathbf{x}^1, ..., \mathbf{x}^t]$. Si stimerà allora la probabilità della sequenza dei tipi da assegnare $\mathbb{P}([g^1, ..., g^t] \,|\, [\mathbf{x}^1, ..., \mathbf{x}^t])$.
Modelli di questo tipo richiedono tecniche più sofisticate, e si appoggiano in generale a metodi di programmazione dinamica. Non entriamo nei dettagli dei modelli a probabilità sequenziale. Rimandiamo a [14] per una trattazione più approfondita.

2 Risoluzione di coreferenze

La risoluzione di coreferenze (*coreference resolution*[1]) è una fase cruciale per la costruzione di sistemi in grado di *comprendere* il linguaggio naturale. Lo scopo è quello di individuare tutti i token di un testo che si riferiscono alla stessa entità. Generalmente, si considerano i pronomi, individuati grazie alla fase di *POS tagging*. Facciamo un esempio:

Assistiamo ad una conferenza del prof. Rossi, i ***cui*** *studi trattano di ...*

In questo caso, lo scopo dell'analisi è di mettere in relazione il pronome ***cui*** con il nome cui si riferisce, cioè ***prof. Rossi***.
In generale, la risoluzione di coreferenze è un processo che avviene dopo aver riconosciuto le named entity di interesse. In questo modo, si cercherà di assegnare i pronomi alle entity riconosciute in precedenza.

Si può affinare ulteriormente l'analisi, e cercare i legami anche tra le named entity. Nello stesso testo si possono infatti trovare named entity scritte diversamente, che si riferiscono alla stessa entità. Ad esempio, *States*, *U.S.A.* e *Stati Uniti* si riferiscono alla stessa entità. Per trovare questi legami, sarà

[1] Nell'ambito del Natural Language Processing, la risoluzione di coreferenze applicata ai pronomi è detta *anaphora resolution*.

necessario disporre di dizionari specifici. Ma non solo, possiamo infatti impiegare metodi statistici. Questo vale in modo particolare per il trattamento dei pronomi.

Classi di equivalenza

Una volta individuati nel testo i pronomi e le named entity di interesse, l'analisi porta a suddividerli in gruppi, in modo che ciascun gruppo si riferisca ad una precisa entità. Si creano cioè delle classi di equivalenza con gli elementi di interesse.
La scelta naturale per realizzare un sistema di *coreference resolution* sembrerebbe quindi l'applicazione di un'analisi cluster, in grado di individuare automaticamente dei gruppi che si riferiscono alle entità. Tuttavia, la difficoltà nella scelta del numero dei gruppi e le peculiarità legate alla successione temporale delle parole rende questa scelta non conveniente. I metodi descritti maggiormente in letteratura sono quindi di natura diversa. Tali metodi prevedono che gli elementi di interesse siano confrontati a coppie, e che per ogni coppia valutata si fornisca una risposta alla domanda *questi due elementi si riferiscono alla stessa entità?*
Si costruiscono quindi dei classificatori supervisionati binari sulle coppie di elementi per fornire la risposta cercata. Più precisamente, dato un elemento, lo si può confrontare con gli elementi precedenti nel testo, applicando la classificazione ad ogni coppia.

Come evitare allora che un pronome sia assegnato a più entità contemporaneamente? È necessario adottare una procedura che impedisca assegnazioni multiple, e che garantisca la formazione di classi di equivalenza. Descriviamo nel seguito un esempio di procedura. Supponiamo di aver individuato in un testo gli elementi $\mathbf{x}_1, ..., \mathbf{x}_n$. Supponiamo che, se si è formato un gruppo g composto dagli elementi $\mathbf{x}_1^g, ..., \mathbf{x}_{n_g}^g$, sia possibile calcolare un indice di *pertinenza* $p(\mathbf{x}_i, g)$ di un generico elemento $\mathbf{x}_i$ con il gruppo g. Daremo in seguito un'indicazione su come definire tale indice. Una possibile procedura per la *coreference resolution* è allora la seguente:

Inizializzazione: l'elemento $\mathbf{x}_1$ è assegnato all'unico gruppo g_1.

Passo i-esimo: supponiamo di aver individuato i gruppi $g_1, ..., g_k$. Allora per $\mathbf{x}_i$:

1. Per ogni gruppo g_j, $j \in \{1, ..., k\}$, si calcola l'*indice di pertinenza* $p(\mathbf{x}_i, g_j)$ dell'elemento $\mathbf{x}_i$ con il gruppo g_j.

2. Se $\forall j \; p(\mathbf{x}_i, g_j)$ è inferiore ad una certa soglia S, allora creiamo un nuovo gruppo g_{k+1} costituito dal solo $\mathbf{x}_i$
3. Se per qualche j si ha $p(\mathbf{x}_i, g_j) \geq S$, allora si assegna $\mathbf{x}_i$ al gruppo con indice di pertinenza massimo.

Indice di pertinenza

Per definire l'indice di pertinenza, possiamo sfruttare i classificatori binari per le coppie di parole. Supponiamo di voler calcolare $p(\mathbf{x}, g)$, dove g è composto dagli elementi $\mathbf{x}_1^g, ..., \mathbf{x}_{n_g}^g$. Per ogni elemento $\mathbf{x}_j^g$, consideriamo la coppia $(\mathbf{x}, \mathbf{x}_j^g)$. Il classificatore fornirà allora una risposta positiva o negativa, a seconda che i due elementi siano considerati in relazione o meno. Si può allora definire $p(\mathbf{x}, g)$ come la proporzione di risposte positive. Per un calcolo più preciso, si può tenere conto non tanto della risposta positiva o negativa del classificatore, quanto di un "punteggio" assegnato ad ogni coppia. Ad esempio, si può usare come indice la funzione f presente nella definizione (2.13) e (2.14) di un classificatore binario lineare.

Costruzione di vettori di feature

L'unità statistica considerata è la coppia di elementi. Per costruire un classificatore supervisionato, è quindi necessario definire un vettore di feature per ogni coppia.
Molte delle feature consistono nella verifica di regole linguistiche (*decision rules*, riguardanti ad esempio la declinazione), o al confronto con dizionari specifici. Si dovrà tenere conto anche della posizione relativa degli elementi (quante parole ci sono tra i due elementi, quante frasi) e del ruolo grammaticale dato dalla fase di *POS tagging*. Si possono poi usare i token che compongono gli elementi ed, eventualmente, i prefissi e suffissi.
In genere, tuttavia, le regole linguistiche sono di gran lunga le feature più rilevanti.

3 Estrazione di relazioni

L'operazione di *coreference resolution* si occupa di ricercare le relazioni di equivalenza tra i token. Possiamo immaginare di estendere questo concetto, ricercando relazioni di qualsiasi tipo. Un'analisi delle relazioni tra i token permette di individuare automaticamente informazioni dettagliate, dando una conoscenza approfondita del testo. In generale, non si considerano tutti i token, ma solo le *named entity* individuate. Inoltre, le informazioni trovate

saranno più precise, se si dispone dei risultati del processo di *coreference resolution*.

Supponiamo di dover completare una base di dati contenente dei nominativi, aggiungendo delle informazioni ad essi associate, e che queste informazioni si trovino in una collezione di documenti di testo. Ad esempio, potremmo interessarci ad individuare l'azienda datore di lavoro di ogni persona presente nel database. Per farlo, sarà necessario definire la relazione "*è impiegato presso*" e ricercare le coppie di token che la verificano. Consideriamo la frase seguente:

Mario Bianchi, capo del dipartimento R&D di Fiat, sostiene che i fondi europei per lo sviluppo dei motori a idrogeno verranno interrotti. Alla Volkswagen sono in disaccordo: Hans Schwartz, direttore tecnico, è convinto che l'UE aumenterà i finanziamenti nel campo dell'idrogeno.

Processando questa frase, il sistema deve essere in grado di riempire la seguente tabella:

Entity 1	**Relazione**	**Entity 2**
Mario Bianchi	*è impiegato presso*	Fiat
Hans Schwartz	*è impiegato presso*	Volkswagen

3.1 Tecniche statistiche

Come nel caso della *coreference resolution*, si possono costruire dei classificatori binari che agiscono su coppie di *entity*. Ancora una volta, la coppia sarà l'unità statistica e si dovrà costruire un vettore di feature. Le feature impiegate saranno simili a quelle relative alla risoluzione di coreferenze, ma potrebbero essere necessari più dettagli. In particolare, l'informazione data dalla fase di *parsing* (analisi logica e del periodo) può risultare fondamentale. Essa però comporta delle difficoltà. L'output di un sistema di *parsing* è infatti costituito da un albero (l'*albero di parsing*). La codifica delle informazioni contenute nell'albero attraverso un vettore di feature può risultare non immediata. Non entriamo nei dettagli della costruzione di tali feature (anche perché sarebbe richiesta un'esposizione sul formato e la costruzione dell'albero di parsing).

Un'altra componente imprescindibile è la definizione di *decision rules* linguistiche. Essendo il problema molto complesso, risolverlo senza decision rules richiederebbe un training set annotato troppo esteso. L'apporto di conoscenza esterna permette quindi di "facilitare l'apprendimento" nell'applicazione di machine learning.

3.2 Template filling

Lo scopo dell'estrazione di relazioni è quello di strutturare l'informazione presente in un testo. Da semplici stringhe di caratteri si vuole ricavare informazione organizzata, rappresentandola in strutture prefissate come tabelle, o meglio *templates*. L'output di un sistema di estrazione di relazioni consisterà dunque nella compilazione di opportuni *templates*, definiti in funzione dell'analisi. Inoltre, una relazione può essere arricchita da informazioni supplementari. In tal caso, il template prevederà dei campi per registrare tali informazioni. Spesso inoltre, quando la descrizione diventa più precisa, non si parla più di *relazione*, ma di *evento*.

Facciamo un esempio. Supponiamo di voler individuare nel testo la descrizione di incontri avvenuti tra due persone. Oltre ai nomi delle persone, si può essere interessati ai dettagli dell'incontro, ad esempio la data ed il luogo. Il template da riempire in tal caso sarà della forma:

Evento	**Persona 1**	**Persona 2**	**Data**	**Luogo**
Incontro				

Per ritrovare tutte le informazioni, sarà necessario applicare diverse tecniche contemporaneamente, in particolare la ricerca di relazioni ed il riconoscimento delle *named entity*.

4 Applicazioni

Le possibili applicazioni della ricerca automatica di informazioni in testi scritti sono vastissime. Tutti i sistemi di riconoscimento del linguaggio naturale si basano su di esse. Recentemente, sono sempre più diffusi gli studi per automatizzare servizi di assistenza clienti, trattando sia e-mail che chiamate verso i call center. In quest'ultimo caso, la voce dell'utente è analizzata e tradotta in testo. Da esso, si estraggono le informazioni necessarie ad indirizzare l'utente verso un operatore in grado di rispondere alla domanda posta. Studi più avanzati si pongono l'obiettivo di creare sistemi in grado di fornire risposte automatiche alle richieste di un utente.

Tecniche per il riconoscimento del linguaggio naturale sono state usate anche nell'ambito giuridico-investigativo. Ad esempio, nell'ambito di un progetto voluto dal dipartimento di Giustizia americano, si è costruito un sistema in grado di analizzare rapporti giudiziari e compilare basi di dati con le informazioni ritrovate. Analogamente, l'università dell'Arizona ha realizzato uno studio in cui, partendo dai rapporti di polizia relativi a casi di droga, si è creato un database con informazioni come nomi, crimini commessi, quantità e tipo di droga sequestrata, ecc...

L'estrazione automatica di informazioni ha permesso anche la creazione di database a partire da annunci di vendita. Ad esempio, se si è interessati agli annunci di vendita di automobili, è possibile registrare caratteristiche come modello, anno, prezzo, colore, ecc... Tali informazioni permettono all'utente una più facile ricerca, la comparazione automatica e, qualora si analizzino annunci provenienti da diversi siti, offrono la possibilità di individuare le offerte più convenienti. È inoltre interessante applicare tecniche simili adottando più lingue contemporaneamente. Si potrebbero quindi comparare gli annunci di vendita su scala europea, permettendo una maggiore concorrenza e moltiplicando le opportunità di acquisto.

Molte applicazioni provengono inoltre dal campo biomedico. Nell'introduzione abbiamo citato i sistemi automatici per individuare le relazioni presenti tra determinate proteine, analizzando pubblicazioni scientifiche. In tale applicazione, il text mining diventa un vero e proprio supporto alla conoscenza scientifica, affiancando i ricercatori nello studio delle relazioni tra proteine. Restando sempre nell'ambito biomedico, nel Capitolo 6 sarà descritto il problema riguardante il riconoscimento di elementi diagnostici in lettere di dimissione ospedaliera redatte da medici.

Capitolo 5

Support Vector Machines

Le Support Vector Machines (SVM) sono attualmente tra le più diffuse tecniche di classificazione supervisionata. Vengono molto usate in sistemi di riconoscimento di immagini, di suoni e di linguaggio naturale. In particolare, nel text mining si applicano soprattutto per l'estrazione di informazioni.

Poiché sono state impiegate nel trattamento delle lettere di dimissione ospedaliera, oggetto dei prossimi capitoli, vengono presentate in maniera più approfondita rispetto agli altri metodi descritti in precedenza. Per un ulteriore approfondimento si rimanda a [13].

1 Idea generale

Una SVM è, in sostanza, un classificatore supervisionato binario. I dati del training set sono descritti tramite i vettori di feature $\{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \mathbb{R}^p$, e sono divisi in due categorie. Per ogni dato $\mathbf{x}_i$, la categoria è espressa tramite il valore $y_i \in \{-1, 1\}$. Diciamo allora che $\mathbf{x}_i$ è un dato positivo se $y_i = 1$, negativo se $y_i = -1$.

L'idea di partenza è quella di trovare un iperpiano di $\mathbb{R}^p$ che separi i dati del training set positivi da quelli negativi. Tale iperpiano sarà poi usato per classificare nuovi dati, di cui non si conosce la categoria. Ovviamente, non sempre i dati del training set sono separabili da un iperpiano. È necessario dunque introdurre opportune generalizzazioni, che prevedano la costruzione dell'iperpiano di separazione permettendo degli errori rispetto al training set. Generalizzando ulteriormente, si possono considerare iper-superfici di separazione non lineari.

A differenza di altri modelli usati per l'estrazione di informazioni, le SVM non richiedono una descrizione probabilistica dei processi di generazione dei dati e di classificazione. In precedenza infatti, avevamo supposto che i vettori del training set $\mathbf{x}_i$ e le y_i fossero realizzazioni rispettivamente delle variabili aleatorie $\mathbf{X}$ e Y. Si cercava poi di stimare la probabilità condizionale $\mathbb{P}(Y = y|\mathbf{X} = \mathbf{x})$. In questo caso invece, il modello alla base delle SVM è puramente geometrico-algebrico. Questo fatto può essere determinante soprattutto per trattare dati sparsi e per i quali $p \gg n$, come nel caso del text mining. Tali condizioni rendono infatti difficile la stima delle distribuzioni di probabilità per le variabili aleatorie $\mathbf{X}$ e Y, difficoltà che non sussiste nel caso di un modello non probabilistico.

2 Dati separabili

Vediamo innanzi tutto come il metodo SVM permette di trovare un iperpiano di separazione nel caso in cui i dati del training set positivi siano separabili da quelli negativi.

2.1 Vettori di supporto

Un iperpiano $\mathcal{H}$ di $\mathbb{R}^p$ è definito tramite un vettore $\mathbf{w} \in \mathbb{R}^p$ ed un valore di *bias* $b \in \mathbb{R}$. Il luogo dei punti appartenenti all'iperpiano è allora esprimibile come

$$\mathcal{H} = \{\, \mathbf{x} \in \mathbb{R}^p \mid \mathbf{w} \cdot \mathbf{x} + b = 0 \,\}$$

Si noti come i valori di $\mathbf{w}$ e b siano definiti a meno di una costante reale. Infatti, vale

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \iff \eta\mathbf{w} \cdot \mathbf{x} + \eta b = 0\,, \qquad \forall \eta \in \mathbb{R} \setminus \{0\}$$

Supponiamo allora di avere già individuato l'iperpiano $\mathcal{H}$, definito attraverso il vettore $\mathbf{w}$ ed il valore di bias b. Sia $\mathbf{x}_{new}$ una nuova osservazione, che vogliamo classificare assegnando un valore a y_{new}. Si avrà allora:

$$y_{new} = \begin{cases} +1 & \text{se} \quad \mathbf{w} \cdot \mathbf{x}_{new} + b > 0 \\ -1 & \text{se} \quad \mathbf{w} \cdot \mathbf{x}_{new} + b < 0 \end{cases}$$

Ora, stiamo considerando il caso particolare in cui i dati positivi del training set sono perfettamente separati da quelli negativi, perciò devono essere classificati correttamente. Deve allora valere

$$y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) > 0 \qquad \forall i = 1, ..., n$$

Un valore elevato di $|\mathbf{w} \cdot \mathbf{x}_i + b|$ indica che il vettore $\mathbf{x}_i$ è lontano dall'iperpiano di separazione. I dati più critici del training set saranno allora quelli più vicini all'iperpiano, cioè gli $\mathbf{x}_i$ per cui $|\mathbf{w} \cdot \mathbf{x}_i + b|$ è minimo. Per fissare un unico valore di $\mathbf{w}$ e b, si può allora richiedere che i vettori del training set più vicini all'iperpiano di separazione siano tali che $|\mathbf{w} \cdot \mathbf{x}_i + b| = 1$. In altri termini, si impone la seguente condizione:

$$\min_{i=1,\ldots,n} |\, \mathbf{w} \cdot \mathbf{x}_i \, + \, b \,| \; = \; 1 \tag{5.1}$$

L'iperpiano che soddisfa la (5.1) è detto *iperpiano canonico*, mentre i vettori $\mathbf{x}_i$ per cui vale $|\mathbf{w} \cdot \mathbf{x}_i + b| = 1$ sono detti *vettori di supporto*.

2.2 Massimizzazione del margine

Affinché la classificazione sui nuovi dati sia più efficace, l'iperpiano $\mathcal{H}$ non deve solo separare i dati del training set, ma deve farlo nel miglior modo possibile. Cosa significa *separare i dati del training set nel miglior modo possibile*? Indichiamo con $d(\mathbf{x}_i, \mathcal{H})$ la distanza del vettore $\mathbf{x}_i$ dall'iperpiano $\mathcal{H}$. Una richiesta ragionevole potrebbe essere quella di massimizzare la somma dei quadrati delle distanze dall'iperpiano, ovvero cercare l'iperpiano ottimo $\hat{\mathcal{H}}$ che soddisfi:

$$\hat{\mathcal{H}} \; = \; \arg \min_{\mathcal{H}} \sum_{i=1}^{n} d(\mathbf{x}_i, \mathcal{H})^2 \tag{5.2}$$

L'unicità della soluzione sarebbe poi garantita imponendo la (5.1).

