ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Seconda Facoltà di FACOLTÀ DI INGEGNERIA
Corso di Laurea in INGEGNERIA DEI SISTEMI E DELLE TECNOLOGIE DELL'INFORMAZIONE

CLASSIFICAZIONE DI DOCUMENTI PRE-ELABORATI CON TECNICHE DI STEMMING

Elaborata nel corso di: SISTEMI INFORMATIVI DISTRIBUITI

Tesi di Laurea di:
ALESSANDRO PIRELLI

Relatore:
Prof. GIANLUCA MORO

Co-relatore:
Ing ROBERTO PASOLINI

ANNO ACCADEMICO 2012-2013
SESSIONE SESSIONE IV

Indice

Capitolo 1

Introduzione

L'obiettivo del text mining è studiare metodi e algoritmi per estrarre automaticamente conoscenza da testo non strutturato, come pagine web, email, forum e documenti in generale, utile per classificare o raggruppare documenti in base ai contenuti. Le applicazioni del text mining sono numerose. Una parte fondamentale nel text mining è la pre-elaborazione del testo per generare variabili, note con il termine di feature, rilevanti per le fasi successive di analisi, modellazione ed estrazione di conoscenza utile rispetto agli obiettivi prefissati.
Tra le possibili pre-elaborazioni c'è lo stemming delle parole, ossia il processo di riduzione della parola alla sua radice che non corrisponde in generale alla radice morfologica, i.e. al lemma. La creazione di algoritmi di stemming fa parte dell'area dell'informatica che si occupa dell'elaborazione del linguaggio naturale ed è impiegata anche nei motori di ricerca per riscrivere le interrogazioni degli utenti.

Nella tesi vengono messi a confronto due algoritmi che eseguono il processo di lemmatizzazione, ovvero la trasformazione di qualsiasi parola in forma ridotta, alla corrispettiva forma presente nel vocabolario per velocizzare il processo di clustering, in particolare l'algoritmo derivato dallo studio di Lovins [34] e l'algoritmo iterativo di Porter [42]. Sono testati entrambi gli algoritmi menzionati sia nella versione completa e sia in quella parziale, ossia in quest'ultimo caso eseguendo solo alcuni dei passaggi previsti dall'intero algoritmo. Dopo la pre-elaborazione vengono condotti esperimenti applicando algoritmi tradizionali di classificazione all'insieme dei documenti. Infine questi risultati di classificazione elaborati con algoritmi classici

sono confronti con un nuovo metodo di classificazione introdotto in questa tesi, basato sull'individuazione di associazioni statistiche tra parole di documenti diversi che trattano e che non trattano il medesimo argomento.

Capitolo 2

Introduzione al Text Mining

L'applicazione della statistica per l'analisi dei testi in linguaggio naturale risale agli anni '60, ma grazie al rapido aumento della disponibilità di testi digitali (Zampolli, Calzolari 1995) e alle pubblicazioni di testi in internet, quindi già pronti per l'analisi, ha reso il processo di analisi automatica dei testi sempre più interessante, sia dal punto di vista accademico, che dal punto di vista commerciale.
Inizialmente le soluzioni sviluppate si basavano principalmente sulla statistica, ma vengono introdotte metodologie diverse, provenienti da varie discipline, come la linguistica e l'umanistica [7].
Si è quindi passati da un metodo di tipo linguistico, a uno di tipo lessicale, arrivando a un analisi di tipo testuale o infine lessico-testuale.

2.1 Elaborazione del linguaggio naturale

La linguistica computazionale è l'applicazione dell'informatica alla linguistica teorica e applicata. Essa studia i problemi teorici e applicativi relativi al linguaggio naturale e al suo uso nell'informatica. Lo sviluppo si sviluppa in due ambiti, cioè nella realizzazione di:

- strumenti informatici per lo studio e la ricerca sul linguaggio;
- applicazioni informatiche di largo uso (correttori ortografici, information retrieval) che sfruttano competenze linguistiche applicate all'informatica.

Per linguaggio naturale si intendono tutte le lingue utilizzate per la comunicazione verbale fra gli esseri umani. Il termine naturale nasce come contrapposizione a linguaggio formale, inteso come linguaggio artificiale completamente formalizzato e, possibilmente, privo di ambiguità, di cui viene fatto largo uso in informatica (ad esempio, tutti i vari linguaggi di programmazione sono linguaggi formali).

2.1.1 Cenni storici

Storicamente, l'elaborazione del linguaggio naturale si suddivide tra diverse discipline, prendendo diversi nomi:

- Linguistica computazionale;
- Elaborazione del linguaggio naturale;
- Riconoscimento del parlato;
- Neurolinguistica;

ognuna delle quali promossa da una disciplina che studia aspetti diversi, dalla linguistica alla psicologia, passando dall'informatica e dall'elettronica[53].

1940-1950 - II guerra mondiale

Vengono presentati i primi modelli applicabili all'informatica,

- Automi a stati finiti;
  - teoria dei linguaggi formali (algebra e teoria degli insiemi per la formalizzazione dei linguaggi);
  - Chomsky, Backus, Naur (grammatiche BNF per la formalizzazione dei linguaggi).
- Algoritmi probabilistici per il riconoscimento del parlato
  - Sviluppo della teoria dell'informazione (Shannon), che non riguarda la forma o il contenuto, ma si basa sulle procedure di trasmissione e ricevimento:

    - rumorosità del canale;
    - codifica e decodifica;
    - entropia di un linguaggio.

La Machine Translation è una delle prime applicazioni desiderate (soprattutto a scopi militari, durante la Guerra Fredda).

Due paradigmi in contrapposizione

Nel dopo guerra, vengono sviluppati due paradigmi che analizzano i linguaggi su aspetti duali:

- Simbolico;
- Stocastico.

Il primo studia il linguaggio naturale tramite regole e grammatiche. Vengono utilizzate fortemente le regole. Si sviluppa quindi la teoria dei linguaggi formali (algoritmi di parsing top-down e bottom-up). Appaiono i primi esempi di intelligenza artificiale (teorie logiche, sviluppo di sistemi domanda-risposta tramite pattern matching, ricerca di keyword e semplici euristiche).
Il secondo parte da documenti reali e li analizza per estrarre le "regole" probabilistiche che li regolano. Uso debole delle regole, che possono essere trasgredite. Metodo bayesiano, tramite uso di dizionari e corpora. Riconoscimento di testi scritti tramite lo sviluppo dell'OCR.

Aumentano i paradigmi

Con il passare del tempo, fra gli anni '70 e '80, vengono sviluppati nuovi paradigmi, osservando nuovi aspetti delle problematiche dell'elaborazione linguistica naturale:

- Stocastico;
- Logic-based;
- Natural language under standing;
- Discourse modelling.

Vengono proposti gli Hidden Markov Model (HMM), modelli stocastici in cui si assume che il sistema sia modellato da un processo di Markov con stati inosservati (hidden).
Si unificano le strutture in feature (si riconoscono le interconnessioni tra le parti del discorso, in modo più performante rispetto alle grammatiche context-free).
Terry Winograd sviluppa SHRDLU, uno dei primi programmi per computer per la comprensione di un linguaggio.
Appaiono le prime analisi delle sottostrutture del discorso, con la risoluzione automatica dei riferimenti.

Empiricismo e FSModels

Negli anni '80 ci sono pochi finanziamenti per il campo dell'elaborazione del linguaggio naturale. L'uso della rappresentazione denotativa aveva ostacolato tutta l'intelligenza artificiale e i pochi risultati avevano fatto perdere fiducia ai finanziatori. La logica è troppo limitata rispetto alla realtà: rende impossibile risolvere alcuni problemi. Sono quindi necessarie altre rappresentazioni. L'uso delle grammatiche è troppo lento per fornire buoni risultati: scrivere una grammatica completa ed efficace per una lingua richiede anni, ma la lingua si evolve molto più in fretta.

- Ritorno all'utilizzo dei modelli a stati finiti, per la fonologia, la morfologia e la sintassi;
- Ritorno all'empiricismo: lavori di IBM per il riconoscimento del parlato basandosi su modelli probabilistici; approcci datadriven (ossia più incentrati su dati preesistenti che non su un modello) per il POS tagging, il parsing e l'annotazione, per la risoluzione delle ambiguità.
- Natural Language Generation.

I due campi si incontrano

L'approccio simbolico e quello stocastico si stanno riunendo. Le difficoltà incontrate con il primo trovano nuove vie di soluzione. Si uniscono ad un pesante uso delle metodologie data-driven e dei modelli probabilistici.
Nascono nuovi ambiti applicativi, come il Web, e nuove possibilità dovute all'aumentata capacità di elaborazione dei sistemi.

Si sono raggiunti risultati positivi nel parsing, nei modelli di interazione, nella morfologia e nell'uso di dizionari e corpora.