La condizione (5.2) si rivela poco interessante, poiché richiede un notevole sforzo computazionale e genera classificatori poco predittivi. Cambiamo allora punto di vista. Supponiamo di considerare, tra tutti gli iperpiani possibili, solamente quelli canonici. Ci interessiamo alla distanza dell'iperpiano dai soli vettori di supporto che corrispondono, come detto in precedenza, ai dati più critici. Cercheremo allora l'iperpiano di separazione $\hat{\mathcal{H}}$, di equazione $\hat{\mathbf{w}} \cdot \mathbf{x} + \hat{b} = 0$, che massimizza la distanza dai vettori di supporto. Ricordiamo che la distanza del vettore $\mathbf{x}$ dall'iperpiano $\mathcal{H}$ di equazione $\mathbf{w} \cdot \mathbf{x} + b = 0$ è data dalla seguente formula:

$$d(\mathbf{x}, \mathcal{H}) \; = \; \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|} \tag{5.3}$$

Se $\mathbf{x}_i$ è un vettore di supporto e l'iperpiano $\mathcal{H}$ è in forma canonica, vale allora

$$d(\mathbf{x}_i, \mathcal{H}) \; = \; \frac{1}{\|\mathbf{w}\|} \tag{5.4}$$

La distanza dai vettori di supporto è detta *margine*. Dalla (5.4) deduciamo quindi che massimizzare il margine per un iperpiano canonico equivale a minimizzare la norma $\|\mathbf{w}\|$. I valori ottimi $\hat{\mathbf{w}}$ e $\hat{b}$ sono allora soluzione del seguente problema di ottimizzazione vincolata:

$$\begin{cases} (\hat{\mathbf{w}}, \hat{b}) \;=\; \arg\min_{(\mathbf{w},b)} \; \dfrac{1}{2}\,\|\mathbf{w}\|^2 \\ \\ \text{s.v.} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \;\geq 0 \quad \forall i = 1, ..., n \qquad \text{(separazione)} \\ \qquad\; \min_i y_i(\mathbf{w}\cdot\mathbf{x}_i + b) = 1 \qquad \text{(iperpiano canonico)} \end{cases} \tag{5.5}$$

Abbiamo scelto come funzione obiettivo il quadrato della norma per ottenere un problema quadratico, più semplice da risolvere. Il secondo vincolo in (5.5) è scritto in una forma particolare, atipica per un problema di ottimizzazione vincolata. Possiamo riformulare il problema sostituendo i due vincoli con un'unica richiesta, ottenendo:

$$\begin{cases} (\hat{\mathbf{w}}, \hat{b}) \;=\; \arg\min_{(\mathbf{w},b)} \; \dfrac{1}{2}\,\|\mathbf{w}\|^2 \\ \\ \text{s.v.} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \;\geq 1 \quad \forall i = 1, ..., n \end{cases} \tag{5.6}$$

Mostriamo che le formulazioni (5.5) e (5.6) sono equivalenti. Si ha infatti

$$\begin{cases} y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \;\geq 0 \quad \forall i = 1, ..., n \\ \min_i y_i(\mathbf{w}\cdot\mathbf{x}_i + b) = 1 \end{cases} \quad \Longrightarrow \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \;\geq 1 \quad \forall i = 1, ..., n$$

Resta da verificare che nella formulazione (5.6) l'iperpiano trovato sia effettivamente canonico, cioè la soluzione sia tale che

$$\min_i y_i(\mathbf{w}\cdot\mathbf{x}_i + b) = 1$$

Supponiamo per assurdo che valga

$$\min_i y_i(\hat{\mathbf{w}}\cdot\mathbf{x}_i + \hat{b}) = \gamma > 1$$

Consideriamo allora i valori $\mathbf{w}_\gamma$ e b_γ definiti come

$$\mathbf{w}_\gamma = \frac{\hat{\mathbf{w}}}{\gamma}\,, \qquad\qquad b_\gamma = \frac{\hat{b}}{\gamma}$$

Chiaramente, essi verificano sia i vincoli di (5.5) che di (5.6). Inoltre, vale

$$\|\mathbf{w}_\gamma\|^2 = \Big(\frac{\|\hat{\mathbf{w}}\|}{\gamma}\Big)^2 \;<\; \hat{\mathbf{w}}$$

il che è assurdo. Perciò, le due formulazioni sono in effetti equivalenti.

2.3 Risoluzione tramite moltiplicatori di Lagrange

Per risolvere il problema (5.6), è possibile applicare il metodo dei moltiplicatori di Lagrange. Se introduciamo i moltiplicatori reali non negativi $\alpha_1, ..., \alpha_n$, indicati per comodità di notazione con il vettore $\boldsymbol{\alpha} \in \mathbb{R}^n$, la lagrangiana si scrive:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) \;=\; \frac{1}{2}\|\mathbf{w}\|^2 \;-\; \sum_{i=1}^{n} \alpha_i \big[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \big] \tag{5.7}$$

Il problema di ottimizzazione diventa perciò un problema di punto sella:

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha}} \; \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) \tag{5.8}$$

Si può verificare che i vincoli imposti in (5.6) definiscono un insieme convesso. Poiché la funzione obiettivo $\frac{1}{2}\|\mathbf{w}\|^2$ è quadratica, il problema è convesso ed i valori $(\hat{\mathbf{w}}, \hat{b}, \hat{\boldsymbol{\alpha}})$ sono soluzione di (5.8) se e solo se soddisfano le condizioni di Karush-Kuhn-Tucker (KKT). Scriviamo perciò le condizioni KKT:

- stazionarietà della lagrangiana rispetto a $\mathbf{w}$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}}(\hat{\mathbf{w}}, \hat{b}, \hat{\boldsymbol{\alpha}}) = 0 \quad \Longleftrightarrow \quad \hat{\mathbf{w}} \;=\; \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i \tag{5.9}$$

- stazionarietà della lagrangiana rispetto a b

$$\frac{\partial \mathcal{L}}{\partial b}(\hat{\mathbf{w}}, \hat{b}, \hat{\boldsymbol{\alpha}}) = 0 \quad \Longleftrightarrow \quad \sum_{i=1}^{n} \hat{\alpha}_i y_i \;=\; 0 \tag{5.10}$$

- condizioni di complementarietà:

$$\hat{\alpha}_i \big[\, y_i(\hat{\mathbf{w}} \cdot \mathbf{x}_i, +, b) \;-1 \,\big] \;=\; 0 \quad \forall i = 1, ..., n \tag{5.11}$$

Ora, possiamo eliminare $\mathbf{w}$ e b nella definizione della lagrangiana (5.7), utilizzando la (5.9) e la (5.10). Si ottiene così la lagrangiana duale

$$\mathcal{L}_D(\boldsymbol{\alpha}) \;=\; \frac{1}{2} \big(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \big) \cdot \big(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j \big) - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \cdot \big(\sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j \big) + \sum_{i=1}^{n} \alpha_i \;=$$

$$= \sum_{i=1}^{n} \alpha_i \;-\; \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{5.12}$$

Possiamo ora scrivere il problema (5.8), che ricordiamo essere equivalente al problema di partenza (5.6), utilizzando la lagrangiana duale. Otteniamo un

problema di ottimizzazione quadratica nelle variabili duali α_i. I valori ottimi $\hat{\alpha}_1, ..., \hat{\alpha}_n$ sono allora soluzioni di:

$$\begin{cases} \hat{\boldsymbol{\alpha}} \;=\; \arg\max_{\boldsymbol{\alpha}} \; \mathcal{L}_D(\boldsymbol{\alpha}) \;=\; -\,\frac{1}{2}\,\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ \\ \text{s.v.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{5.13}$$

Consideriamo ora le condizioni di complementarietà (5.11). Affinché siano verificate, deve valere

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 1 \implies \alpha_i = 0$$

Ciò significa che α_i è diversa da zero solamente se $\mathbf{x}_i$ è un vettore di supporto. In altri termini, solo i vettori di supporto intervengono nella definizione dell'iperpiano. Essendo questi ultimi in numero molto inferiore rispetto alla quantità n di vettori del training set, il calcolo numerico della soluzione di (5.13) comporta una quantità ridotta di operazioni e rende il metodo attrattivo da un punto di vista computazionale.

Infine, una volta risolto il problema di ottimizzazione duale (5.13), si può trovare l'iperpiano di separazione ottimo ricavando $\hat{\mathbf{w}}$ e $\hat{b}$, il primo direttamente dalla (5.9), il secondo con un'espressione leggermente più complessa che deriva dalla (5.11). Se indichiamo con $\mathbf{x}_s$, $s = 1, ..., N_{sv}$, i vettori di supporto del training set, vale allora:

$$\hat{b} = \frac{1}{N_{sv}} \Big[\sum_{s=1}^{N_{sv}} (\frac{1}{y_s} - \hat{\mathbf{w}} \cdot \mathbf{x}_s) \Big]$$

3 Dati non separabili

Fino ad ora, abbiamo trattato solo il caso più semplice in cui i dati del training set sono perfettamente separabili da un iperpiano. Questa situazione si verifica però raramente. Come possiamo tenere conto dei casi più complessi? Se i dati non sono separabili, rispetto al problema di ottimizzazione (5.6) non sono più soddisfacibili i vincoli

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \qquad \forall i = 1, ..., n \tag{5.14}$$

È necessario permettere che qualche dato del training set sia misclassificato. Per farlo, introduciamo le variabili reali positive $\xi_1, ..., \xi_n$, sostituendo la (5.15) con la seguente condizione:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \qquad \forall i = 1, ..., n \tag{5.15}$$

Ovviamente, più le ξ_i assumono un valore elevato, più il dato $\mathbf{x}_i$ rischia di essere misclassificato. Si deve tener conto di ciò aggiungendo un termine di penalità nella funzione obiettivo. Cerchiamo allora di massimizzare la nuova funzione costo $\mathcal{J}(\mathbf{w}, \boldsymbol{\xi})$ definita a meno di due parametri reali positivi C e μ nel modo seguente:

$$\mathcal{J}(\mathbf{w}, \xi) \;=\; \frac{1}{2}\,\|\mathbf{w}\|^2 + C\big(\sum_{i=1}^{n} \xi_i\big)^{\mu} \tag{5.16}$$

Il problema di ottimizzazione per la ricerca dell'iperpiano di separazione ottimale diventa allora

$$\begin{cases} \min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{J}(\mathbf{w}, \boldsymbol{\xi}) \\ \\ \text{s.v.} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \end{cases} \tag{5.17}$$

Solitamente, il parametro μ è posto uguale a 1. Con questa scelta, risolvendo il problema (5.17) attraverso il metodo dei moltiplicatori di Lagrange come in precedenza, la dipendenza dalle ξ_i non sarà più esplicita. Vedremo meglio nel seguito questo dettaglio. Il coefficiente C permette invece di variare il peso dei dati del training set misclassificati nella definizione dell'iperpiano. Più C è grande, più i dati misclassificati influenzano la definizione dell'iperpiano.

3.1 Risoluzione tramite moltiplicatori di Lagrange

Come nel caso di dati separabili, usiamo il metodo dei moltiplicatori di Lagrange per risolvere il problema di ottimizzazione. Introducendo i vettori di variabili duali $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_n]^t$ e $\boldsymbol{\beta} = [\beta_1, ..., \beta_n]^t$, la lagrangiana per $\mu = 1$ si scrive

$$\begin{aligned} \mathbf{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \;=\; & \frac{1}{2}\|\mathbf{w}\|^2 \;+\; C\big(\sum_{i=1}^{n} \xi_i\big) \;+ \\ & - \sum_{i=1}^{n} \alpha_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i] \;-\; \sum_{i=1}^{n} \beta_i \xi_i \end{aligned} \tag{5.18}$$

Ancora una volta, le condizioni KKT sono necessarie e sufficienti per determinare il punto sella $(\hat{\mathbf{w}}, \hat{b}, \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}})$. Abbiamo allora:

- stazionarietà rispetto a $\mathbf{w}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}}(\hat{\mathbf{w}}, \hat{b}, \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) = 0 \Longleftrightarrow \hat{\mathbf{w}} = \sum_{i=1}^{n} \hat{\alpha}_i y_i \mathbf{x}_i \tag{5.19}$$

- stazionarietà rispetto a b:

$$\frac{\partial \mathcal{L}}{\partial b}(\hat{\mathbf{w}}, \hat{b}, \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) = 0 \Longleftrightarrow \sum_{i=1}^{n} \hat{\alpha}_i y_i = 0 \tag{5.20}$$

- stazionarietà rispetto alle ξ_i:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi_i}}(\hat{\mathbf{w}}, \hat{b}, \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) = 0 \Longleftrightarrow \hat{\alpha_i} + \hat{\beta}_i = C \tag{5.21}$$

- condizioni di complementarietà:

$$\hat{\alpha_i}[y_i(\hat{\mathbf{w}} \cdot \mathbf{x}_i + \hat{b}) - 1 + \hat{\xi}_i] = 0 \qquad \forall i = 1, ..., n \tag{5.22}$$

Riscriviamo ora la (5.18) sostituendo $\mathbf{w}$ con la (5.19):

$$\mathcal{L}(b, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\xi}) \;=\; \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \;-\; \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \;+$$

$$+ C \sum_{i=1}^{n} \xi_i \;-\; \sum_{i=1}^{n} \alpha_i y_i b \;+\; \sum_{i=1}^{n} \alpha_i \;-\; \sum_{i=1}^{n} \alpha_i \xi_i - \sum_{i=1}^{n} \beta_i \xi_i$$

Utilizzando poi le condizioni (5.20) e (5.21), otteniamo infine l'espressione per la lagrangiana duale, che non presenta dipendenza dalle β_i e ξ_i:

$$\mathcal{L}_D(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \;+\; \sum_{i=1}^{n} \alpha_i \tag{5.23}$$

Il problema duale per trovare l'ottimo $\hat{\boldsymbol{\alpha}}$ diventa allora

$$\begin{cases} \hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \mathcal{L}_D(\boldsymbol{\alpha}) \\ \\ \text{s.v.} \quad 0 \leq \alpha_i \leq C \quad \forall i = 1, ..., n \\ \qquad \sum_i \alpha_i y_i = 0 \end{cases} \tag{5.24}$$

Dai vincoli espressi nella (5.17) e dalle condizioni di complementarietà (5.22), ricordando che le ξ_i assumono valori non negativi e devono essere minimizzate, si ha che gli unici α_i non nulli corrispondono ai vettori del training set di supporto o misclassificati. Ancora una volta, il calcolo dell'iperpiano ottimale prevede un numero limitato di calcoli, rispetto alla dimensione del training set.

4 SVM non lineari

È possibile generalizzare ulteriormente il metodo SVM, introducendo ipersuperfici di classificazione non lineari. La costruzione di una superficie non lineare avviene, per i metodi SVM, attraverso un procedimento particolare. Dato il training set composto da vettori $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^p$, si cerca una mappa $\mathbf{\Phi} : \mathbb{R}^p \to F$ che trasformi gli $\mathbf{x}_i$ nei vettori $\mathbf{z}_i \in F$, con $\mathbf{z}_i = \mathbf{\Phi}(\mathbf{x}_i)$. Si procede poi con la ricerca di un iperpiano di separazione per i nuovi dati $\mathbf{z}_1, ..., \mathbf{z}_n$ nello spazio F. L'idea è di scegliere lo spazio F più grande rispetto a quello di partenza $\mathbb{R}^p$, in modo che l'iperpiano trovato in F separi meglio i dati. Geometricamente, se $\mathbf{\Phi}$ è non lineare, un iperpiano di F corrisponde ad una superficie non lineare in $\mathbb{R}^p$. Un classificatore SVM non lineare per dati in $\mathbb{R}^p$ corrisponde dunque ad un classificatore SVM lineare in uno spazio di dimensione maggiore F.

4.1 Classificazione di un nuovo dato

Supponiamo per il momento di conoscere la mappa $\mathbf{\Phi}$. Costruiamo allora l'iperpiano di classificazione ottimo di equazione $\hat{\mathbf{w}} \cdot \mathbf{z} + \hat{b} = 0$ utilizzando i dati trasformati $\mathbf{z}_i$. Per farlo, adottiamo lo stesso procedimento visto in precedenza, ed indichiamo ancora una volta $\hat{\alpha}_1, ..., \hat{\alpha}_n$ i valori ottimi per le variabili duali.
Immaginiamo poi di dover classificare un nuovo dato osservato $\mathbf{x}_{new}$, la cui rappresentazione nel nuovo spazio F è indicata con $\mathbf{z}_{new} = \mathbf{\Phi}(\mathbf{x}_{new})$. La classificazione consiste nell'assegnare all'*etichetta* y_{new} un valore scelto tra -1 e $+1$. Abbiamo allora

$$y_{new} = \text{sign}(\hat{\mathbf{w}} \cdot \mathbf{z}_{new} + \hat{b}) \tag{5.25}$$

Utilizzando le variabili duali ottimali $\hat{\alpha}_i$, la (5.25) può essere scritta come:

$$y_{new} = \text{sign}\big(\sum_{i=1}^{n} \hat{\alpha}_i \mathbf{z}_i \cdot \mathbf{z}_{new} + \hat{b}\big) = \sum_{s=1}^{N_{sv}} \big(\sum_{i=1}^{n} \hat{\alpha}_s \mathbf{z}_s \cdot \mathbf{z}_{new} + \hat{b}\big) \tag{5.26}$$

dove gli $\mathbf{z}_s$ sono i vettori di supporto espressi nello spazio F. Poiché $\hat{b}$ è calcolato inizialmente una volta per tutte, la classificazione di un nuovo dato richiede solamente le operazioni di prodotto scalare del nuovo vettore trasformato $\mathbf{z}_{new}$ con i vettori di supporto trasformati $\mathbf{z}_s$. Non è perciò necessario calcolare esplicitamente il vettore $\hat{\mathbf{w}}$ per definire l'iperpiano di separazione. Sarà inoltre necessario scegliere la mappa $\mathbf{\Phi}$ in modo tale che

il calcolo del prodotto scalare

$$\mathbf{z}_s \cdot \mathbf{z}_{new} = \mathbf{\Phi}(\mathbf{x}_s) \cdot \mathbf{\Phi}(\mathbf{x}_{new})$$

sia vantaggioso da un punto di vista computazionale.