2.1.2 Le diverse unità di analisi del testo

Il problema essenziale per un'analisi automatica di un testo è riconoscerne il senso. Con il termine parola si indica convenzionalmente l'unità di analisi del testo. A seconda degli obiettivi, tale unità può essere una forma grafica, un lemma, un poliforme o un'unità mista (lessia), in grado di catturare al meglio il contenuto presente nel testo.
Nella statistica testuale, le analisi basate sulle forme grafiche hanno il vantaggio di essere indipendenti dalla lingua. Si tratta di un approccio puramente formale che privilegia i segni (significanti) per arrivare al senso (in quanto insieme di significati) come rappresentazione del contenuto o del discorso.
Il senso linguistico, come noto, è composto di un significante distito dal punto di vista "fonico" (parlato) e/o "grafico" (scritto) e di un significato a sua volta distinto dal punto di vista della "forma" (come classe "sintattica": grammatica, morfologia e sintassi) e della "sostanza" (come classe "semantica"). L'analisi statistica, secondo i cosiddetti formalisti, è condotta "a prescindere dal significato delle unità di testo". Il senso (significato/accezione) di una parola è determinato dalle parole che la circondano (asse sintagmatico), ma anche dalla selezione delle altre parole che possono rimpiazzarla nella stessa frase (asse paradigmatico); ossia dall'insieme delle parole che possono essere sostituite fra loro nel sintagma, senza modificare la struttura dell'enunciato, poichè "funzionano" in maniera equivalente. Il senso sottostate un testo/discorso, di cui s'intende dare una rappresentazione con metodi statistici, è costitutito dal sistema dei significati che "si tiene" (come una sorta di ecosistema) sulla base dell'insieme delle co-occorenze dell'intero corpus di dati testuali. Accanto a questa tradizione statistica di tipo "formalista", negli stessi anni, alcuni linguisti di tradizione harrisiana sistematizzano la formalizzazione linguistica di particolari classi di parole, di forme composte e sviluppano strumenti concreti di lessicografia e linguistica computazionali, quali dizionari elettronici e automi/trasduttori a stati finiti per la descrizione di grammatiche locali. I linguisti quantitativi, cimentandosi nei primi tentativi di lemmatizzazione automatica, mettono a punto nuovi lessici di frequenza. In Italia, grazie a un lemmatizzatore dell'IBM,

De Mauro costruisce un prototipo di vocabolario elettronico della lingua italiana (Veli) e il lessico dell'italiano parlato (Lip).

2.1.3 Un metodo per l'analisi automatica dei testi

Per dare un adeguata rappresentazione del corpus, dopo il parsing del testo secondo un'opportuna unità di analisi, occorrono diversi step integrati fra loro in una filiera. Pensare a filiere in tale ambito non vuol dire "cristallizzare" le procedure possibili, in un contesto in cui se ne possono concepire infinite varianti, bensì fissare soltanto alcuni passi fondamentali per un'analisi automatica del testo.
Le principali fasi che individuano una filiera "ideale" sono quattro:

1. preparazione del testo;
2. analisi lessicale;
3. estrazione d'informazione;
4. analisi testuale.

La fase di preparazione è essenziale per una corretta scansione del testo secondo l'unità di analisi prescelta. Questa fase andrebbe sempre più consolidata, per creare degli standard nel trattamento dei dati testuali, ancora lontani dall'essere comunemente condivisi. Essa consiste in primo luogo nella pulizia (definizione del set di caratteri alfabeto/separatori, spoliazione dei formati di gestione del testo (XML o altro) e nella normalizzazione del testo consistente nell'uniformare spazi, apostrofi e accenti, riconoscere a priori entità particolari (date, numeri, valute, titoli, sigle, abbreviazioni), nonché nomi, toponimi, società, personaggi o espressioni e locuzioni d'interesse. Per queste ultime, un problema consiste nella loro fixedness: la "stabilità", intesa come univocità di significato, non sempre può essere garantita (ad esempio: "una volta" o "a volte" hanno un senso variabile; diverso è il caso di poliremiatiche come "punto di vista", "carta di credito" che hanno un loro significato).
Ma fanno parte ancore di questa fase "preliminare" i differenti step di annotazione del testo che consistono nell'associare meta-informazioni alle parole. Fra queste: la categoria grammaticale, il lemma di appartenenza, una eventuale etichettatura semantica, possibili tagging di tipo relazionale (quali

sinonimie, iper/iponimie o altri link previsti nelle ontologie), il numero di occorrenze nel corpus, alcune caratteristiche morfologiche o altro, tutte annotazioni sfruttabili nelle tre fasi successive. Esistono software in grado di gestire questo livello di meta-informazioni sul testo, in maniera trasparente rispetto alla lettura automatica del testo.

La fase di analisi lessicale fornisce una rappresentazione paradigmatica del corpus: lo studio del suo vocabolario, ossia del linguaggio. È un'analisi di tipo "verticale" in cui la rappresentazione del testo è fatta senza poter tener conto dello sviluppo del discorso ma solo estraendo le parole come da un'urna, che in questo contesto viene chiamata "bag of words". Ricostruire il lessico con un "corpus" vuol dire produrre statistiche sui verbi, avverbi, sostantivi, aggettivi, ossia le principali classsi di parole cosiddette "piene" (di contenuto) evidenziandone le più frequenti, ma anche quelle appartenenti a determinati gruppi morfologici (enclitiche verbali unite ai pronomi personali; derivati; esotismi), utili per evidenziare alcune costanti di quel lessico, particolarmente significative.
Un ulteriore livello di analisi "verticale" riguarda lo studio delle parole "vuote" (connettivi, preposizioni, congiunzioni, determinanti, interiezioni), degli incipit di frase, della punteggiatura, della lunghezza e struttura della frase o altre analisi d'interesse più strettamente linguistico.
In particolare, con gli strumenti della Staitstica, l'analisi lessicale consente una descrizione di alcune costanti del linguaggio, in termini d'incidenza percentuale di alcune classi di parole (imprinting) in grado di differenziare i testi originari, di individuarne il livello e il tipo (l'incidenza del vocabolario di base (VdB), la presenza di discoro astratto/concreto, il tono positivo/negativo).