4.2 Funzioni kernel

Il calcolo dei prodotti scalari in F non è richiesto solo per la classificazione di un dato nuovo. La lagrangiana duale definita in (5.23) diventa infatti

$$\mathcal{L}_D(\boldsymbol{\alpha}) \;=\; -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{z}_i \cdot \mathbf{z}_j \;+\; \sum_{i=1}^{n} \alpha_i \;=$$

$$=\; -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j) \;+\; \sum_{i=1}^{n} \alpha_i$$

Ora, i vincoli associati al problema duale impongono solo condizioni sulle α_i. Perciò, sia per il problema di ottimizzazione che per la classificazione di un nuovo dato, non è necessario conoscere esplicitamente i vettori $\mathbf{z}_i$, poiché essi compaiono sempre come membri di un prodotto scalare.

Formalmente, possiamo esprimere il prodotto scalare tra due vettori $\mathbf{z}_i = \mathbf{\Phi}(\mathbf{x}_i) \in F$, $\mathbf{z}_j = \mathbf{\Phi}(\mathbf{x}_j) \in F$ con una funzione $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, detta *funzione kernel*:

$$\mathbf{z}_i \cdot \mathbf{z}_j = \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$$

Grazie all'introduzione della funzione kernel K, non è più necessario trattare i dati trasformati $\mathbf{z}_i$, né conoscere esplicitamente la mappa $\mathbf{\Phi}$. Possiamo allora cambiare completamente il punto di vista. Invece di introdurre uno spazio F in cui mappare i dati del training set e successivamente definire un prodotto scalare in F esprimibile attraverso una funzione kernel $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, possiamo scegliere dapprima una funzione kernel K, e definire eventualmente in un secondo momento lo spazio F (si noti che non è più necessario conoscere esplicitamente F per la definizione del classificatore).

Ma quali funzioni kernel corrispondono effettivamente ad un prodotto scalare in uno spazio F? E come definire F in funzione di K? Innanzi tutto, diamo una risposta alla prima domanda attraverso il seguente risultato.

Proposizione Sia $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ simmetrica. K rappresenta un prodotto scalare in uno spazio F di dimensione maggiore di $\mathbb{R}^p$ (eventualmente dimensione infinita) se per ogni funzione $g : \mathbb{R}^p \to \mathbb{R}$ di classe L^2 vale:

$$\int_{\mathbb{R}^p \times \mathbb{R}^p} K(\mathbf{x}_1, \mathbf{x}_2)\, g(\mathbf{x}_1)\, g(\mathbf{x}_2)\, d\mathbf{x}_1\, d\mathbf{x}_2 \;\; > \;\; 0 \tag{5.27}$$

La proposizione precedente impone una condizione che la funzione kernel deve soddisfare, per poter essere impiegata in un metodo SVM. Se K soddisfa la (5.27), possiamo definire la mappa $\mathbf{\Phi} : \mathbb{R}^p \to F$. Indichiamo con $\{(\varphi_i, \lambda_i)\}$ la successione di coppie di autovettori e autovalori definiti dal problema

$$K(\mathbf{x}_1, \mathbf{x}_2)\varphi_i(\mathbf{x}_1)d\mathbf{x}_1 \;\; = \;\; \lambda_i \varphi_i(\mathbf{x}_2) \tag{5.28}$$

K è allora scomponibile nel modo seguente:

$$K(\mathbf{x}_1, \mathbf{x}_2) \;\; = \;\; \sum_i \lambda_i \varphi_i(\mathbf{x}_1)\varphi_i(\mathbf{x}_2) \tag{5.29}$$

Possiamo allora rappresentare il vettore $\mathbf{z} = \mathbf{\Phi}(\mathbf{x})$ attraverso la successione di componenti (eventualmente infinita)

$$[\, \sqrt{\lambda_1}\varphi_1(\mathbf{x})\,,\; ...,\; \sqrt{\lambda_i}\varphi_i(\mathbf{x}_i)\,,\; ...]$$

Questa rappresentazione permette allora di far coincidere il prodotto scalare tra due vettori $\mathbf{z}_1 = \mathbf{\Phi}(\mathbf{x}_1)$ e $\mathbf{z}_2 = \mathbf{\Phi}(\mathbf{x}_2)$ con l'espressione (5.29):

$$\mathbf{z}_1 \cdot \mathbf{z}_2 \;\; = \;\; \sum_i z_{1i}\, z_{2i} \;\; = \;\; \sum_i \lambda_i \varphi_i(\mathbf{x}_1)\varphi_i(\mathbf{x}_2)$$

Per riassumere, data la funzione kernel $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, il problema di ottimizzazione per la costruzione del classificatore si scrive:

$$\begin{cases} \hat{\boldsymbol{\alpha}} = \arg\max_{\boldsymbol{\alpha}} \sum_{i=1}^n \alpha_i - \dfrac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \\ \text{s.v.} \quad 0 \le \alpha_i \le C \quad \forall i = 1, ..., n \\ \qquad\quad \sum_{i=1}^n \alpha_i y_i = 0 \end{cases} \tag{5.30}$$

e la funzione di classificazione per $\mathbf{x}_{new}$ assegna ad y_{new} il valore

$$y_{new} = \text{sign}(\sum_{i=1}^{n} \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}_{new}) + \hat{b}) \tag{5.31}$$

Il valore di bias $\hat{b}$ spesso non compare nella formula, poiché è possibile eliminarlo scegliendo opportunamente la funzione kernel.

Vediamo infine alcuni esempi di funzioni kernel. Le più usate sono:

- kernel polinomiale di grado d:

$$K(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 \cdot \mathbf{x}_2 + 1)^d$$

- kernel cosiddetto *RBF* (con $\sigma \in \mathbb{R}$):

$$K(\mathbf{x}_1, \mathbf{x}_2) = \exp\{ -\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\sigma^2} \}$$

- kernel gaussiano:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \exp\{ -\frac{1}{2}(\mathbf{x}_1 - \mathbf{x}_2)^t \Sigma^{-1} (\mathbf{x}_1 - \mathbf{x}_2) \}$$

- kernel che genera l'algorimo *multilayer perceptron*[1]:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\mathbf{x}_1 \cdot \mathbf{x}_2 + b)$$

L'ultima espressione di K origina un classificatore equivalente allo stesso che si otterrebbe tramite un altro noto algoritmo di classificazione supervisionata binaria, il *multilayer perceptron*. Non si ritiene utile descrivere anche quest'ultimo, basti sapere che può essere espresso come caso particolare di metodo SVM.

5 Training set sbilanciato

Nelle applicazioni di text mining relative all'estrazione di informazioni, si hanno in genere dei training set popolati per la maggior parte da dati negativi. Se consideriamo i token come unità statistiche da classificare, è facile pensare che la maggior parte dei token di un testo non faccia parte di una sequenza che descrive un'entità d'interesse. Questo sbilanciamento nell'informazione data dal training set potrebbe quindi influire negativamente sulle performance di un classificatore.

[1] b può assumere solo determinati valori ammissibili in $\mathbb{R}$. Si rimanda a [13] per ulteriori dettagli

Intuitivamente, essendo i dati negativi meglio rappresentati nel training set, se ne conosce meglio il posizionamento in $\mathbb{R}^p$ rispetto ai dati positivi. Di conseguenza, è ragionevole costruire la iper-superficie di separazione in modo che sia più *vicina* ai dati negativi. Poniamoci nel caso lineare. Cosa significa *avvicinare* l'iperpiano di separazione ai dati negativi? Nella definizione dell'iperpiano canonico, avevamo imposto che i dati $\mathbf{x}_s$ più vicini all'iperpiano, detti vettori di supporto, fossero tali per cui $|\mathbf{x}_s \cdot \mathbf{w} + b| = 1$. Per avvicinare l'iperpiano ai dati negativi, possiamo introdurre un parametro $\tau \in (0, 1)$ ed imporre una condizione diversa sui vettori di supporto:

$$\begin{cases} \mathbf{x}_s \cdot \mathbf{w} + b = 1 & \text{se } y_s = +1 \\ \mathbf{x}_s \cdot \mathbf{w} + b = -\tau & \text{se } y_s = -1 \end{cases}$$

Nel caso di dati non separabili linearmente, la modifica è la stessa. Il problema di ottimizzazione (5.17) per determinare l'iperpiano di classificazione diventa allora:

$$\begin{cases} (\hat{\mathbf{w}}, \hat{b}, \hat{\boldsymbol{\xi}}) = \arg\min_{(\mathbf{w},b,\boldsymbol{\xi})} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\big(\sum_{i=1}^n \xi_i\big)^\mu \\ \\ \text{s.v.} \quad \mathbf{w} \cdot \mathbf{x}_i + \xi_i + b \geq 1 \quad \text{se } y_i = +1 \\ \qquad\ \ \mathbf{w} \cdot \mathbf{x}_i - \xi_i + b \leq -\tau \quad \text{se } y_i = -1 \end{cases} \tag{5.32}$$

6 Classificazione non binaria

Resta ora da mostrare come vengano utilizzate le SVM, definite per problemi di classificazione binaria, nel caso vi siano più di due classi. Supponiamo allora che i dati debbano essere classificati nei gruppi A, B e C. Le strategie usate sono fondamentalmente due: *one against others* oppure *one against another*.

One against others

Nel primo caso, viene costruito un classificatore binario indipendente per ogni gruppo. Nell'esempio specifico, si avranno perciò tre classificatori, ognuno dei quali suddividerà il training set nel modo seguente:

	dati positivi	**dati negativi**
classificatore per A	A	B, C
classificatore per B	B	A, C
classificatore per C	C	A, B

La nuova osservazione $\mathbf{x}_{new}$ sarà allora classificata nel gruppo il cui classificatore ha dato il più ampio margine (cioè, il classificatore per cui il valore $\mathbf{w} \cdot \mathbf{x}_{new} + b$ è massimo).

Questo metodo è adottato nel caso dell'estrazione di informazioni. Supponiamo di essere interessati ad un processo di *named entity recognition*. Possiamo immaginare che il gruppo A dell'esempio precedente corrisponda ad un'entità di tipo A, il gruppo B ad un'entità di tipo B, il gruppo C ai token irrilevanti. Costruiremo allora un classificatore per A ed un classificatore per B. L'ultimo classificatore riportato nella tabella precedente diventa così inutile, poiché nel gruppo C rientrano tutti i token non classificati A né B.

One against another

In questo caso, si costruisce un classificatore binario per ogni coppia di gruppi. Rispetto all'esempio precedente, avremo i classificatori seguenti:

	gruppo 1	**gruppo 2**
classificatore A vs. B	A	B
classificatore per A vs. C	A	C
classificatore per B vs. C	B	C

Anche in questo caso, per scegliere il gruppo in cui classificare un dato nuovo $\mathbf{x}_{new}$, si considerano i risultati dei classificatori binari.

Capitolo 6

Applicazione: lettere di dimissione ospedaliera

1 Descrizione del problema

Come accennato nell'introduzione, lo studio di caso affrontato nell'ambito di questa tesi riguarda l'estrazione di informazioni da lettere di dimissione ospedaliera, relative a pazienti ricoverati per infarto miocardico acuto (IMA). Le lettere di dimissione ospedaliera sono conservate in appositi database testuali, per permettere al paziente di avere un facile accesso alla propria documentazione clinica pregressa, ma anche allo scopo di raccogliere informazioni sulla qualità del servizio sanitario. A tali database si affiancano i registri amministrativi, che riportano i dati relativi ai ricoveri in forma più schematica.

Lo scopo finale del presente lavoro è quello di fornire uno strumento in grado di comprendere la diagnosi descritta in ciascuna lettera, per avere un riscontro con i dati riportati nei database amministrativi. Questi ultimi registrano le diagnosi dei pazienti ricoverati attraverso un sistema di codifica che corrisponde agli standard ICD-9-CM ed ICD-10-CM[1] (dove ICD sta per *International Classification of Diseases*). Tali standard sono definiti dall'Organizzazione Mondiale della Sanità (OMS). D'altro lato, la formulazione di una diagnosi avviene seguendo regole precise. Ad esempio, un medico può diagnosticare un caso di IMA solamente se il paziente riporta sintomi particolari e se le analisi cliniche mostrano determinati risultati. I segni che un

[1] per maggiori informazioni si rimanda alla pagina web dell'Organizzazione Mondiale della Sanità dedicata all'ICD `http://www.who.int/classifications/icd/en/`

medico deve riscontrare per emettere una diagnosi sono chiamati *elementi diagnostici*.

Nel caso dell'IMA, gli elementi diagnostici sono tre: dolore toracico, alterazione degli enzimi presenti nel sangue e riscontri dell'esame elettrocardiografico (ECG). Tra i riscontri dell'ECG, ve ne sono alcuni fondamentali per formulare la diagnosi (li indicheremo come *primari*) ed altri solamente di supporto (*secondari*). Inoltre, vi sono alcuni sintomi che accompagnano il dolore toracico e che rafforzano la diagnosi di infarto. Essi saranno indicati come *sintomi secondari*.

Si è quindi interessati a verificare che in ogni lettera di dimissione relativa ad un caso di IMA, siano effettivamente presenti tutti gli elementi diagnostici necessari, accompagnati eventualmente dai riscontri secondari. Per farlo, vogliamo realizzare un sistema di text mining in grado di riconoscere le parti di testo relative a tali elementi. Il processo è sostanzialmente analogo a quello di *named entity recognition* descritto nel Capitolo 4.

Indichiamo nel seguito i cinque riscontri diagnostici cui siamo interessati, fornendo una veloce spiegazione senza entrare nei dettagli medici. D'ora in avanti per semplicità, con un abuso di notazione, chiameremo indistintamente elementi diagnostici tutti e cinque i riscontri.

1. **Dolore toracico:** è il sintomo principale che indica un possibile infarto. Sotto questa categoria faremo rientrare anche i riscontri di dolori agli arti superiori e parti del corpo limitrofe.
2. **Sintomi secondari:** vi sono molti sintomi secondari che, associati al dolore toracico, aumentano notevolmente le probabilità che il paziente sia effettivamente affetto da infarto. Citiamo ad esempio nausea, palpitazioni e pesantezza degli arti.
3. **Enzimi alterati:** l'analisi ematochimica può fornire alcuni elementi che indicano senza ombra di dubbio un infarto. Espressioni come *movimento enzimatico* possono descrivere situazioni problematiche, ma in generale le lettere di dimissione riportano i risultati numerici delle analisi. Elenchiamo di seguito gli enzimi di interesse con i relativi valori di soglia, oltre i quali si riconosce un elemento diagnostico tipico dell'infarto[2].

[2]In realtà, i valori critici dipendono da variabili quali sesso e peso del paziente. Non disponendo di tali dati abbiamo scelto, su indicazione del Dott. Pietro Barbieri, valori di soglia oltre i quali la probabilità di infarto sia generalmente alta.

Enzima	**Valori critici**
CPK	> 200
CPK-MB	> 25
Troponina	$> 0,06$

4. **Riscontri elettrocardiografici primari**: i risultati dell'ECG sono determinanti per una diagnosi di infarto. Vi sono alcuni elementi che indicano con poco margine di dubbio l'infarto, altri che rafforzano la convinzione del medico a riguardo. I primi sono indicati come riscontri primari, e sono descritti nel testo da espressioni quali *sovraslivellamento*, *sottoslivellamento* ed *onde T negative*. A volte i risultati dell' ECG possono essere descritti addirittura tramite espressioni inequivocabili come *segni di IMA inferolaterale*.

5. **Riscontri elettrocardiografici secondari:** gli elementi dell'ECG che servono solo a rafforzare un'ipotesi di infarto sono indicati come secondari. Tra questi rientrano ad esempio *ischemie* e *lesioni*.

Chiaramente, i diversi tipi di *named entity* da riconoscere nel testo sono i cinque elementi diagnostici sopra indicati. Nel seguito del capitolo vedremo dunque come è stato possibile costruire un sistema di text mining in grado di svolgere tale compito.

2 Formato dei testi

Innanzi tutto, va precisato che ogni lettera di dimissione ospedaliera è composta da diverse parti, che devono seguire determinati standard per quanto riguarda il contenuto informativo. Non vi è invece uno standard sul formato specifico della lettera comune a tutti gli ospedali. Nel nostro caso, le diverse parti sono riportate in campi diversi dello stesso documento. Tra questi, i campi *anamnesi* ed *esami* sono di particolare importanza, poiché contengono le informazioni sulla diagnosi cui siamo interessati. Riportiamo di seguito una descrizione più precisa di questi due campi.

1. **Anamnesi:** riporta la situazione clinica del paziente, eventuali problemi e trattamenti ricevuti nel passato, cause e modalità del ricovero. Può essere redatto al momento del ricovero o successivamente, sulla base di un dialogo del medico con il paziente (sempre che il paziente sia in stato di coscienza), attraverso cui il medico viene a conoscenza dei problemi del paziente e formula una prima ipotesi di diagnosi.

2. **Esami:** descrive brevemente i risultati degli esami precedenti alla dimissione del paziente. Questi possono comprendere elettrocardiogramma, esami RX del torace, coronarografie ed esami ematochimici. Nel nostro caso, siamo interessati solo ai dati dell'ECG e degli esami ematochimici. A volte, questo campo può risultare poco influente, poiché può semplicemente richiamare i referti degli esami contenuti nella cartella clinica del paziente con formule del tipo *vedi allegato*.

Le lettere di dimissione sono registrate in un database MS Access (anche in questo caso, non esiste uno standard sul formato dell'archiviazione). Per avere accesso ai soli campi *anamnesi* ed *esami*, l'esportazione da Access ha generato due distinti file di testo (estensione `.txt`) per ogni lettera di dimissione, ognuno relativo ad un campo. Dato il formato di solo testo, non è possibile sfruttare eventuali informazioni provenienti dalla formattazione (ad esempio eventuali titoli che permettano una distinzione tra sezioni).
È importante tenere presente che i testi sono in genere molto brevi, composti da frasi sintetiche e caratterizzati da un linguaggio settoriale molto specifico. Sebbene la terminologia cardiologica sia circoscritta, il linguaggio medico è molto vario, soprattutto per quanto riguarda la sintassi, ma anche per l'enorme varietà delle possibili abbreviazioni usate per indicare gli stessi termini. Si avrà quindi molta variabilità a seconda del medico che redige i documenti, anche nella lunghezza stessa dei testi, che può andare da 12-15 parole a circa 100.

Esempio di lettera di dimissione

I dati su cui lavoriamo sono sensibili, quindi riservati. Non è perciò possibile presentare un esempio effettivo tratto dalla collezione di documenti. Vista la brevità dei testi, riteniamo comunque interessante riportare un *fac-simile* redatto ad hoc, per dare un'idea più precisa del problema trattato. Si noti in questo esempio come il linguaggio sia ostico, praticamente incomprensibile ai non addetti ai lavori, in particolare nel secondo documento, relativo agli esami clinici.

Anamnesi:

Fumo. Storia di ipertensione arteriosa. Trattamento pregresso con amiodarone. Precedenti episodi di dolore toracico. Giunge in PS per dolore toracico gravativo. Modificazioni ecg (segni come da ischemia-lesione subendocardica diffusi), in seguito comparsa di onde T negative e dispersione enzimatica.

Esami:

Esami ematochimici: CPK max 916, MB 110, GOT 112, GPT 23, colesterolo tot. 213. Coronarografia: leggera coronaropatia monovasale (vedi allegato). ECG: ritmo sinusale, onde T negative antero-settali. Rx torace: nessuna lesione pleuroparenchimale.

Il lavoro sui testi si annuncia quindi particolarmente complesso, soprattutto dal punto di vista linguistico. Sarà infatti necessario un lavoro approfondito non solo per comprendere il linguaggio medico specifico, ma per creare degli strumenti che *normalizzino* i termini impiegati. Ci occuperemo di questi aspetti nelle prossime sezioni.

3 Preprocessing

3.1 Tokenization

La prima fase di preprocessing consiste nella suddivisione della successione dei caratteri che compongono il testo in singoli elementi, generalmente parole e punteggiatura, detti *token*. Abbiamo già descritto l'operazione e le difficoltà ad essa legate nel Capitolo 1. Nel caso delle lettere di dimissione ospedaliera, non vi sono particolari difficoltà ulteriori. Il processo è dunque affidato ad uno strumento incluso in GATE, l'*UNICODE Tokenizer*, che permette di ottenere risultati discreti, fatta eccezione per i valori numerici.

3.2 Valori numerici

L'*UNICODE Tokenizer* di GATE riscontra dei problemi, qualora siano riportati nel testo numeri con valori decimali. Ad esempio, la stringa 0.12 sarà processata dal tokenizer individuando tre elementi distinti:

Token	**Tipo**
0	numero
.	punteggiatura
12	numero

Ora, i valori numerici diventano fondamentali quando associati agli enzimi. Analizzando i documenti che riassumono i risultati degli esami ematochimici, vogliamo poter distinguere tra valori al di sopra o al di sotto di una soglia critica. Diventa perciò necessario intervenire nella fase di tokenization, permettendo il riconoscimento di numeri che contengono cifre decimali. Si

è scelto allora di scrivere un'apposita decision rule, per correggere i risultati del tokenizer di GATE. La regola assegna inoltre il numero ad uno fra quattro gruppi:

Gruppo	A	B	C	D
Valori	$(-\infty, 0.06]$	$(0.06, 25]$	$(25, 200]$	$(200, \infty)$

I gruppi sono definiti in modo da agevolare il riconoscimento di valori critici negli esami ematochimici.

3.3 Stemming e POS tagging

Di fondamentale importanza è la fase di *stemming*. Ricordiamo che lo *stemming* prevede una *normalizzazione* delle parole, in modo da ricondurre allo stesso termine, o meglio *lemma*, le diverse forme date da coniugazioni e declinazioni. Nel nostro caso, l'operazione di *stemming* è più complessa. Medici diversi utilizzano infatti diversi tipi di abbreviazioni o formule per indicare le stesse entità. Ad esempio, per indicare il pronto soccorso si possono usare le scritture *pronto soccorso*, *p-s*, *p.s.* e *ps*. E non si può escludere che la lettura di nuovi documenti di anamnesi, scritti da nuovi medici, presenti ulteriori forme fin qui non individuate. È quindi necessaria una normalizzazione approfondita di tali formule, per rendere più agevole un'analisi di text mining.

Non disponendo di strumenti di stemming universali per la lingua italiana (e qualora ve ne fossero, difficilmente sarebbero applicabili ad un linguaggio così specifico come quello medico cardiologico), sono stati redatti dei dizionari appositi[3], in grado di ricondurre le varie forme ad un unico lemma. Inoltre, i dizionari possono presentare informazioni supplementari, tra cui il ruolo grammaticale delle parole. La fase di stemming è allora realizzata in un unico momento con quella di POS-tagging, attraverso la redazione dei dizionari.
Per finire, i dizionari non consentono solamente la normalizzazione dei termini medici, ma sono suddivisi in base a tematiche precise. Ad esempio, i termini relativi alle onde riscontrate da un esame ECG sono registrati in un dizionario distinto da quello che contiene i nomi degli enzimi di interesse per un esame ematochimico. Ciò fornisce una conoscenza linguistica molto approfondita, che permette il riconoscimento degli elementi diagnostici anche grazie a regole deterministiche (senza ricorrere quindi a metodi statistici). Tuttavia, essendo gli strumenti linguistici strettamente legati al contesto in

[3] Il difficile lavoro di redazione dei dizionari è opera del Dott. Stefano Ballerio.

cui sono stati creati, saranno difficilmente applicabili in ambiti diversi. In altri termini, le conoscenze linguistiche ottenute esaminando il gergo utilizzato da tre o quattro cardiologi di un certo ospedale possono essere estremamente efficaci per analizzare nuovi testi redatti dagli stessi medici, ma possono rilevarsi imprecisi per l'analisi di lettere di dimissione redatte da altri medici in ospedali diversi.

Due altre componenti linguistiche di fondamentale importanza per il riconoscimento degli elementi diagnostici sono gli avverbi temporali e le negazioni. Facciamo un esempio. Supponiamo che un testo contenga le seguenti due frasi[4]:

1. *Si esclude la presenza di onde T negative.*
2. *Negli anni precedenti, episodi di dolore toracico.*

Nella frase 1, il sistema potrebbe riconoscere erroneamente un riscontro elettrocardiografico primario, quando invece la negazione precedente lo esclude. La frase 2, invece, presenta la descrizione di sintomi lontani nel tempo, non direttamente legati all'attuale ricovero. Vorremmo quindi che il sistema non riconoscesse, in questo caso, l'elemento diagnostico *dolore toracico*.
Sembra perciò utile riconoscere nel testo sia le espressioni temporali che indicano eventi passati, sia le espressioni usate per formulare una negazione. A questo scopo, sono stati redatti dei dizionari che riportano una grande varietà di espressioni.

La stesura dei dizionari sopra descritti, in particolare quelli strettamente legati al gergo medico, rientra negli scopi dello studio linguistico sulle risorse testuali ospedaliere. Vi è infatti un progetto a lungo termine che prevede la costruzione di una serie di risorse linguistiche, che servano di appoggio ad un'analisi automatica dei testi con metodi di text mining o di natural language processing.

Prima di concludere la sezione, riportiamo un brevissimo esempio estratto da un dizionario contenente termini relativi a sintomi secondari. Le scritture *gnr*, *nmb* e *POS* indicano, rispettivamente, genere, numero e *part of speech* (ruolo grammaticale). Il codice relativo sarà mostrato nell'Appendice A.

[4]Le frasi riportate sono molto semplicistiche e difficilmente possono essere ritrovate in forme simili. Rendono tuttavia l'idea delle difficoltà che si possono incontrare in caso di negazioni o di descrizioni di eventi pregressi.

termine	gnr	lemma	nmb	POS
palpitazione	f	palpitazione	s	N
palpitazioni	f	palpitazione	p	N
sudorazione	f	sudorazione	s	N
sudorazioni	f	sudorazione	p	N
algido	m	algido	s	A
algidi	m	algido	p	A
algida	f	algido	s	A
algide	f	algido	p	A

4 Decision rules

Prima di impiegare tecniche di machine learning, si è cercato di realizzare il processo di named entity recognition attraverso la definizione di regole deterministiche. Grazie ad un'attenta analisi linguistica delle lettere di dimissione, il Dott. Stefano Ballerio ha così definito delle decision rules in grado di riconoscere con buona precisione gli elementi diagnostici. Le regole si basano sull'identificazione nel testo di particolari sequenze di elementi appartenenti ai dizionari. In una prima fase si individuano i potenziali elementi diagnostici, in seguito vengono eliminati gli elementi che si riferiscono a eventi passati o che sono negati.

Sarà interessante provare ad utilizzare le decision rules come feature per la classificazione. Scegliendo opportunamente le altre feature osservabili, si potrebbe pensare di colmare le eventuali imperfezioni delle decision rules.

5 Classificazione supervisionata

Vediamo infine come è stato impostato il problema di classificazione supervisionata per il riconoscimento degli elementi diagnostici. Innanzi tutto, ricordiamo che l'unità statistica per la classificazione è il token. Per ogni token si costruisce un vettore di feature, rappresentato matematicamente da un vettore $\mathbf{x} \in [0,1]^p$, come spiegato nel Capitolo 4. Le componenti inizialmente binarie sono moltiplicate, sempre in relazione alle notazioni introdotte nel Capitolo 4 (pag. 54), per i coefficienti $\mu_i = \frac{1}{i}$ nel caso si riferiscano a feature osservate nella posizione relativa $\pm i$.

Prima di descrivere le feature impiegate per la costruzione dei vettori, è necessario un chiarimento sui gruppi in cui i token devono essere classificati. Come sappiamo, vogliamo riconoscere cinque tipi di elementi diagnostici. Ogni elemento diagnostico è riconosciuto nel testo come una particolare sequenza di token. Concentriamoci per il momento su un unico tipo, ad esempio *sintomi secondari* (le stesse considerazioni si applicano a tutti i tipi). Intuitivamente, si potrebbe pensare di classificare ogni token, indicandone l'appartenenza o meno ad una sequenza che descrive un sintomo secondario. Tuttavia, questa impostazione si rivela problematica. Come trattare i casi in cui una sequenza che descrive un sintomo secondario è riconosciuta solo parzialmente? Per aggirare il problema, è preferibile definire un classificatore che riconosca solamente il token iniziale ed il token finale di ogni sequenza. I gruppi per la classificazione saranno allora in totale 11: due per ogni tipo di elemento diagnostico (token iniziale e token finale) ed il gruppo dei token irrilevanti. Più precisamente, poiché saranno usati metodi SVM, costruiremo due classificatori binari per ogni tipo di elemento diagnostico.

5.1 Feature osservate

Per valutare l'impatto positivo degli strumenti linguistici messi a disposizione dal dott. Stefano Ballerio, è interessante effettuare diverse analisi, considerando in ognuna di esse feature diverse per la costruzione dei vettori. Inizieremo dunque con un'analisi più grezza, basata sulle stringhe di caratteri dei token e sui numeri, per poi includere man mano informazioni linguistiche più approfondite, date dai dizionari e dalle decision rules. Effettueremo perciò cinque diverse analisi, in ognuna delle quali ad ogni token corrisponderà un vettore di feature costruito diversamente.

I - Feature basate sui token: non si utilizzano i dizionari né le decision rules. Le feature considerate sono:

- il token stesso (la stringa di caratteri);
- i token presenti nelle posizioni relative $-8, -7, ..., -1, +1, ..., +8$;
- se il token è un numero, se ne considera come feature il gruppo (definito a pag. 84);
- i gruppi di eventuali numeri presenti nelle posizioni relative $+1$ e $+2$.

II - Feature basate sui dizionari: non si considerano più le stringe di caratteri che compongono i token, ma solo i token normalizzati e le informazioni supplementari date dai dizionari. Ricordiamo che la struttura dei dizionari permette la suddivisione delle voci in categorie. Abbiamo allora le seguenti feature:

- il token normalizzato (*lemma*);
- i lemmi presenti nelle posizioni relative $-3, ..., +3$;
- la *part of speech* del token;
- la *part of speech* dei token nelle posizioni relative $-3, ..., +3$;
- la categoria da cui proviene la voce del dizionario relativa al token;
- la categoria da cui provengono le voci del dizionario relative ai token in posizione $-3, ..., +3$;
- se il token è un numero, se ne considera come feature il gruppo;
- i gruppi di eventuali numeri presenti nelle posizioni relative $+1$ e $+2$.

III - Feature basate sia sui token che sui dizionari: si considerano tutte le feature elencate in precedenza.

IV - Feature basate sulle decision rules e i token: si utilizzano le stesse feature della prima analisi sui token (I), aggiungendo le indicazioni date dalle decision rules.

V - Feature basate sulle decision rules e sui dizionari: si utilizzano le feature della seconda analisi (II), aggiungendo le indicazioni date dalle decision rules.

5.2 Algoritmi impiegati

Come detto in precedenza, per la classificazione supervisionata adottiamo metodi SVM. In particolare, testeremo i classificatori definiti con funzioni kernel lineari e polinomiali di grado 3. In riferimento alle notazioni impiegate nel Capitolo 5, il parametro C sarà fissato al valore di 0.7, mentre τ verrà fatto variare.

Confidenza del classificatore

L'implementazione dei metodi SVM presente in GATE apporta una leggera modifica rispetto a quanto esposto nel Capitolo 5 (si veda [16] per ulteriori dettagli). Se un token è descritto tramite un vettore $\mathbf{x} \in \mathbb{R}^p$, ricordiamo che la costruzione di un classificatore binario corrisponde alla definizione di una funzione $f : \mathbb{R}^p \to \mathbb{R}$, il cui segno determina l'assegnazione del dato alla classe positiva o negativa. Ad esempio, la funzione f nel caso di una SVM lineare è della forma

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

Ora, è possibile introdurre un termine che indichi la confidenza con cui il dato $\mathbf{x}$ è classificato positivo (cioè una stima della probabilità che $\mathbf{x}$ sia effettivamente positivo). Indichiamo questa quantità $s(\mathbf{x})$ e la definiamo come

$$s(\mathbf{x}) := \frac{1}{1 + \exp\{-2f(\mathbf{x})\}} \tag{6.1}$$

Per classificare $\mathbf{x}$ è possibile imporre un valore di soglia S in modo che $\mathbf{x}$ sia classificato positivo se $s(\mathbf{x}) > S$. Poiché nel nostro caso il riconoscimento di un elemento diagnostico richiede la presenza contemporanea di un token iniziale $\mathbf{x}_{init}$ ed uno finale $\mathbf{x}_{fin}$, è possibile avere un'indicazione sulla confidenza attribuibile al chunk riconosciuto attraverso il prodotto $s(\mathbf{x}_{init})s(\mathbf{x}_{fin})$.

Postprocessing

Infine, l'implementazione SVM utilizzata elabora i risultati della classificazione attraverso delle operazioni di postprocessing. Le fasi di postprocessing sono tre.

1. **Controllo di consistenza:** si verifica che tutti i token classificati come iniziali abbiano un corrispettivo token finale (e viceversa). Gli eventuali casi isolati sono riclassificati come negativi.
2. **Controllo sulla lunghezza:** per ogni possibile sequenza composta da m token riconosciuta dal classificatore, si verifica la presenza nel training set di sequenze relative allo stesso tipo. Se nessuna di queste è composta da m token, la sequenza ritrovata viene ignorata. Si scartano quindi tutte le possibili sequenze che non hanno un corrispettivo nel training set della stessa lunghezza.
3. **Selezione finale:** si fissa una soglia S_{tot} di confidenza (solitamente 0.2). Di tutte le sequenze possibili rimaste dopo le prime due fasi, si considerano solamente quelle con confidenza $s(\mathbf{x}_{init})s(\mathbf{x}_{fin}) \geq S_{tot}$.

Nel nostro caso, scegliamo $S = 0.4$ ed $S_{tot} = 0.2$. Questi ultimi sono i valori di default proposti da GATE, scelti sulla base di esperimenti precedenti.

6 Set di dati ed analisi condotte

6.1 Set di dati

Le lettere di dimissione che si hanno a disposizione per l'analisi provengono dall'ospedale Uboldo di Cernusco sul Naviglio (MI)[5]. In particolare, sono

[5]Si ringrazia il Dott. Pietro Barbieri per la gentile concessione. Si precisa che ogni trattamento delle lettere di dimissione ospedaliera, essendo dati sensibili, è avvenuto sotto

state annotate manualmente 188 lettere (anamnesi ed esami, per un totale di 376 documenti) per formare il training set. Le stesse 188 lettere sono state l'oggetto dello studio linguistico che ha portato a definire le decision rules.

In seguito, saranno analizzate 50 nuove lettere di dimissione, provenienti dallo stesso ospedale, su cui verranno applicati sia due diversi algoritmi di machine learning (addestrati con il training set precedente), che le decision rules.

6.2 Analisi condotte

Per fornire una valutazione dell'applicazione di named entity recognition realizzata, si eseguiranno diversi test. In primo luogo, verranno valutati i classificatori in base alle feature utilizzate, attraverso una procedura di cross-validazione basata sul partizionamento del training set in 10 gruppi. Per ogni tipologia di feature (dalla I alla V), si valuteranno i seguenti algoritmi SVM:

- kernel lineare, tenendo conto dei dati sbilanciati con $\tau = 0.4$;
- kernel lineare, senza tenere conto dei dati sbilanciati ($\tau = 1$);
- kernel polinomiale di grado 3, $\tau = 0.4$;
- kernel polinomiale di grado 3, $\tau = 1$.

In seguito, si eseguirà un'analisi sull'influenza del parametro τ, nel caso di feature basate sui dizionari (II). Si registreranno i risultati facendo variare τ tra 0 ed 1.
Infine, testeremo l'applicazione di named entity recognition sul campione di 50 lettere più recenti, che non sono state usate né per i test precedenti, né per lo studio linguistico su cui si basano le decision rules. Per questo test finale, sceglieremo gli algoritmi nelle configurazioni più interessanti: SVM lineari, $\tau = 0.4$ con feature dapprima basate sui dizionari (II), in seguito basate su dizionari e decision rules (V).

la supervisione del Dott. Ballerio, responsabile dell'analisi linguistica.

Capitolo 7

Risultati

Esponiamo infine i risultati relativi alle diverse analisi condotte, descritte nel capitolo precedente. In primo luogo, per ogni scelta delle feature I - V[1] riportiamo i risultati al variare del tipo di metodo SVM usato. In seguito, mettiamo in risalto il miglioramento delle performance dei classificatori all'aumentare dell'informazione linguistica fornita, cioè passando da una scelta delle feature osservabili di tipo I fino al tipo V. Mostriamo poi l'influenza del parametro τ (relativo al trattamento di training set sbilanciati per le SVM) sul processo di named entity recognition. Infine, riportiamo i risultati ottenuti applicando i sistemi di named entity recognition alla collezione di lettere più recenti. Innanzi tutto però, è necessario precisare alcuni aspetti relativi al conteggio degli errori di misclassificazione.