La fase di estrazione di informazione costituisce un momento importante dell'analisi di un testo, in quanto porta a concentrare l'attenzione su quella parte del linguaggio che risulta particolarmente significativa. Tale fase è utile per selezionare il cosiddetto linguaggio peculiare, ossia quel 12-15% di vocabolario in genere più rilevante per condurre l'analisi testuale. Potremmo distinguere due situazioni, rispettivamente generate con o senza una qualche query. L'estrazione di linguaggio peculiare senza l'input di una specifica query può condursi ricorrendo a risorse esogene (mediante calcoso di uno scarto standardizzato d'uso della parola, rispetto alla frequenza d'u-

so di riferimento in un lessico assunto come modello, ove queste ultime frequenze sono da assumersi come valori attesi) oppure ricorrendo a risorse endogene (mediante calcolo delle specificità) per selezionare il linguaggio specifico di una partizione (quello dei masthi rispetto alle femmine, o dei giovani/adulti/anziani ecc.). Quando invece si utilizza una query, il calcolo di un indice come il TFIDF permette di selezionare i termini più vicini alla richiesta, al fine di ordinare secondo un principio di rilevanza i documenti ripescati.
A livello di analisi di sequenze, l'estrazione di espressioni tipiche del corpus avviene, a partire dall'inventario dei segmenti ripetuti, grazie al calcolo di un indice IS che filtra i segmenti rilevanti secondo la loro capacità di assorbimento delle occorrenze delle parole componenti.

La fase di analisi testuale riguarda tutte le operazioni rivolte direttamente sul corpus, quindi in grado di fornire una rappresentazione sintagmatica del testo, sia puntualmente attraverso analisi di concordanze più o meno sofisticate a seconda del tipo di query, sia globalmente attraverso analisi di co-occorrenze. Queste utlime possono ricostruirsi sia direttamente dall'analisi statistica delle sequenze (predecessori/successori in un LAG predefinito) rispetto a parole pivot, sia indirettamente mediante ricostruzione di dimensioni semantiche latenti prodotte con tegniche di riduzione dimensionale di tipo: analsisi fattoriale delle corrispondenze, singular value decomposition, multidimensional scaling.
Ma l'analisi testuale, quando non si fa uso di tecniche statistiche multidimensionali, consente di:

1. rispondere a interrogazioni complesse sul corpus (analisi di concetti) estraendo i documenti più rilevanti che le verificano;

2. visualizzare le entità di interesse ricercate;

3. classificare i frammenti di testo creando nuove variabili testuali, che poi alimentano campi di un database strutturato.

2.1.4 Analisi lessicale e analisi testuale

Operazioni dello stesso tipo possono applicarsi sia in analisi lessicale alle unità di testo (parole o lessie) costituenti il vocabolario, sia in analisi tes-

tuale alle unità di contesto (documenti o frammenti del "discorso") costituenti il corpus come insieme totale delle occorrenze. La Tabella 1.1 illustra in parallelo queste "analogie" nei due tipi di analisi: dalle operazioni di base o di Text Mining alla ricerca di concordanze, dall'utilizzo di meta-informazioni alla estrazione d'informazione con risorse sia interne che esterne, dagli output primari frutto di suddette investigazioni agli output secondari utili per successive analisi statistiche multidimensionali.

Tipo di analisi	Analisi Lessicale	Analisi testuale
Livello di analisi	Paradigmatico (verticale)	Sintagmatico (orizzontale)
Ricerche su	Vocabolario	Corpus
Unità di analisi	Unità di testo: Parole à Lessie (ULT)	Unità di contesto: Frammenti / Record, documenti
Operazioni di base	classificazione grammaticale lemmatizzazione Fusioni delle classi di untià di testo, imprinting Ponderazione: dispersione, uso TFIDF	Etichette / annotazioni sulle singole occorrenze Individuazione di sequenze, di strutture di strutture Disambiguazioni Ponderazione: TFIDF
Text Mining	Query semplici Query predefinite (complesse), piani di lavoro (insiemi di query predefinite) Query per tipi/classi di unità di testo	Information Retrieval (recupero dei frammenti che verificano la query) Information Extraction (visualizzazione delle unità di testo oggetto della query nei frammenti selezionati)
Ricerca di concordanze	Semplici - IR sul vocabolario (per disambiguare le parole) Per tipi, classi o gruppi di unità di testo	Ricerche full text di parole o entità d'interesse (date,numeri, valute, misure, ...) Ricerche full text di entità note (nomi toponimi, società, ...)
Utilizzo di meta-informazioni	Classificazione delle ULT da tagging grammaticale / semantico	Classificazione dei frammenti da dizionari tematici e da regole
Estrazione di informazione con risorse interne	Parole rilevanti nel vocabolario da TFIDF parole caratteristiche in una partizione da analisi di specificità	Frammenti rilevanti di TFIDF (IR) rispetto all'intero vocabolario o a specifiche query (forme selezionate)
Risorse esterne	Linguaggio peculiare da lessici di frequenza Incidenza d'uso del vocabolario di base terminologia da dizionari tematici (positivo/negativo, cibo, ecc.)	Classificazione dei frammenti da dizionari o da regole con popolamento di campi di un DB tradizionali
Output primari	Indici / liste con ordinamento alfabetico, lessico metrico, inverso ...	Ricostruzione del corpus annotato con etichettatura grammaticale / semantica
Output secondari su matrici per analisi multidimensionali	Matrice forme x testi (da partizione del corpus in sottosistemi di frammenti secondo variabili categoriali)	Matrice frammenti x forme con filtri su singole sezioni del corpus (sub-corpus) o sulle forme (selezionate secondo un criterio predefinito) Matrice forme x forme di co-occorrenze semplici o pesate

Tabella 2.1: Sinottico sulle caratteristiche proprie dell analisi lessicale e dell analisi testuale

2.1.5 La produttività delle parole

Quando una "parola" è molto frequente in un corpus è altamente probabile che la sua produttività morfologica in quel testo sia elevata. Per produttività s'intende la capacità di generare una varietà di forme a partire dal suo lessema o radice (Figura1).

Nel grafo si illustra il caso del lessema "politic_" che, nel corpus "Rep90", produce una variet`à di 198 forme grafiche diverse per un totale di 344930 occorrenze. Di queste il 99,4% riguarda le quattro forme base (politica/o/i/he) e lo 0,6% le altre formazioni che, espresse in lemmi, si articolano in 128 prefissi (2530 occorrenze) e/o 28 suffissi (1869 occorrenze).

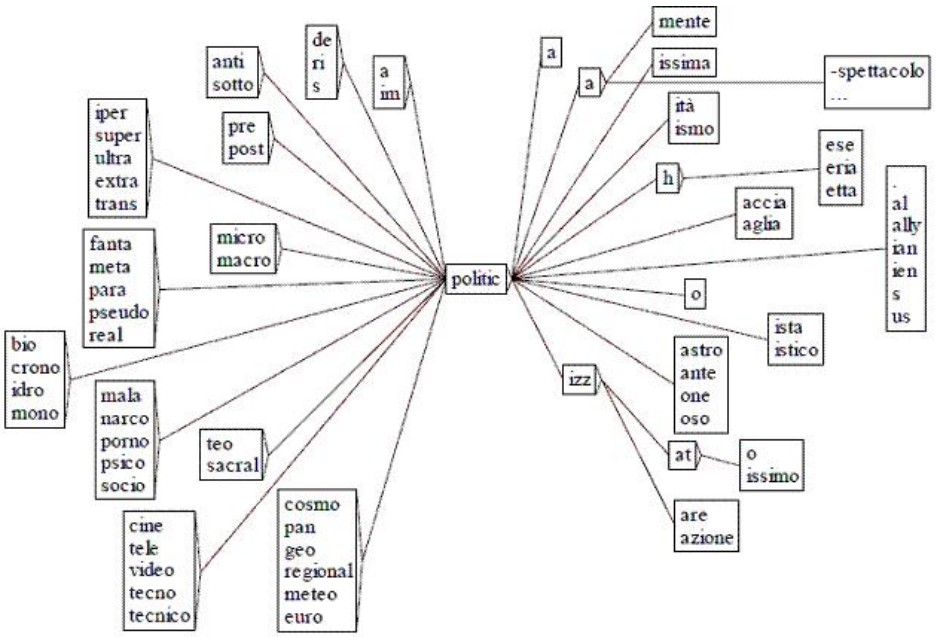

Figura 2.1: Grafo dei prefissi e dei suffissi della base "politic" in Rep90

2.2 Morfologia e dizionari

I primi programmi di elaborazione del linguaggio naturale, fino agli anni '80, usavano dizionari di migliaia di parole per effettuare il POS tagging in fase di parsing. I primi prototipi, tuttavia, erano poco potenti per scarsità di vocabolario.
Si studiarono poi i *morphology based look-up systems*, cioè sistemi basati su un insieme di dizionari di segmenti lessicali (basi, prefissi, suffissi, desinenze) e su un insieme di regole per la formazione delle parole.

I primi approcci erano troppo semplicistici perché sviluppati per l'inglese, che è una lingua poco flessa. L'approccio standard utilizzato attualmente viene dalla Finlandia ed è stato sviluppato da Koskenniemi. Sfrutta gli stessi principi elencati prima, ma include una parte fonemica particolarmente sofisticata e in grado di trattare aspetti morfo-fonemici complessi.

2.2.1 Definizioni utili

Vengono presentate alcune definizioni che risulteranno utili per affrontare lo studio sviluppato successivamente [11].

Morfema

Si tratta della più piccola unità significativa di un linguaggio. Una parola, o anche una sua parte (radice, suffisso, ecc).