Conteggio degli errori

Il sistema per il riconoscimento di elementi diagnostici fornisce in output le lettere di dimissione ospedaliera annotate. Ogni istanza di elemento diagnostico deve essere riconosciuta nel testo come una particolare successione di token, che viene quindi annotata. Ovviamente, se la successione annotata automaticamente non ha niente a che fare con l'elemento diagnostico riconosciuto, l'annotazione è considerata un errore, più precisamente un *falso positivo*. Se invece nel testo è descritto un elemento diagnostico ed il sistema non produce alcuna annotazione a riguardo, avremo a che fare con un *falso negativo*.
Ora, gli errori sono contati automaticamente da GATE, confrontando le annotazioni generate con le annotazioni manuali del training set. Come trattare i casi in cui un'annotazione manuale comprende più token rispetto alla successione annotata automaticamente? Devono essere considerati come falsi

[1] si fa riferimento alle notazioni introdotte nel Capitolo 6.

negativi? E quando l'annotazione automatica comprende dei token in più si avranno dei falsi positivi? È chiaro che una valutazione di questo tipo sarebbe troppo rigida. Lo scopo del sistema è infatti quello di individuare nel testo gli elementi diagnostici. Se l'ultima di una sequenza di parole che descrive un elemento diagnostico non è annotata, possiamo comunque affermare che il sistema ha riconosciuto correttamente quell'elemento. Le annotazioni parziali saranno perciò considerate corrette.

Gli indici per la valutazione dei sistemi di named entity recognition sono *precision*, *recall* e *F-measure*, definiti nel Capitolo 2 rispettivamente con le espressioni (2.24), (2.25) e (2.26). Essi possono essere calcolati seguendo due approcci diversi.
Il primo consiste nel considerare ogni tipologia di elemento diagnostico separatamente dalle altre. Gli indici saranno quindi calcolati in base al conteggio dei falsi positivi e falsi negativi solamente in relazione ad uno specifico elemento diagnostico. Questo metodo richiederà perciò cinque volte il calcolo dei tre indici, per ciascuno dei cinque tipi di elementi diagnostici.
Il secondo approccio prevede invece un conteggio globale degli errori, senza distinzione tra le tipologie di elementi diagnostici. Si dovranno calcolare in questo caso solamente i tre indici globali.

1 Influenza del tipo di algoritmo SVM

Sono stati testati algoritmi SVM con kernel lineare o polinomiale di grado 3. In entrambi i casi, si è scelta sia l'implementazione che tiene conto del training set sbilanciato ponendo $\tau = 0.4$, sia l'implementazione *classica*, che equivale a porre $\tau = 1$. I risultati si riferiscono ad un procedimento di cross-validazione basato su una partizione del training set in 10 sottoinsiemi. Nel seguito riportiamo gli indici *precision*, *recall* e *F-measure* calcolati in base ad un conteggio globale degli errori.

I **- Feature basate sui token:**

Algoritmo	**Precision**	**Recall**	**F-measure**
Lineare, $\tau = 0.4$	0.881	0.798	0.836
Lineare, $\tau = 1$	0.829	0.579	0.680
Polinomiale, $\tau = 0.4$	0.890	0.797	0.840
Polinomiale, $\tau = 1$	0.957	0.629	0.756

II - **Feature basate sui dizionari:**

Algoritmo	**Precision**	**Recall**	**F-measure**
Lineare, $\tau = 0.4$	0.924	0.832	0.875
Lineare, $\tau = 1$	0.933	0.746	0.828
Polinomiale, $\tau = 0.4$	0.918	0.861	0.887
Polinomiale, $\tau = 1$	0.941	0.790	0.857

III - **Feature basate su token e dizionari:**

Algoritmo	**Precision**	**Recall**	**F-measure**
Lineare, $\tau = 0.4$	0.901	0.838	0.868
Lineare, $\tau = 1$	0.929	0.767	0.839
Polinomiale, $\tau = 0.4$	0.901	0.869	0.884
Polinomiale, $\tau = 1$	0.953	0.752	0.839

IV - **Feature basate su decision rules e token:**

Algoritmo	**Precision**	**Recall**	**F-measure**
Lineare, $\tau = 0.4$	0.915	0.850	0.879
Lineare, $\tau = 1$	0.963	0.775	0.858
Polinomiale, $\tau = 0.4$	0.921	0.870	0.894
Polinomiale, $\tau = 1$	0.969	0.738	0.837

V - **Feature basate su decision rules e dizionari:**

Algoritmo	**Precision**	**Recall**	**F-measure**
Lineare, $\tau = 0.4$	0.950	0.837	0.888
Lineare, $\tau = 1$	0.962	0.808	0.877
Polinomiale, $\tau = 0.4$	0.918	0.889	0.902
Polinomiale, $\tau = 1$	0.955	0.842	0.894

Innanzi tutto, è bene specificare che nel nostro caso gli errori di *falso positivo* e *falso negativo* rivestono la stessa importanza. Poiché l'indice *precision* non tiene conto dei falsi negativi ed il *recall* non tiene conto dei falsi positivi, ci interessiamo soprattutto alla *F-measure*, che consiste in una media geometrica tra i due indici. Detto questo, le tabelle riportate in precedenza mostrano risultati più che soddisfacenti. Anche il modello più semplice (I), in cui si considerano solamente i token e per la cui costruzione non è richiesto alcun lavoro linguistico, il sistema è in grado di riconoscere con buona precisione la maggior parte degli elementi diagnostici riportati dalle

lettere di dimissione. Usando un kernel lineare con $\tau = 0.4$, se il sistema individua 100 elementi diagnostici, 88 sono corretti (*precision*). D'altro lato, se nei testi sono descritti 100 elementi diagnostici, il sistema ne riconosce correttamente 80 (*recall*).

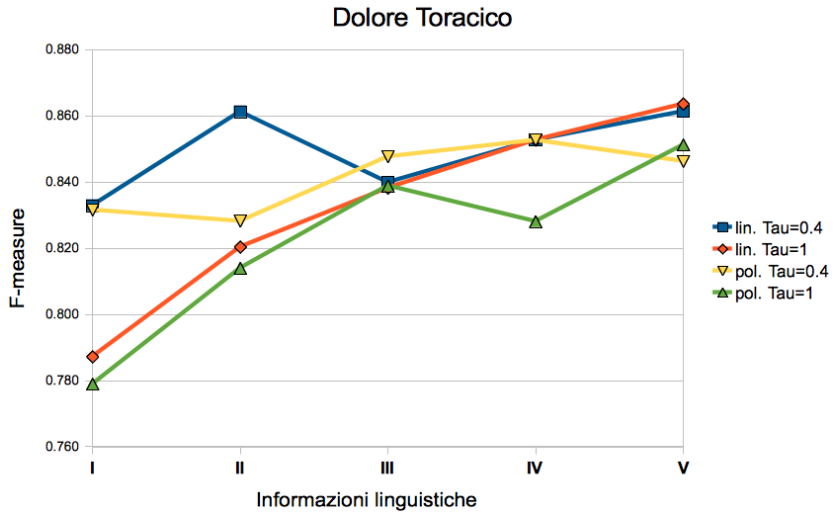
I risultati mostrano inoltre che, in tutti i casi, la scelta di $\tau = 0.4$ comporta un miglioramento nella qualità della classificazione. È perciò importante prendere in considerazione la composizione sbilanciata del training set, i cui dati sono per la maggior parte negativi. Presenteremo nel seguito un'analisi più approfondita sulla scelta del parametro τ.

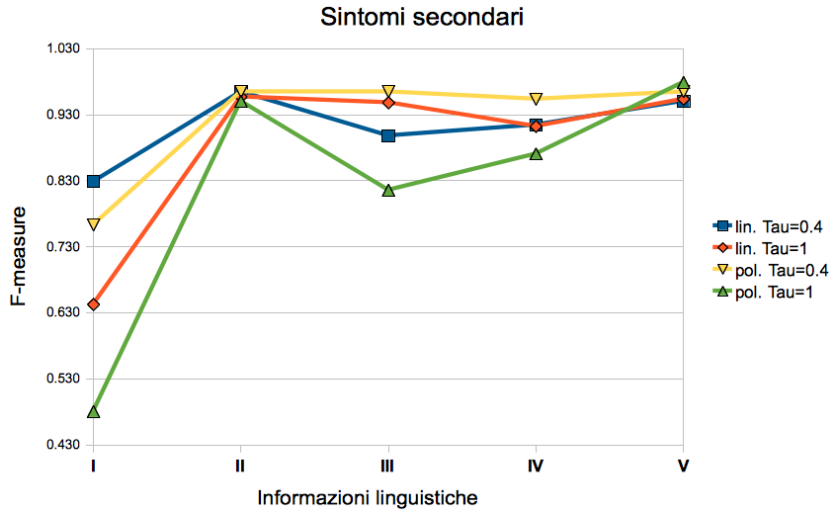
Per quanto riguarda la scelta delle funzioni kernel, i polinomi di ordine 3 sembrano garantire risultati leggermente migliori dei kernel lineari. In particolare, i polinomi permettono di ridurre i falsi negativi (fatta eccezione per le feature di tipo I), il che corrisponde ad un aumento del *recall*. Tuttavia, sebbene non sia specificato, i classificatori costruiti con kernel polinomiali richiedono uno sforzo computazionale maggiore che si traduce in un maggiore tempo di calcolo (circa 20% in più rispetto al caso lineare).

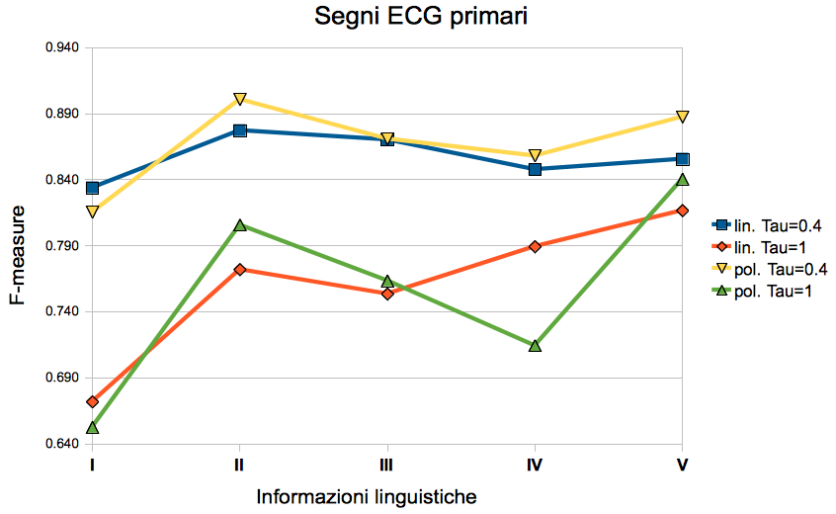
Infine, si può notare come strumenti linguistici quali dizionari e decision rules permettano di ottenere risultati migliori, rispetto ad un'analisi basata semplicemente sulle stringhe di caratteri che formano i token. Ci interessiamo meglio a questo aspetto nella prossima sezione.

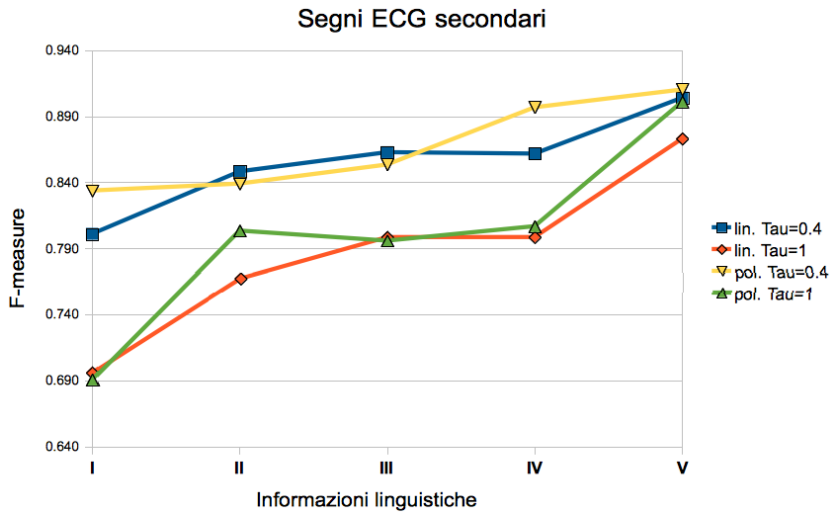
2 Influenza dell'informazione linguistica

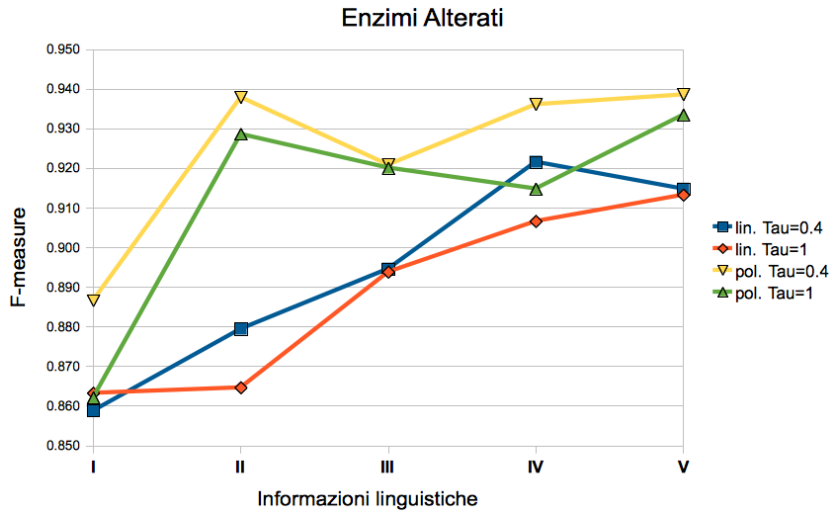
È ragionevole supporre che maggiori sono le informazioni linguistiche usate per costruire il modello, migliori saranno i risultati di classificazione. Per avere un'idea più precisa dei miglioramenti effettivamente introdotti grazie ai dizionari ed alle decision rules, riportiamo gli andamenti della *F-measure* al variare delle informazioni usate per costruire i vettori di feature, dal modello I al V. In questo caso, mostreremo i risultati divisi in base al tipo di elemento diagnostico. Saremo perciò in grado di valutare quali elementi diagnostici siano facilmente riconoscibili con poche informazioni linguistiche e quali invece richiedano un lavoro più approfondito. Su ognuno dei grafici seguenti, riportiamo l'andamento della *F-measure* ottenuta con classificatori SVM lineari e polinomiali, con $\tau = 0.4$ e $\tau = 1$.


Dolore Toracico
0.880
0.860
0.840
0.820
0.800
0.780
0.760
F-measure
I
II
III
IV
V
Informazioni linguistiche
lin. Tau=0.4
lin. Tau=1
pol. Tau=0.4
pol. Tau=1


Sintomi secondari
1.030
0.930
0.830
0.730
0.630
0.530
0.430
F-measure
I
II
III
IV
V
Informazioni linguistiche
lin. Tau=0.4
lin. Tau=1
pol. Tau=0.4
pol. Tau=1


Segni ECG primari
0.940
0.890
0.840
0.790
0.740
0.690
0.640
F-measure
I
II
III
IV
V
Informazioni linguistiche
lin. Tau=0.4
lin. Tau=1
pol. Tau=0.4
pol. Tau=1





Come avevamo supposto, l'andamento della *F-measure* è effettivamente crescente al crescere della conoscenza linguistica fornita al modello. In primo luogo, si può notare come l'uso dei dizionari (passaggio dal modello I al modello II) apporti sempre un miglioramento nei risultati, tranne che nell'unico caso isolato riguardante il dolore toracico, con algoritmo SVM polinomiale e $\tau = 0.4$ (primo grafico, linea gialla). Ciò mostra l'importanza di un lavoro linguistico in supporto all'applicazione di metodi di machine learning.

La costruzione di vettori di feature considerando contemporaneamente i lemmi dei dizionari ed i token (modello III) sembra invece non apportare evidenti miglioramenti rispetto al modello II. Ciò è visibile soprattutto per i

sintomi secondari e per i segni elettrocardiografici primari, per i quali il modello II dà risultati migliori rispetto al III. Per le altre tre classi di elementi diagnostici si osserva un andamento variabile, che vede dei leggeri miglioramenti o peggioramenti dei risultati a seconda dell'algoritmo SVM impiegato. L'aggiunta dei token fa aumentare notevolmente la dimensione p dei vettori che descrivono le unità statistiche da classificare. Inoltre, la variabilità talvolta imprevedibile del linguaggio medico e la presenza di errori di battitura aggiungono rumore ai dati. Questi due aspetti rendono probabilmente più imprecisi i classificatori nel caso si usino come feature sia i dizionari che i token.

L'aggiunta delle decision rules dà luogo ad ulteriori miglioramenti, ma vanno fatte alcune considerazioni. Concentriamoci innanzi tutto sul modello IV, che si basa sull'osservazione dei token e delle decision rules. Se lo confrontiamo al modello I, in cui si usano solo i token, il salto di qualità è evidente. Tuttavia, il miglioramento rispetto ai modelli II e III è meno rimarcato ed in qualche caso si assiste ad una lieve involuzione nelle performance (ad esempio considerando i segni ECG primari). Il modello V invece, che prevede l'utilizzo contemporaneo delle decision rules e dei dizionari, è in generale quello che produce i risultati migliori. Come ci eravamo aspettati, l'utilizzo del modello che comprende la maggiore quantità di informazioni linguistiche produce i sistemi di riconoscimento automatico migliori.

2.1 Machine learning e human learning

È doveroso aggiungere qualche ulteriore considerazione sull'analisi precedente. Osserviamo il problema da un punto di vista dell'apprendimento. Il modello I corrisponde ad un sistema in cui l'algoritmo impara *da solo* a riconoscere gli elementi diagnostici. Per istruirlo, è necessario fornire solamente un numero sufficiente di esempi annotati. Sarà l'algoritmo che, sulla base degli esempi, riconoscerà le parole interessanti e, per quanto possibile, ritroverà nel testo le strutture utili a riconoscere gli elementi diagnostici.