Infisso

Parte aggiunta posta in mezzo ad una parola per alterarne il significato: **hingi** → h**um**i**ngi** (esempio dal filippino).

Circonfissi

Una parte precede la parola e una parte segue: **sag**en → ge**sag**t (esempio dal tedesco)

Inflessioni

Modificano parte della parola alterandone alcune caratteristiche. Possono essere **regolari** o **irregolari**.

- Plurale/singolare
- Genitivo sassone
- Terza persona dei verbi inglesi

Declinazioni

Sono trasformazioni di una parola sorgente per ottenere una nuova parola appartenente ad una diversa classe grammaticale, ottenute con l'aggiunta di prefissi, suffissi, ecc...

- nome → verbo;
- nome → avverbio;
- nome → aggettivo;
- aggettivo → nome;
- verbo → aggettivo;

Il linguaggio è molto flessibile. Avere le regole di flessione e declinazione ci permette di trattare parole nuove (es: fax) e immediatamente derivarne aggettivi, verbi, ecc. Ciò non sarebbe possibile se avessimo solo un lessico più grande comprendente tutte le forme possibili.

→

2.2.2 I corpora

Un corpus è un insieme di **testi riguardanti un certo argomento**, sufficientemente rappresentativi delle varie sottoaree di interesse dell'argomento stesso. La rappresentatività è uno degli indici della bontà di un corpus. La **dimensione** è un altro indice: se contiene fino a 1M parole, è di livello medio-basso. Fino a 100M è di buon livello. Intorno ai 500M di parole è di ottimo livello[41].
L'uso dei corpora risale alla linguistica computazionale e alla machine translation.
I corpora hanno il loro uso principale negli **approcci statistici**. Hanno portato alla realizzazione di metodi per l'inferenza delle feature linguistiche dai corpora tramite tecniche di apprendimento automatico (HMM, metodi statistici e reti neurali).
All'atto della generazione, è necessario affrontare i **problemi legati ai *raw* data**, come ad esempio la presenza della punteggiatura, degli spazi e degli errori: è necessario ripulire i dati tramite pre-processing prima di utilizzarli.

Si procede all'eliminazione di aspetti legati alla **formattazione** che non interessano. Si applicano euristiche per studiare la presenza di maiuscole e minuscole (sono titoli? inizi di frasi? sigle? l'identificazione è difficile!), gli apostrofi (elisione o genitivo sassone), i trattini (parole composte? divisione in sillabe? raggruppamento di sottoporzioni delle frasi?).
Si devono considerare anche **omografie** (una parola, più significati), **segmentazione** (più formati per una sola informazione: numeri di telefono, date), ***ditto tag*** (parole/espressioni sempre uguali a se stesse: "in spite of", "in order to", "because of").
I **testi** all'interno di un corpus **sono annotati** da esperti tramite linguaggi di markup e tagging (esistono vari standard per fare ciò). Si va dalla marcatura delle semplici strutture di base, fino alla marcatura sintattica completa (ad es, UPenn Treebank, banca di alberi sintattici della University of Pennsylvania). Di solito si usa un POS tagging utilizzando un tag set standard (Brown, C5, Upenn, ...), applicato a mano o con tecniche automatiche.
Lo **studio delle collocazioni e della frequenza** (anche solo dei bigrammi più frequenti) porta a raccogliere molte informazioni statistiche utili per la generazione del linguaggio. Ad esempio, seppur grammaticalmente corrette, "strong computer" e "powerful tea" non hanno significato, mentre lo hanno "strong tea" a "powerful computer". Tuttavia **le parole componenti una forma idiomatica possono comparire non adiacenti**. Si analizza quindi il corpus con finestrature di dimensione variabile e si calcolano media e varianza che caratterizzano la distribuzione della distanza tra le parole del corpus. Per evitare di individuare delle relazioni casuali presenti nel corpus, si può quindi utilizzare il test t di Student o il test χ^2 per decidere se è possibile rifiutare l'ipotesi nulla di esistenza di una reale forma idiomatica. Si possono inoltre utilizzare il rapporto di verosimiglianza delle ipotesi di dipendenza e indipendenza oppure i rapporti di frequenze relative di più termini per individuare la probabilità che i termini siano correlati. Un esempio di tali misure, è la **mutua informazione**:

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)\,P(y)} = \log_2 \frac{P(x|y)}{P(x)}$$

che è una buona misura dell'indipendenza, ma non della dipendenza, in quanto in caso di dipendenza il punteggio ottenuto dipende dalla frequenza delle parole componenti l'espressione.

Una delle prime applicazioni dell'individuazione di forme idiomatiche, è la possibilità di tradurle in lingue straniere.

2.2.3 La lemmatizzazione

La lemmatizzazione è "quel complesso di operazioni che conducono a riunire tutte le forme sotto il rispettivo lemma", intendendo per lemma "ciascuna parola-titolo o parola-chiave di un dizionario" e per forma ogni possibile diversa realizzazione grafica di un lemma. La lemmatizzazione, quindi, "-consiste nell'attribuire le varianti o flessive (uomini) o grafiche (omo) a una stessa parola (uomo), che funge da lemma" [10]: queste varianti sono le forme del lemma.
Sono esempi di forme mangerò, mangeremmo, mangiaste, mangi: il lemma cui ciascuna di queste forme è riducibile è mangiare (ovvero, la parola che appare come entrata sui dizionari), in quanto è convenzione italiana che il lemma verbale sia rappresentato dalla forma coniugata all'infinito presente attivo.
Da questo semplice esempio si illuminano già un paio di questioni:

- ci sono delle convenzioni di lmz proprie di ciascuna lingua: come visto, in italiano è uso convenzionale che il lemma verbale sia la forma coniugata all'infinito presente attivo. In latino, invece, il lemma verbale è coniugato alla prima persona singolare dell'indicativo presente attivo: si trova, infatti, sui dizionari l'entrata tollo e non tollere;

- il lemma è anche una forma: mangiare è, quindi, sia un lemma (che, da solo, rappresenta tutte le proprie possibili forme), sia una forma. Il contrario non vale: una forma non è necessariamente lemma. Infatti, mangiasse è una forma, ma non un lemma.

La lemmatizzazione è, dunque, una pratica apparentemente facile, se non, addirittura, ovvia ed intuitiva: più, o meno tutti, infatti, esercitiamo quotidianamente la conoscenza della differenza tra lemma e forma. Tuttavia, alla prova dei fatti, la lmz rivela una serie di problemi, il più delle volte non immaginabili prima di averne fatto esperienza diretta: queste difficoltà rendono il lemmatizzare un esercizio interessante, in quanto costringe chi lo esercita a riflettere su:

- quanti e quali automatismi l'uomo metta inconsciamente in atto ogni volta che parla, o scrive;

- quanto sia difficile formalizzare questi automatismi.

Esistono due tipi di lemmatizzazione:

- morfologica: analizza le forme di parole in isolamento, ovvero fuori dal contesto sintattico, fornendone tutti i valori che sono possibili in un dato sistema linguistico. Dal momento che la lemmatizzazione morfologica interessa le parole in sé, ovvero svincolate dalla sintassi, essa resta valida di ciascuna parola sempre e in qualsiasi contesto: ciò è estremamente necessario in informatica umanistica, in quanto è un dato che può essere assunto aprioristicamente, indipendentemente dal testo che, di volta in volta, viene preso in esame (si veda in proposito più sotto, dove tratto della lemmatizzazione semi-automatica);

- morfo-sintattica: analizza le forme di parole entro il contesto sintattico. Non è mai ambigua, ma sempre univoca, in quanto l'immersione della forma nella frase ne precisa il valore. Quindi, mentre la lemmatizzazione morfologica è indipendente dal testo, la lemmatizzazione sintattica è, invece, legata al testo su cui è applicata.

Capitolo 3

Stato dell'arte sugli algoritmi di clustering del testo

Il problema del clustering è stato studiato ampiamente nella letteratura dei database e statistiche nel contesto di un'ampia varietà di attività data mining [29, 8]. Il problema del clustering è definito in modo tale da trovare gruppi di oggetti simili nei dati.
I problemi del clustering possono essere molto utili nel dominio del testo, in cui gli oggetti da cluster possono essere di granularità diverse come documenti, paragrafi, frasi o termini. Il clustering è particolarmente utile per organizzare i documenti per migliorare il recupero e il supporto navigazione [3, 14]. Lo studio del problema del clustering precede la sua applicabilità al dominio del testo.
I metodi tradizionali di clustering sono generalmente concentrati sul caso di dati quantitativi [23, 39, 29, 8, 62], in cui gli attributi dei dati sono di tipo numerico. Il problema è stato studiato anche per il caso di dati categorico [2, 21, 24], in cui gli attributi possono assumere valori nominali. Un'ampia panoramica del clustering (che si riferisce a dati generici numerici e categorici) si possono trovare in [29, 8].
Un certo numero di implementazioni degli algoritmi di clustering, come applicato ai dati di testo, possono essere trovati in molti toolkit come Lemur [28] e PRUA toolkit in [35]. Il problema del clustering trova applicabilità per una serie di attività:

- Organizzazione dei documenti per la navigazione: organizzazione gerarchica dei documenti in un insieme coerente le categorie può essere

molto utili per l'esplorazione sistematica della raccolta dei documenti. Un classico esempio di questo è il motodo Scatter/Gather [13], il quale fornisce una sistematica esplorazione tecnica con l'uso di cluster per l'organizzazione della raccolta dei documenti.

- Corpus Summarization: tecniche di Clustering forniscono una sintesi coerente della raccolta in forma di cluster-digest [50] o word-cluster [5, 6], che può essere utilizzato allo scopo di fornire informazioni in sintesi del contenuto globale del corpus sottostante. Varianti di tali metodi, in particolare clustering di frasi, può anche essere utilizzato per il riepilogo. Il problema del clustering è anche strettamente correlato a quello della riduzione di dimensionamento e modellazione argomento.