I modelli IV e V corrispondono invece al caso opposto. Il sistema di machine learning *non parte da zero*, ma si basa sulla conoscenza linguistica con cui sono stati definiti i vettori di feature. Per redigere i dizionari serve una conoscenza approfondita del gergo medico, oltre che una conoscenza linguistica di base. Per costruire le decision rules è inoltre necessaria un'attenta analisi delle lettere di dimissione, per ritrovare nel testo le strutture e le espressioni regolari che caratterizzano gli elementi diagnostici. Il sistema possiede quin-

di una grande conoscenza a priori, alla quale si aggiungerà in un secondo momento la conoscenza empirica fornita dagli esempi del training set. La fase di machine learning consiste in questo caso in una rifinitura delle conoscenze a priori fornite dagli strumenti linguistici: se l'esperto che ha scritto le decision rules non ha tenuto conto di alcune caratteristiche importanti, è compito dell'algoritmo sopperire alla mancanza.

Infine, i modelli II e III costituiscono un caso intermedio tra i due precedenti: si fornisce al sistema solamente la conoscenza linguistica data dai dizionari. La fase di machine learning si occupa in un secondo momento di ricercare le strutture regolari che caratterizzano gli elementi diagnostici, osservando gli esempi del training set.

Analizzando in quest'ottica i risultati mostrati in precedenza, possiamo confrontare la conoscenza apportata da uno studio linguistico approfondito con quella appresa automaticamente dall'algoritmo di machine learning. Il lavoro linguistico di maggior difficoltà risiede nella ricerca delle espressioni regolari che indicano gli elementi diagnostici, cioè la scrittura delle decision rules. Ora, sebbene le decision rules permettano di migliorare i classificatori, l'apporto maggiore è senz'altro dato dai dizionari. In altri termini, l'approccio machine learning si rivela competitivo con lo *human learning* nella fase di ricerca delle espressioni regolari. Per quanto riguarda i dizionari, invece, la conoscenza linguistica fornita a priori sembra essere di fondamentale importanza. Ciò è ragionevole se si considera che, nella redazione dei dizionari, le informazioni fornite non provengono solo dall'analisi testuale, ma da una conoscenza linguistica più generale, che non può essere acquisita semplicemente istruendo il sistema con poco più di un centinaio di lettere di esempio.

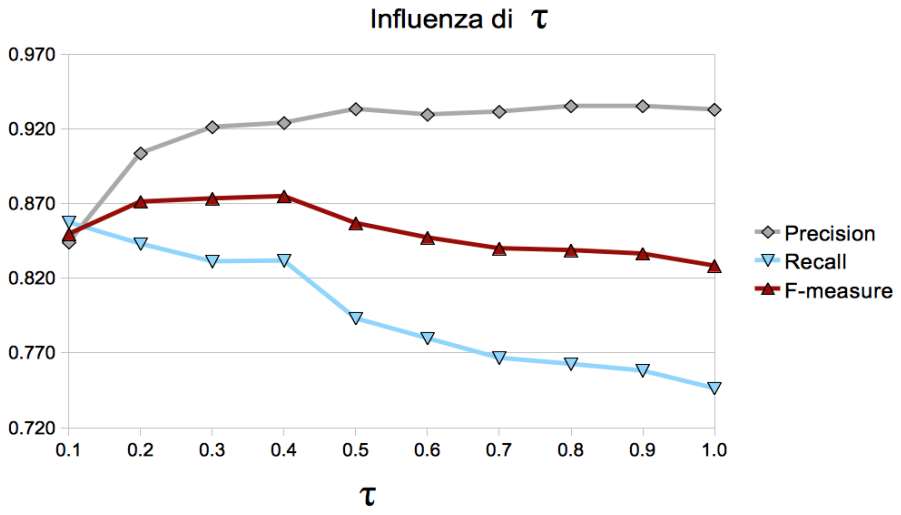
Un'ultima considerazione riguarda la generalità degli strumenti linguistici. Sebbene i dizionari siano stati redatti anche sulla base di un attento studio delle lettere a disposizione, essi costituiscono uno strumento di carattere più generale rispetto alle decision rules. I primi saranno ancora validi nel caso si vogliano analizzare lettere provenienti da contesti diversi, le seconde saranno invece più difficilmente applicabili, o quantomeno necessiteranno di un difficile lavoro di aggiornamento. Disponendo invece di dizionari sufficientemente completi, un sistema di machine learning basato su un modello di tipo II o III sarà facilmente aggiornabile, richiedendo solamente l'annotazione di alcune lettere di esempio da aggiungere al training set.

3 Influenza del parametro τ

Abbiamo visto che, nel caso si disponga di un training set in cui i dati negativi sono molto maggiori di quelli positivi, imporre $\tau = 0.4$ porta a dei miglioramenti nei classificatori. Abbiamo perciò analizzato l'influenza del parametro τ, facendolo variare da 0.1 a 1. I test sono stati realizzati applicando un modello di tipo II, con un algoritmo SVM lineare.

Aumentando τ, l'algoritmo diventa più selettivo e riduce il rischio di falsi positivi, a scapito però di qualche falso negativo. Valori elevati di τ coincidono quindi con una più alta *precision* ed un *recall* inferiore. Al contrario, valori bassi permettono di riconoscere più istanze di elementi diagnostici, a scapito di qualche falso positivo in più. In questo caso, il *recall* sarà alto e la *precision* bassa. Per ottimizzare la *F-measure*, dobbiamo quindi scegliere un valore intermedio.

Riportiamo di seguito in un grafico l'andamento di *precision*, *recall* e *F-measure*, riferiti ad un conteggio globale degli errori. Si noti come il valore $\tau = 0.4$, già indicato in [15] come ottimo per un'applicazione di named entity recognition, permetta anche in questo caso di ottenere i valori di *F-measure* maggiori.



4 Annotazione di nuovi documenti

Come ultimo test, abbiamo applicato i sistemi di riconoscimento automatico degli elementi diagnostici a 50 nuove lettere di dimissione. Le lettere, che provengono dallo stesso ospedale di quelle contenute nel training set, sono

più recenti. Chiaramente, non sono state usate per redigere i dizionari, né per scrivere le decision rules.

Dizionari (modello II)

In un primo momento, abbiamo testato il modello II, in cui le feature osservabili sono basate sui dizionari. Sebbene dalle analisi precedenti gli algoritmi SVM con kernel polinomiale sembrino garantire risultati migliori, i metodi lineari sono spesso più facilmente generalizzabili, e possono fornire risultati migliori in caso di dati nuovi. Scegliamo quindi un kernel lineare, con $\tau = 0.4$. Di seguito sono riassunti i risultati, che però risultano falsati da un problema legato alla codifica dei numeri nelle nuove lettere. Ciò comporta un deterioramento della *precision* relativa agli enzimi alterati. Per poter riconoscere correttamente un elemento diagnostico relativo alle alterazioni enzimatiche, è infatti spesso necessario distinguere tra valori al di sopra o al di sotto di una certa soglia (si veda il Capitolo 6, Sezione 1). Un problema di codifica dei numeri porta quindi il sistema a riconoscere come elementi diagnostici anche i casi in cui il valore registrato è al di sotto della soglia, aumentando i falsi positivi. Si consideri quindi che la *F-measure* relativa agli enzimi alterati potrebbe crescere fino a circa 0.87, mentre la *F-measure* calcolata globalmente potrebbe salire ad un valore di 0.86 rispetto ai valori indicati nel seguito. In ogni caso, l'errore influenza in modo contenuto i risultati. Riportiamo nella tabella seguente i valori relativi all'analisi.

Elemento diagnostico	**Precision**	**Recall**	**F-measure**
Dolore toracico	0.827	0.896	0.860
Sintomi secondari	1.000	1.000	1.000
Segni ECG primari	0.794	0.844	0.818
Segni ECG secondari	0.909	0.800	0.851
Enzimi Alterati	0,869	0.779	0.822
Globale	0.852	0.833	0.843

Decision rules e dizionari (modello V)

Infine, abbiamo applicato il modello V, sempre con un algoritmo SVM lineare e $\tau = 0.4$, ottenendo i risultati riassunti nella tabella sottostante. L'errore sulla codifica dei numeri non è influente in questo caso, poiché le decision rules comprendono già il confronto con i valori di soglia e processano i documenti in modo diverso, aggirando il problema.

Elemento diagnostico	**Precision**	**Recall**	**F-measure**
Dolore toracico	0.830	0.936	0.880
Sintomi secondari	1.000	1.000	1.000
Segni ECG primari	0.794	0.818	0.806
Segni ECG secondari	1.000	0.840	0.913
Enzimi Alterati	0.984	0.882	0.930
Tot.	0.903	0.883	0.893

Si noti come i risultati siano migliori nel caso delle decision rules, in particolare per quanto riguarda gli enzimi alterati (valgono le considerazioni fatte in precedenza riguardo alla codifica dei numeri) e i segni ECG secondari. Per gli altri tre tipi di elementi diagnostici, l'uso dei soli dizionari dà risultati simili a quelli ottenuti impiegando anche le decision rules. Queste ultime si rivelano ancora una volta strumenti utili a migliorare i classificatori, ma allo stesso tempo poco influenti per alcuni elementi diagnostici.

Conclusioni

Nella prima parte, questa tesi è consistita essenzialmente in un lavoro bibliografico, da cui è stata tratta l'esposizione generale sul text mining. Le applicazioni ad esso legate rivestono un'importanza fondamentale, data la sovrabbondanza di risorse testuali digitali che contengono una quantità di informazioni ingestibili. Per questo motivo il text mining è un ambito di studio in continuo sviluppo e di grande attualità. Nonostante ciò, pochi sono gli esempi in letteratura di testi che riportano una trattazione unitaria e generale dell'argomento, la maggior parte dei quali consiste nella pubblicazione in un unico volume di più articoli, ciascuno relativo ad un preciso studio di caso. Si spera quindi che il presente lavoro possa diventare una base ed uno stimolo per ulteriori lavori di ricerca sul text mining al Politecnico.

La seconda parte del lavoro si è invece concentrata su uno specifico caso di studio, nell'ambito del *Progetto IMA* per il Servizio Sanitario della Lombardia. Utilizzando le tecniche di estrazione delle informazioni illustrate nel Capitolo 4 e le support vector machines presentate nel Capitolo 5, è stato realizzato un sistema di machine learning in grado di riconoscere con una buona precisione gli elementi che definiscono una diagnosi di infarto miocardico acuto. Per realizzare tale sistema, è stato usato un software libero sviluppato dall'Università di Sheffield, GATE, che offre diverse funzionalità per realizzare operazioni di Natural Language Processing e per l'applicazione di tecniche di machine learning. Si è sfruttato inoltre il grande lavoro di supporto linguistico realizzato dal Dott. Stefano Ballerio, che ha fornito strumenti fondamentali per il miglioramento degli algoritmi di classificazione supervisionata alla base del sistema realizzato.

L'applicazione alle lettere di dimissione ospedaliera ha mostrato risultati più che positivi. Senza avvalersi di alcuno strumento linguistico, il sistema è in grado di riconoscere l'80% degli elementi diagnostici riportati dalle lettere (*recall*). Inoltre, di tutti gli elementi indicati dal sistema, l'88% corrisponde correttamente ad un elemento diagnostico (*precision*). L'aggiunta degli stru-

menti linguistici permette di incrementare il *recall* fino all'89% e la *precision* fino al 92%. Anche il test su lettere di dimissione più recenti ha dato buoni risultati, simili ai precedenti.

In conclusione, possiamo affermare che il text mining costituisce un potenziale supporto per l'estrazione di informazioni dai testi scritti presenti nei database del sistema sanitario e, più in generale, permette il trattamento dei dati, in pratica inaccessibili per la quantità e la mancanza di strutturazione, provenienti dai database testuali amministrativi in generale.

Appendice A

Cenni sui modelli di linguaggio statistici

Il text mining è strettamente legato agli ambiti di studio della linguistica computazionale e del natural language processing. Può essere quindi utile una brevissima introduzione sui modelli statistici di linguaggio più diffusi nelle teorie linguistiche. Tali modelli stanno alla base di applicazioni come il riconoscimento vocale, la traduzione automatica e metodi avanzati di POS tagging.

Un modello statistico del linguaggio consiste nell'assegnazione di una misura di probabilità sulle sequenze di parole (o simboli) che compongono una comunicazione orale o scritta in linguaggio naturale. Supponiamo che il nostro linguaggio sia composto dai simboli (ad esempio, parole e punteggiatura) contenuti nell'insieme Θ. Se consideriamo una parte di comunicazione, ad esempio una frase, essa sarà composta da una sequenza $\{x_1, ..., x_n\}$ di elementi di Θ. Si vuole perciò assegnare una probabilità ad ogni possibile sequenza.

Nel seguito, introduciamo le idee alla base dei principali modelli di linguaggio statistici, senza però una descrizione formale approfondita dei metodi.

1 N-grams

Gli N-grams descrivono il linguaggio come un processo markoviano. L'insieme degli stati coincide allora con Θ, mentre indichiamo con X_i la variabile che descrive il sistema all'istante i. Data la sequenza $\{x_1, ..., x_{i-1}\}$, si cerca una distribuzione di probabilità per lo stato successivo X_i. Per un modello

di ordine k, la proprietà di Markov si esprime nel modo seguente:

$$\begin{aligned}&\mathbb{P}(X_i = x_i \,|\, X_{i-1} = x_{i-1}, X_{i-2} = x_{i-2}, ..., X_1 = x_1) \;= \\ &= \;\mathbb{P}(X_i = x_i \,|\, X_{i-1} = x_{i-1}, ..., X_{i-k} = x_{i-k})\end{aligned} \tag{A.1}$$

Definizione: un N-gram è un modello markoviano di ordine N-1.

In genere, nonostante l'estrema complessità del linguaggio naturale, si utilizzano modelli di basso ordine, noti come uni-gram, bi-gram, tri-gram e four-gram[1]. Ciò è dovuto soprattutto alle risorse computazionali richieste da modelli N-gram con N grande.

1.1 Probabilità di una frase

Per esprimere la probabilità di una sequenza di simboli $\mathbb{P}(\{x_1, ..., x_n\})$, che possiamo considerare come una frase, si è soliti introdurre dei simboli deterministici di inizio frase, che indichiamo con s. Prendendo ad esempio un bi-gram, avremo allora:

$$\begin{aligned}\mathbb{P}(\{X_1 = x_1, ..., X_n = x_n\}) \;&= \;\mathbb{P}(X_1 = x_1|X_0 = s)\,\mathbb{P}(X_2 = x_2|X_1 = x_1)\,... \\ &...\,\mathbb{P}(x_n|X_{n-1} = x_{n-1})\end{aligned}$$

Disponendo di una sufficiente quantità di frasi di esempio, in genere si stima la probabilità $\mathbb{P}(X_i = x_i|X_{i-1} = x_{i-1}, ..., X_{i-k} = x_{i-k})$ come:

$$\mathbb{P}(X_i = x_i|X_{i-1} = x_{i-1}, ..., X_{i-k} = x_{i-k}) \;\simeq\; \frac{\#\,\{x_i, ..., x_{i-k}\}}{\#\,\{x_{i-1}, ..., x_{i-k}\}}$$

dove "#" si riferisce al conteggio delle volte in cui la sequenza compare nelle frasi di esempio.

1.2 Uni-gram: modello bag of words

La semplificazione più estrema consiste nel trascurare ogni legame tra i simboli che compongono una frase. È il caso degli uni-gram, per i quali la presenza di una certa parola non dipende dalle altre parole nella frase, né tantomeno dall'ordine in cui compaiono.
Supponiamo allora di descrivere con un uni-gram un intero documento. Il modello che si ottiene coincide con quello descritto nel Capitolo 1, detto bag of words. Per ogni documento, ciò che conta è la presenza o meno dei simboli del linguaggio (che possiamo identificare con i token), non il loro ordine.

[1] In generale, i linguisti non apprezzano l'uso di questi termini, poiché insieme alla desinenza greca *gram* non compaiono i corretti prefissi greci *mono*, *di*, *tri* e *tetra*.

In generale, ciò che noi descriviamo come un vettore-documento costruito con un modello bag of words (qualsiasi sia la scelta dei pesi - binari, *tf* o *tf-idf*) sarà indicato da uno studioso di linguistica come un uni-gram sul documento.

2 Hidden Markov Models

Uno sviluppo dei modelli markoviani consiste nel considerare gli stati del sistema *nascosti*. Introduciamo allora l'insieme Q degli stati. Come nel caso precedente, indichiamo Θ l'insieme dei simboli osservabili del linguaggio (che non coincidono più con lo stato del sistema).

Una realizzazione di un *Hidden Markov Model* consiste allora in una sequenza $\{q_1, ..., q_n\} \subset Q$ di stati ed una sequenza $\{x_1, ..., x_n\} \subset \Theta$ di simboli. Ad ogni istante i, il sistema, che si trova nello stato q_i, emette il simbolo x_i. x_i è realizzazione della variabile aleatoria X_i, la cui distribuzione di probabilità dipende solo dallo stato del sistema q_i.
Come si è detto, gli stati del sistema non sono osservabili, ma sono solo deducibili dall'osservazione dei simboli emessi $x_1, ..., x_n$.

Non andiamo oltre nella descrizione degli Hidden Markov Models, rimandando ad esempio a [7] (Capitolo VII) per ulteriori dettagli.

3 Modello *back-off* di Katz

Per stimare le distribuzioni di probabilità associate ad un N-gram, è necessaria una raccolta di frasi di esempio. Ora, più N è grande, più sarà difficile individuare negli esempi sequenze di N simboli, meno le stime saranno accurate. Per questo, e per questioni legate a risorse di calcolo, si usano in pratica solo modelli con N piccolo (al massimo 4). Tuttavia, scegliendo N piccolo, si descrive in modo impreciso il linguaggio. Come fare per utilizzare modelli con N più grande? L'idea di Katz è la seguente: si potrebbe utilizzare N elevato solo per le sequenze di N parole sufficientemente frequenti negli esempi. Quando si ha una sequenza rara, si potrebbe allora passare ad un modello con N inferiore.

Il modello *back-off* di Katz prevede allora l'utilizzo contemporaneo di più N-grams, con $N = 1, ..., k+1$. Cerchiamo un'espressione per la probabilità

$$\mathbb{P}(X_i = x_i \,|\, X_{i-1} = x_{i-1}, ..., X_{i-k} = x_{i-k}, ..., X_1 = x_1) \tag{A.2}$$

In un primo momento, si cerca di utilizzare il modello più completo. Se le frasi di esempio non lo consentono, si passerà ai modelli ridotti. Fissiamo un valore di soglia ϕ. Avremo allora i seguenti modelli per (A.2).

- $(k+1)$**-gram:** se nelle frasi di esempio vale $\#\{x_i, ..., x_{i-k}\} > \phi$, allora (A.2) è stimato con

$$\mathbb{P}(X_i = x_i \,|\, X_{i-1} = x_{i-1}, ..., X_{i-k} = x_{i-k}) \simeq \frac{\#\{x_i, ..., x_{i-k}\}}{\#\{x_{i-1}, ..., x_{i-k}\}}$$

- k**-gram:** altrimenti, se nelle frasi di esempio vale $\#\{x_i, ..., x_{i-k+1}\} > \phi$, allora (A.2) è stimato con

$$\mathbb{P}(X_i = x_i \,|\, X_{i-1} = x_{i-1}, ..., X_{i-k} = x_{i-k+1}) \simeq \frac{\#\{x_i, ..., x_{i-k+1}\}}{\#\{x_{i-1}, ..., x_{i-k+1}\}}$$

- si procede allo stesso modo, scendendo di livello se necessario, eventualmente fino ad un uni-gram.

In realtà, il modello di Katz prevede altri coefficienti, che vanno a loro volta stimati. Non si vuole in quest'appendice entrare troppo nei dettagli, che sono spiegati in [12].

Appendice B

Introduzione all'uso di GATE

1 Cos'è GATE

GATE (*General Architecture for Text Engineering*) è un software distribuito liberamente, sviluppato dall'università di Sheffield ed in continuo sviluppo da oltre quindici anni. È utilizzato e sostenuto da una grandissima comunità, composta sia da studenti di linguistica computazionale e ricercatori, che da varie imprese. Fornisce strumenti per operare praticamente ogni tipo di analisi computazionale che riguardi il linguaggio naturale scritto.

Diverse sono le funzionalità di GATE. È infatti composto da una *famiglia* di prodotti, che comprende:

- **GATE Developer -** un ambiente di sviluppo integrato (in inglese *IDE - integrated development environment*), ovvero un software che assiste i programmatori nello sviluppo delle applicazioni di Natural Language Processing e text mining;
- **GATE Teamware -** una piattaforma funzionante sul web che fornisce un'interfaccia per la manipolazione di documenti di testo, oltre che per la definizione di strumenti di text mining e la gestione di progetti a distanza;
- **GATE Embedded -** un framework comprendente una serie di librerie che permettono di esportare le componenti realizzate con GATE Developer ed utilizzarle in diverse applicazioni;
- **CREOLE -** una serie di *plugin* che contengono le principali risorse linguistiche ed algoritmiche per costruire applicazioni di text mining e Natural Language Processing;

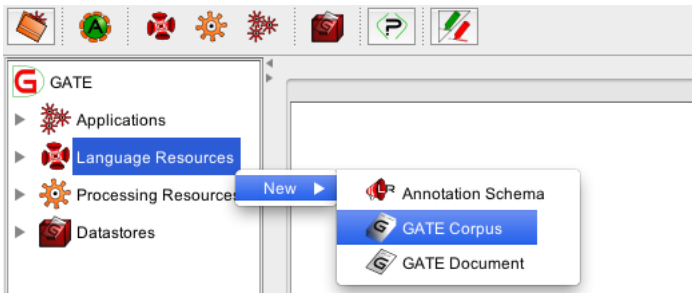
- **JAPE -** un linguaggio basato su Java per creare *decision rules* e grammatiche, per annotare automaticamente ed in forma standard i testi.

Nel seguito, si vuole dare una breve illustrazione di come GATE sia stato utilizzato per il lavoro riguardante questa tesi, nella speranza che possa essere utile in futuro ad altri studenti o ricercatori alle prime armi con software di Natural Language Processing e Text Mining. Si consiglia in ogni caso la lettura della guida [8]. Per una brevissima guida che illustri passo dopo passo attraverso un esempio alcuni possibili impieghi di GATE, consigliamo il tutorial[1] di Brian Davis disponibile on-line all'indirizzo `http://gate.ac.uk/wiki/quick-start/`.

2 Importazione di una collezione di documenti

Il primo passo per una qualsiasi analisi di text mining consiste nella creazione di un set di dati, o meglio di una collezione di documenti. Ciò può essere realizzato ripercorrendo i passi riportati di seguito.

- **Creare un Corpus:** nella schermata principale di GATE Developer, cercare nel menu sulla sinistra la voce *Language Resources*. Cliccare con il tasto destro e selezionare *new*, poi *GATE Corpus*. Assegnare il nome desiderato alla collezione di documenti, ad esempio *My_corpus*. Il nuovo Corpus dovrebbe essere visibile sotto la voce *Language Resources*.



- **Caricare nel Corpus i documenti:** cliccando con il tasto destro sull'icona del corpus appena creato, selezionare la voce *Populate*, indicando in seguito la cartella che contiene i testi e la codifica di questi (Unicode, UTF-8, ecc.).

[1] Il tutorial in questione mostra solo alcuni aspetti di GATE e non riguarda in alcun modo l'utilizzo di metodi di machine learning. Si consiglia comunque di leggerlo per prendere familiarità con gli strumenti linguistici implementati in GATE (validi soprattutto per la lingua inglese) e con le regole di annotazione definite attraverso JAPE.

- **Creare un Datastore:** per salvare la collezione di documenti mantenendo le manipolazioni successivamente effettuate con GATE, è preferibile la creazione di un Datastore. Si consiglia di creare in un primo momento una cartella apposita. In seguito, nell'ambiente GATE Developer, cercare la voce *Datastore* nel menu a sinistra. Cliccare con il tasto destro, selezionare *Create Datastore* e scegliere il tipo (nel nostro caso, *SerialDataStore*). Il Datastore sarà quindi creato e avrà il nome della cartella scelta.
- **Salvare la collezione di documenti nel Datastore:** per salvare le collezioni di documenti, cliccare con il tasto destro sull'icona del corpus e selezionare *Save to Datastore*.

3 Annotazione dei documenti

Per la costruzione di un qualsiasi classificatore supervisionato, è necessario disporre di un training set annotato. Esistono diversi modi di creare delle annotazioni, ma il più intuitivo è senza dubbio l'annotazione manuale. Prima di mostrare come annotare manualmente il testo, è utile fare qualche precisazione.

3.1 Struttura delle annotazioni

Annotaiton name

Nell'ambito della linguistica computazionale, molti degli elementi descritti nei capitoli precedenti sono identificati tramite annotazioni. Ad esempio, la fase di *tokenization* farà in modo che, nel testo, ogni sequenza di caratteri che costituisce un token sia riconoscibile tramite l'annotazione "*Token*". Se invece operiamo una fase di *named entity recognition* per individuare delle date, la stringa di caratteri che nel testo indica una data sarà contraddistinta da un'annotazione "*Data*". Esistono perciò diversi *tipi* di annotazioni, indicati in GATE come *annotation name*. In questo modo, un token sarà indicato tramite un'annotazione il cui annotation name è *Token*, mentre una data sarà indicata da un'annotazione il cui annotation name è *Data*.

Feature

Ogni annotazione non è contraddistinta solo dall'annotaion name. Possono infatti essere definite una o più *feature*, ovvero delle caratteristiche legate all'oggetto annotato. Per ogni feature si assegna poi un valore. Ad esempio, un'annotazione il cui annotation name è *Data* può essere caratterizzata da

una feature *Precisione*, cui sarà assegnato il valore *generica* se la data è generica (ex. *l'anno scorso*, *qualche mese fa*, ecc.) oppure il valore *precisa* se indica il giorno esatto (ex. *31 marzo 2011*).
Le feature, così come gli annotation name, sono definite dall'utente.

Annotation set

Infine, gli annotation name possono essere raggruppati in insiemi, detti *annotation set*. Qualora siano stati definiti molti annotation name e si voglia suddividerli in base alla loro utilizzazione, è possibile creare un annotation set per ogni tipo di analisi che si vuole condurre sui documenti. Ogni annotation set conterrà allora gli annotation name utili per quell'analisi.

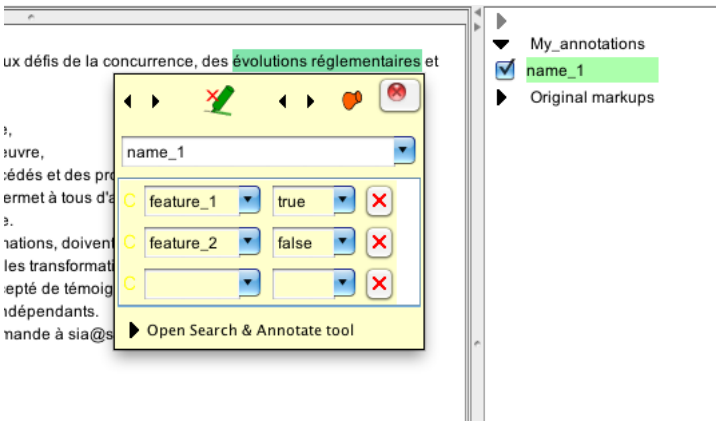
3.2 Annotazione manuale

Vediamo dunque come annotare manualmente un testo. Il procedimento è piuttosto intuitivo. Innanzi tutto, bisogna visualizzare il testo da annotare in GATE Developer (è sufficiente un doppio click sull'icona relativa al testo in esame). Si consiglia poi di seguire le seguenti procedure:

- Nella schermata, sopra il testo vi sono alcuni pulsanti per mostrare o nascondere gli strumenti di annotazione. Selezionare *Annotation Sets*, *Annotation List* e *Text*. Sulla destra compare l'elenco degli annotation name (di default è presente *Original Markups*), in basso la lista delle annotazioni selezionate.

- Creare un nuovo annotation set attraverso il pulsante *new* (in basso a destra nella schermata), impostando ad esempio il nome *My_annotations*.

- Selezionare nell'elenco degli annotation set *My_annotations* con un click. Ora, per definire una nuova annotazione, selezionare la stringa di testo che si vuole annotare. Lasciando per pochi istanti il cursore del mouse sulla stringa selezionata, si aprirà automaticamente una finestra. Sarà allora possibile definire l'annotation name, ad esempio *name_1*, e delle feature con i rispettivi valori, ad esempio *feature_1* con valore *true* e *feature_2* con valore *false*. Dopo aver annotato tutto il testo, è possibile visualizzare le varie annotazioni selezionandole e deselezionandole dall'elenco sulla destra. Esse saranno evidenziate nel testo con colori diversi.



- Per salvare il documento annotato, cercarne l'icona nel menu a sinistra, cliccare con il tasto sinistro e selezionare *save to its datastore*.

4 Utilizzo delle risorse linguistiche

Un insieme completo di risorse linguistiche, principalmente per la lingua inglese, è implementato in GATE con il pacchetto ANNIE (*A Nearly New Information Extraction system*). Per utilizzarne alcuni strumenti, lo carichiamo cliccando su *File*, *Manage CREOLE Plugins* e selezionando nell'elenco *ANNIE*. Selezioniamo inoltre, per utilizzare le funzioni che vedremo in seguito, le voci *Learning* e *Machine_Learning*.
Vediamo dunque come assemblare una serie di strumenti per processare i testi realizzando delle annotazioni automaticamente.

Tokenization

GATE fornisce un Tokenizer generico, funzionante su testi scritti in qualsiasi lingua. Per utilizzarlo, è sufficiente cercare *Processing Resources* nel menu di sinistra, cliccare con il tasto destro, selezionare *new*, poi *GATE Unicode Tokenizer*. Inserire un nome, ad esempio *My_tokenizer* e specificare il tipo di codifica nel campo *encoding string* (di default UTF-8, potrebbe essere unicode, UTF-9, ecc.).

Dizionari

Un dizionario è composto fondamentalmente da uno o più elenchi di parole (più precisamente, elenchi di stringhe di caratteri). Qualora queste siano ritrovate nei documenti, saranno annotate automaticamente da GATE. Le annotazioni così create avranno annotation name *Lookup*, e saranno contraddistinte da diverse feature.

- *majorType* e *minorType* - i valori di queste feature dipendono dall'elenco in cui si trova la parola. Per ogni elenco definito, l'utente dovrà quindi specificare questi due valori.
- Ulteriori feature definite dall'utente. Può essere ad esempio interessante assegnare ad ogni parola del dizionario ruolo grammaticale, genere e numero. L'utente dovrà allora specificare queste caratteristiche durante la redazione delle liste.

Per creare un dizionario sono allora necessari due elementi (nell'Appendice C riportiamo alcuni file di esempio):

1. Uno o più file di testo che contengono gli elenchi di parole con le relative feature e i valori di queste (da salvare con estensione `.lst`).
2. Un file di testo di definizione del dizionario, contenente l'elenco di tutte le liste da includere, che riporta anche per ciascuna lista i valori di *majorType* e *minorType* (da salvare con estensione `.def`).

Caricare un dizionario Per caricare un dizionario in GATE Developer, selezionare nel menu a sinistra *Processing Resources* (click destro), *new*, *Hash Gazetteer*. A questo punto, scegliere un nome (ad esempio *My_dictionnary*), specificare la codifica, se si vuole un sistema case sensitive o meno e indicare il percorso del file di definizione del dizionario.

5 Decision rules

È possibile introdurre delle decision rules basate sulle annotazioni esistenti. Attraverso una decision rule si stabiliscono precise condizioni da verificare nel testo. Quando queste condizioni sono verificate, alla parte di testo interessata verrà assegnata un'annotazione definita nella decision rule stessa.

Le decision rules si definiscono con JAPE, e sono in genere chiamate *grammatiche*. Non entriamo nei dettagli della scrittura di una grammatica JAPE, poiché sarebbe necessario un approfondimento sulla sintassi. Tuttavia, si tenga presente che la sintassi JAPE prende spunto da Java (non solo, può accettare istruzioni Java). Per un'illustrazione della sintassi JAPE si rimanda alla guida ed al tutorial citati in precedenza.

Per caricare una grammatica JAPE, come al solito, selezioniamo dal menu *Processing Ressources* (click destro), poi *new* e *Jape Transducer*. Al solito, indichiamo un nome (potrebbe essere *My_rule*), la codifica dei testi ed il percorso del file con estensione `.jape` che definisce la grammatica.

6 Applicazioni

Abbiamo visto come definire degli strumenti in grado di generare automaticamente annotazioni. Resta ancora un ultimo passo da compiere per eseguirli: la creazione di un'applicazione.

Creazione di un'applicazione Per creare un'applicazione, basta un click destro sulla voce *Applications* del menu, selezionando *new*, *corpus pipeline* e definendo un nome, ad esempio *My_application*. L'applicazione così creata processerà tutti i documenti del corpus in una sola volta. Per salvare l'applicazione, è sufficiente cliccare con il tasto destro su *My_application* nel menu e selezionare *Save Application State*.

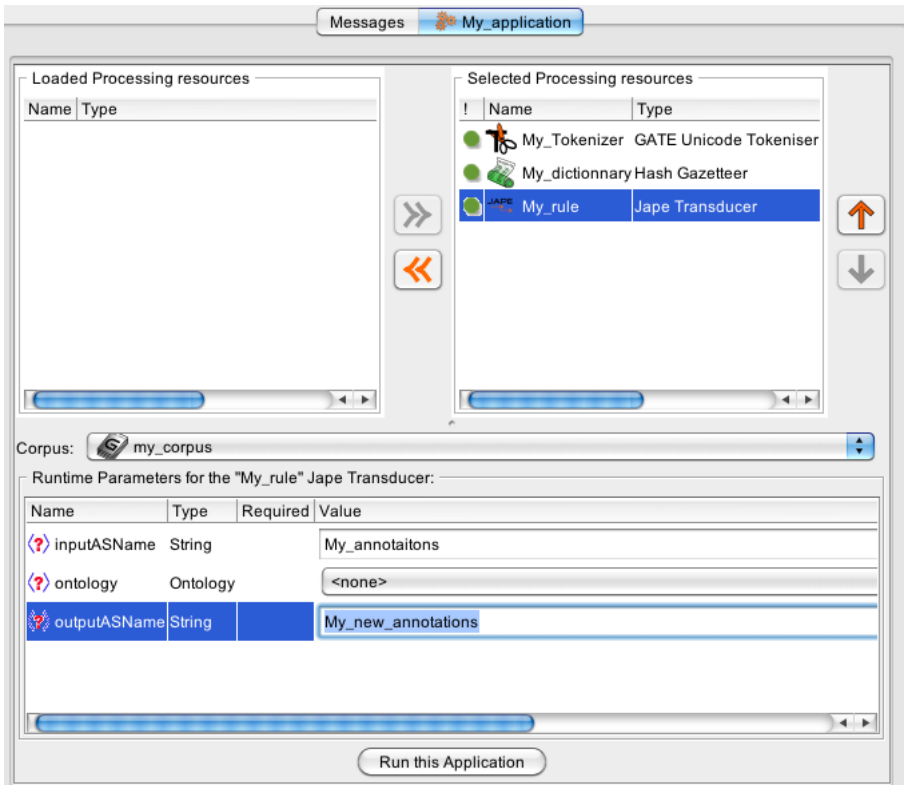
Costruzione dell'applicazione Una volta creata, l'applicazione va costruita assemblando gli strumenti visti nelle sezioni precedenti. Supponiamo di voler comporre *My_application* con i seguenti processi:

1. tokenization, utilizzando *My_tokenizer*;
2. stemming e POS-tagging attraverso il dizionario *My_dictionnary*;
3. annotazione automatica tramite la grammatica JAPE *My_rule*.

Per farlo, basta seguire poche semplici istruzioni. Con un doppio click su *My_application* nel menu si visualizza la finestra per comporre l'applicazione. Si devono allora selezionare uno alla volta dall'elenco *Loaded Processing resources* i processi con cui vogliamo comporre l'applicazione. È importante notare che l'ordine con cui i processi compaiono nell'elenco *Selected Processing resources* è fondamentale. Nel nostro caso si deve avere:

1. *My_tokenizer*
2. *My_dictionnary*
3. *My_rule*

Durante l'inserimento del dizionario e della grammatica JAPE, è possibile scegliere gli annotation set in cui lavorare. Con la grammatica JAPE è inoltre possibile creare un nuovo annotation set, che conterrà le annotazioni definite dalla decision rule.
Infine, si deve indicare il corpus da processare e cliccare su *Run this Application*.



7 Machine Learning con GATE

Tra i processi che si possono aggiungere ad un'applicazione, vi sono anche degli algoritmi di classificazione supervisionata. Sono implementati in GATE i seguenti metodi:

- Knn (*k nearest neighbors*)
- Naive Bayes
- SVM (*Support Vector Machine* con kernel lineari o polinomiali)
- PAUM (*Perceptron Algorithm with Uneven Margins*)
- l'algoritmo *decision tree C4.5*

Notazione: per costruire un classificatore, è necessario descrivere ogni unità statistica attraverso un vettore di feature. Quello che abbiamo chiamato fino ad ora *vettore di feature* sarà chiamato nel seguito *vettore di attributi*. Il termine *feature* è infatti già usato in GATE per designare le caratteristiche legate alle annotazioni (si veda la sezione precedente sulle annotazioni). Questa terminologia è adottata anche nella guida [8] (*attributes*).

7.1 File di configurazione

Per usare un metodo di classificazione supervisionata, è necessario creare un file di configurazione. I principali elementi da specificare sono:

- l'unità statistica considerata (ex. l'intero documento per un'applicazione di text categorization o il token per la *named entity recognition*);
- le feature da considerare per ogni unità statistica;
- la feature (unica) che determina la classificazione (il gruppo in cui l'elemento è classificato);
- l'algoritmo da usare.

A parte la scelta dell'algoritmo, gli elementi specificati nel file di configurazione dipendono dalle annotazioni dei documenti. Si tenga presente che tutte le annotazioni usate per la classificazione supervisionata devono appartenere allo stesso annotation set.
La sintassi del file di configurazione è di tipo XML. Sono riportati nell'Appendice C alcuni esempi di file di configurazione. Di seguito, si danno delle indicazioni su come scrivere il file di configurazione per un'applicazione di *text categorization*. Supponiamo inoltre che tutti i documenti siano già classificati, e che quindi la differenza tra training set e test set sia fatta in un secondo momento. Per ulteriori dettagli sulla struttura del file di configurazione si rimanda alla guida [8] (capitolo dedicato al Machine Learning).

Supponiamo di voler descrivere i documenti attraverso un modello bag-of-words. Per costruire il classificatore, è necessario un training set annotato nel seguente modo:

- Il testo intero di ogni documento presenta un'annotazione con annotation name *documento*, e con la feature *classe*. Il valore assegnato alla feature *classe* indica la categoria in cui è classificato il documento.
- I testi sono stati processati con una fase di tokenization; in questo modo ogni elemento annotato come token ha una feature *string*, il cui valore è la stringa di caratteri

Vediamo allora gli elementi da specificare nel file di configurazione.

Tag iniziali e finali

La prime due linee del file di configurazione devono contenere le seguenti istruzioni:

```
<?xml version="1.0"?>
<ML-CONFIG>
```

L'ultima linea del file deve essere invece:

```
</ML-CONFIG>
```

Algoritmo e metodo di valutazione

Vogliamo usare il metodo knn. Per farlo includiamo l'istruzione:

```
<ENGINE nickname="KNN" implementationName="KNNWeka" />
```

Vogliamo inoltre testare il classificatore con un procedimento di cross-validation, suddividendo il training set in 5 gruppi ed utilizzando ogni volta un gruppo diverso come test set. Includiamo allora l'istruzione:

```
<EVALUATION method="kfold" runs="5"/>
```

Definizione degli attributi per il vettore-documento

Le prossime istruzioni devono essere tutte contenute tra le seguenti due tag:

```
<DATASET>
...
</DATASET>
```

Innanzi tutto, va specificata l'unità statistica, indicando un annotation name che la contraddistingua. In riferimento alle annotazioni descritte per questo esempio, indichiamo:

```
<INSTANCE-TYPE>documento</INSTANCE-TYPE>
```

In seguito, vogliamo semplicemente usare un modello bag-of-words, che tenga conto di tutti i token presenti in ogni documento. Come spiegato nell'Appendice A, tale modello corrisponde ad un uni-gram, che definiamo nel modo seguente (poiché si richiede di assegnare un nome all'N-gram, scegliamo *MyNgram*):

```
<NGRAM>
      <NAME>MyNgram</NAME>
      <NUMBER>1</NUMBER>
      <CONSNUM>1</CONSNUM>
      <CONS-1>
            <TYPE>Token</TYPE>
            <FEATURE>string</FEATURE>
      </CONS-1>
</NGRAM>
```

Per indicare i pesi usati per il vettore documento, si sceglie un valore tra 1 e 3 (1 per pesi binari, 2 per pesi *tf*, 3 per pesi *tf-idf*). Scegliamo pesi *tf-idf*, indicati dall'istruzione:

```
<ValueTypeNgram>3</ValueTypeNgram>
```

Definizione delle etichette per la classificazione

Abbiamo supposto che le annotazioni *dcumento* abbiano una feature *classe* che indica il gruppo in cui il documento è classificato (in altri termini, l'*etichetta* della classificazione). Per definirlo nel file di configurazione, si devono aggiungere le seguenti istruzioni, sempre all'interno delle tag `<DATASET>` e `<\DATASET>`:

```
<ATTRIBUTE>
      <NAME>Etichetta</NAME>
      <SEMTYPE>NOMINAL</SEMTYPE>
      <TYPE>documento</TYPE>
      <FEATURE>class</FEATURE>
      <POSITION>0</POSITION>
      <CLASS/>
</ATTRIBUTE>
```

Per una spiegazione del significato di tutti i termini presenti nelle istruzioni XML si rimanda alla guida di GATE [8].

7.2 Creazione di un'applicazione di machine learning

Dopo aver scritto e salvato il file di configurazione, si deve creare in GATE Developer l'applicazione di machine learning corrispondente.
Selezionare la voce *Processing Resources* dal menu di sinistra. Click destro, scegliere *new*, poi *Batch Learning PR*. Assegnare un nome, ad esempio *My_algorithm*, ed indicare il percorso per il file di configurazione.

In seguito, creare una nuova applicazione (come visto in precedenza), chiamandola ad esempio *My_classificator*. Con un doppio click sulla voce *My_classificator* si accede alla schermata per costruire l'applicazione. Aggiungere *My_algorithm* alle *Selected Processing resources*, ed indicare il corpus annotato su cui si vuole applicare la classificazione.
Si è inoltre interessati a testare l'algoritmo tramite cross-validation, quindi per la voce *LearningMode* selezionare *EVALUATION*.

Il classificatore è ora pronto per essere usato. Cliccando su *Run This Application*, verranno processati i documenti. I risultati della classificazione e della valutazione saranno salvati in specifici file di output. Per una descrizione di tali file, rimandiamo alla guida [8].

Se invece volessimo costruire un classificatore per poi applicarlo ad una collezione di documenti non ancora annotata, è necessario seguire un procedimento in due fasi. Innanzi tutto, è necessario disporre di due corpus di documenti, il primo contenente il training set annotato (supponiamo che sia nominato *Corpus1*) ed il secondo su cui vogliamo applicare il classificatore (*Corpus2*). Supponiamo che l'applicazione *My_Classificator* sia impostata come visto in precedenza. In un primo momento, si deve scegliere la modalità di esecuzione *TRAINING* e lanciare l'applicazione su training set (*Corpus1*). Una volta terminata l'elaborazione, si può passare alla modalità *APPLICATION* e processare *Corpus2* per ottenere la classificazione dei documenti.

Appendice C

Codici

File di configurazione per le applicazioni di machine learning

Riportiamo solo i codici per definire i modelli I, II e V.

Modello I

```
<?xml version="1.0"?>
<!--
FILE DI CONFIGURAZIONE PER IL MODELLO CON
LE FEATURE COSTRUITE CON I TOKEN.
SVM LINEARE, TAU=0.4
/-->
<ML-CONFIG>

<SURROUND value="true"/>
<multiClassification2Binary method="one-vs-others"/>
<EVALUATION method="kfold" runs="10"/>

<ENGINE nickname="SVM" implementationName="SVMLibSvmJava"
options=" -c 0.7 -t 0 -m 100 -tau 0.4 "/>


<DATASET>

<INSTANCE-TYPE>Token</INSTANCE-TYPE>

<!-- ATTRIBUTO CHE DEFINISCE I GRUPPI DELLA CLASSIFICAZIONE -->
<ATTRIBUTE>
<NAME>Classe</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>ElementoDiagnostico</TYPE>
<FEATURE>class</FEATURE>
```

```
<POSITION>0</POSITION>
<CLASS/>
</ATTRIBUTE>

<!-- ATTRIBUTO: token in una finestra DA -8 A +8 -->
<ATTRIBUTELIST>
<NAME>sorround_tokens</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>Token</TYPE>
<FEATURE>string</FEATURE>
<RANGE from="-8" to="8"/>
</ATTRIBUTELIST>

<!-- ATTRIBUTO: tipo di token (parola, numero o punteggiatura) -->
<ATTRIBUTELIST>
<NAME>sorround_token_type</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>Token</TYPE>
<FEATURE>kind</FEATURE>
<RANGE from="-8" to="8"/>
</ATTRIBUTELIST>

<!-- ATTRIBUTO: numeri (divisi in 4 gruppi) -->
<ATTRIBUTELIST>
<NAME>numbers</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>number</TYPE>
<FEATURE>gruppo</FEATURE>
<RANGE from="0" to="2"/>
</ATTRIBUTELIST>

</DATASET>
</ML-CONFIG>
```

Modello II

```
<?xml version="1.0"?>
<ML-CONFIG>

<!--
FILE DI CONFIGURAZIONE PER MODELLO CON FEATURE
BASATE SUI DIZIONARI.
SVM POLINOMIALE, TAU=0.4
/-->
```

```
<SURROUND value="true"/>
<multiClassification2Binary method="one-vs-others"/>
<EVALUATION method="kfold" runs="10"/>

<ENGINE nickname="SVM" implementationName="SVMLibSvmJava"
options=" -c 0.7 -t 1 -d 3 -m 250 -tau 0.4 "/>

<DATASET>

<INSTANCE-TYPE>Token</INSTANCE-TYPE>

<ATTRIBUTE>
<NAME>Classe</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>ElementoDiagnostico</TYPE>
<FEATURE>class</FEATURE>
<POSITION>0</POSITION>
<CLASS/>
</ATTRIBUTE>

<ATTRIBUTELIST>
<NAME>Lookup_maj</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>Lookup</TYPE>
<FEATURE>majorType</FEATURE>
<RANGE from="-3" to="3"/>
</ATTRIBUTELIST>

<ATTRIBUTELIST>
<NAME>Lookup_min</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>Lookup</TYPE>
<FEATURE>minorType</FEATURE>
<RANGE from="-3" to="3"/>
</ATTRIBUTELIST>

<ATTRIBUTELIST>
<NAME>Lookup_lemma</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>Lookup</TYPE>
<FEATURE>lemma</FEATURE>
<RANGE from="-3" to="3"/>
</ATTRIBUTELIST>

<ATTRIBUTELIST>
```

```
<NAME>Lookup_POS</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>Lookup</TYPE>
<FEATURE>pos</FEATURE>
<RANGE from="-3" to="3"/>
</ATTRIBUTELIST>

<ATTRIBUTELIST>
<NAME>numbers</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>number</TYPE>
<FEATURE>gruppo</FEATURE>
<RANGE from="0" to="2"/>
</ATTRIBUTELIST>

</DATASET>

</ML-CONFIG>
```

Modello V

```
<?xml version="1.0"?>
<ML-CONFIG>

<!--
FILE DI CONFIGURAZIONE PER MODELLO CON FEATURE
BASATE SULLE DECISION RULES E I TOKEN
SVM LINEARE, TAU=0.4
/-->

<SURROUND value="true"/>
<multiClassification2Binary method="one-vs-others"/>
<EVALUATION method="kfold" runs="10"/>
<ENGINE nickname="SVM" implementationName="SVMLibSvmJava"
options=" -c 0.7 -t 0 -m 250 -tau 0.4 "/>

<DATASET>

<INSTANCE-TYPE>Token</INSTANCE-TYPE>

<!-- ATTRIBUTO CHE DEFINISCE I GRUPPI DELLA CLASSIFICAZIONE -->
<ATTRIBUTE>
<NAME>Classe</NAME>
```

```
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>ElementoDiagnostico</TYPE>
<FEATURE>class</FEATURE>
<POSITION>0</POSITION>
<CLASS/>
</ATTRIBUTE>

<ATTRIBUTE>
<NAME>DecisionRules</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>RULE</TYPE>
<FEATURE>class</FEATURE>
<POSITION>0</POSITION>
</ATTRIBUTE>

<ATTRIBUTELIST>
<NAME>numbers</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>number</TYPE>
<FEATURE>gruppo</FEATURE>
<RANGE from="0" to="2"/>
</ATTRIBUTELIST>

<ATTRIBUTELIST>
<NAME>sorround_tokens</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>Token</TYPE>
<FEATURE>string</FEATURE>
<RANGE from="-8" to="8"/>
</ATTRIBUTELIST>

</DATASET>

</ML-CONFIG>
```

Esempio di dizionario

Riportiamo il codice relativo all'esempio di dizionario presentato nel cap. 5. Supponiamo che il codice sia salvato in un file nominato `sintomi.lst`:

```
palpitazione&gnr=f&lemma=palpitazione&nmb=s&pos=N
palpitazioni&gnr=f&lemma=palpitazione&nmb=p&pos=N
sudorazione&gnr=f&lemma=sudorazione&nmb=s&pos=N
sudorazioni&gnr=f&lemma=sudorazione&nmb=p&pos=N
algido&gnr=m&lemma=algido&nmb=s&pos=A
```

```
algidi&gnr=m&lemma=algido&nmb=p&pos=A
algida&gnr=f&lemma=algido&nmb=s&pos=A
algide&gnr=f&lemma=algido&nmb=p&pos=A
```

Per richiamare il dizionario, è necessario un file con estensione `.def` contenente una linea con l'istruzione:

```
sintomi.lst:minType:majType
```

dove *minType* e *majType* definiscono i valori delle feature *majorType* e *minorType*.

Bibliografia

[1] Arthur, D., Vassilvitskii, S., *k-means++: the advantages of careful seeding*. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. pp. 1027–1035, 2007.h

[2] Borthwick. A., *A Maximum Entropy Approach to Named Entity Recognition*, Ph.D. thesis, New York University, 1999.

[3] Clark, A., Fox, C., Lappin, S., *The Handbook of Computational Linguistics and Natural Language Processing*, Wiley-Blackwell, 2010.

[4] Darroch, J., Ratcliff, D., *Generalized iterative scaling for log-linear models*, The Annals of Mathematical Statistics, vol. 43, pag. 1470–1480, 1972.

[5] Feinerer I., *Introduction to the tm Package Text Mining in R*, The Comprehensive R Archive Network, 2007.

[6] Feinerer I., Hornik K., Meyer D., *Text Mining Infrastructure in R*, Jounal of Statistical Software, Volume 25, Issue 5, March 2008 (URL: `http://www.jstatsoft.org/v25/i05/`).

[7] Feldman, R., Sanger, J., *The Text Mining Handbook - Advances approaches in analyzing unstructured data*, Cambridge University Press, 2007.

[8] Funk, A., Maynard, D., et al., *Developing Language Processing Components with GATE*, University of Sheffield, 2010 (URL: `http://gate.ac.uk/userguide`).

[9] Hayes, P., Weinstein, S., *A system for content-based indexing of a database of news stories*, in Proceedings of the 2nd Conference on Innovative Applications of Artificial Intelligence, pag. 49–66, AAAI Press, Menlo Park, CA, 1990.

[10] Jardine, N., Van Rijsbergen, *The use of hierarchical clustering in information retrieval*, Information Storage and Retrieval, vol. 7, pag. 217–240, 1971.

[11] Kao, A., Poteet, S. R., *Natural Language Processing and Text Mining*, Springer, 2006.

[12] Katz, S. M., *Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer*, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 35, issue 3, 1987.

[13] Kecman, V., *Learning and Soft Computing - Support Vector Machines, Neural Networks, and Fuzzy Logic Models*, the MIT Press, 2001.

[14] Lafferty, J., McCallum, A., Pereira, F., *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*, In Proceedings of ICML-01, pag. 282–289, Morgan Kaufmann, San Francisco, CA, 2001.

[15] Li, Y., Bontcheva, K., Cunningham, H., *Using uneven margins SVM and perceptron for information extraction*, CONLL '05 Proceedings of the Ninth Conference on Computational Natural Language Learning, 2005.

[16] Li, Y., Bontcheva, K., Cunningham H., *SVM Based Learning System For Information Extraction*, in Winkler, M. N. J., Lawerence, N., editors, *Deterministic and Statistical Methods in Machine Learning*, LNAI 3635, pages 319–339, Springer Verlag, 2005.

[17] Luhn, H., *Auto-encoding of documents for information retrieval systems*, in M. Boaz, editor, Modern Trends in Documentation, pag. 45–58, Pergamon Press, London, 1959.

[18] Manning, C. D., Schütze, H., *Foundations of Statistical Natural Language Processing*, The MIT Press, 2000.

[19] Maron, M., Kuhns, J., *On relevance, probabilistic indexing and information retrieval*, Journal of the ACM, vol. 7, pag. 216–244, 1960.

[20] Perim, G., Wandekokem, E., Varejão, F., *K-Means Initialization Methods for Improving Clustering by Simulated Annealing*, Advances in Artificial Intelligence – IBERAMIA 2008, Lecture Notes in Computer Science, Springer, 2008.

[21] Porter, M., *An algorithm for suffix stripping*, Program, vol. 14, issue 3, 1980.

[22] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0, 2010 (URL: `http://www.R-project.org`).

[23] Ratnaparkhi, A., *A maximum entropy part-of- speech tagger*, Computational Linguistics, vol. 21, issue 4, 1995.

[24] Rousseeuw, P.J., *Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis*, Journal of Computational and Applied Mathematics, vol. 20, 1987.

[25] Salton, G., *A document retrieval system for man-machine interaction*, in Proceedings of the 19th Annual International ACM National Conference, pages L2.3.1–L2.3.20, ACM Press, New York, 1964.

[26] Salton, G., Wong, A., Yang, C., *A vector space model for automatic indexing*, Communications of the ACM, vol. 18, pag. 613–620, 1975.

[27] Srivastava, A., Sahami, M., *Text Mining - Classification, Clustering and Applications*, Chapman & Hall - CRC Data Mining and Knowledge Discovery Series, 2009.

[28] Struyf, A., Hubert, M., Rousseeuw, P., *Clustering in an Object-Oriented Environment*, Journal of Statistical Software, vol. 1, issue 4, 1997.

[29] Weiss, S. M., Indurkhya, N., Zhang, T., Damerau, F. J., *Text Mining - Predictive methods for analyzing unstructured information*, Springer, 2005.