- Classificazione dei documenti: mentre il clustering è di per sé un metodo di apprendimento senza supervisione, possono essere utilizzate per migliorare la qualità dei risultati nella sua variante controllata. In particolare, word-cluster [5, 6] e i metodi di formazione [40] possono essere usati per migliorare l' accuratezza di applicazioni di classificazioni supervisionate con l'uso di tecniche di clustering.

Si nota che molte classi di algoritmi come gli algoritmi gerarchici sono general-purpose, possono essere estesi a qualsiasi tipo di dati, inclusi i dati di testo. Un documento di testo può essere rappresentato sia sotto forma di dati binari, quando usiamo la presenza o l'assenza di una parola nel documento al fine di creare un vettore binario . In tali casi, è possibile utilizzare una varietà di algoritmi di clustering di dati categoriali [2, 21, 24] sulla rappresentazione binaria.
Una rappresentazione più raffinata che include metodi di pesatura basati sulle frequenze delle singole parole nel documento , nonché le frequenze di parole in una intera raccolta (ad es. ponderazione TF-IDF [49]). I dati quantitativi gli algoritmi di clustering [23, 39, 62] possono essere utilizzati in combinazione con tali frequenze al fine di determinare i più importanti gruppi di oggetti nei dati.
Tuttavia, tali tecniche ingenue generalmente non funzionano bene per il clustering dei dati di testo. Questo perché i dati di testo hanno un numero di caratteristiche uniche che richiedono la progettazione di algoritmi specifici per l'attività . Le caratteristiche che contraddistinguono la rappresentazione del testo sono i seguenti :

- La bidimensionalità della rappresentazione del testo è molto grande, ma i dati sottostanti sono scarsi. In altre parole, il lessico di elaborazione dei documenti in parole può essere dell'ordine di 10^1, ma un dato documento può contenere solo poche centinaia parole. Il problema è ancora più grave quando i documenti che vengono raggruppati sono molto brevi (ad esempio il clustering frasi o tweet).

- Mentre il lessico di un corpus di documenti pu ò essere grande, le parole sono in genere correlate tra loro. Ciò significa che il numero di concetti (o componenti principali) dei dati è molto minore della funzione nello spazio. Ciò richiede un attenta progettazione di algoritmi che possono rappresentare le correlazioni di parole nel processo di clustering.

- Il numero di parole (o non-zero voci) nei vari documenti può variare ampiamente. Pertanto, è importante per normalizzare il documento rappresentarle opportunamente durante le attività di clustering .

La rappresentazione sparsa e tridimensionale dei vari documenti richiede la progettazione di specifici algoritmi per la rappresentazione ed elaborazione, un argomento molto studiato per il recupero di informazioni nella letteratura dove molte tecniche sono state proposte per ottimizzare la rappresentazione per migliorare la precisione di una query [49, 4]. La maggior parte di queste tecniche può essere utilizzato anche per migliorare la rappresentazione per il clustering.

Al fine di consentire un efficace processo di clustering, le frequenze delle parole devono essere normalizzate per quanto riguarda la loro frequenza relativa di presenza e per l'intera collezione. In generale, una rappresentazione comune utilizzata per l'elaborazione del testo è il vettore-basato sullo spazio TF-IDF [48]. Nella rappresentazione TF-IDF, il termine della frequenza di ogni parola è normalizzata per la frequenza inversa o IDF.
La frequenza inversa riduce il peso dei termini che ricorrono più frequentemente nella raccolta. In questo modo si riduce l'importanza dei termini più comuni nella raccolta, assicurando che l'abbinamento di documenti sia più influenzato da quello dei termini più discriminatori che sono relativamente poco frequenti della raccolta.

[1] 5

In aggiunta, una sotto-funzione di trasformazione lineare è applicata spesso al fine di evitare l'effetto indesiderato del dominante di un qualsiasi termine che potrebbe essere molto frequente in un documento. Il lavoro di normalizzazione è di per sé un vasto settore della ricerca, in una vasta gamma di queste tecniche può essere trovata in [51, 49].
Gli algoritmi di clustering sono suddivisi in un'ampia varietà di tipi diversi, come, ad esempio algoritmi di clustering agglomerativo, algoritmi di partizionamento e standard parametrici basati su metodi come l' algoritmo EM. Inoltre, le rappresentanze possono anche essere trattate come stringhe (piuttosto che sacchetti di parole). Tali rappresentazioni diverse richiedono la progettazione di differenti classi di algoritmi di clustering.

3.1 Selezione delle funzioni e metodi di trasformazione per il testo

Il Clustering di qualsiasi data mining come la classificazione dipende fortemente dalla rumorosità delle caratteristiche che vengono utilizzate per il processo di elaborazione. Per esempio, comunemente parole come 'il', non possono essere molto utili per migliorare la qualità del clustering. Pertanto, è fondamentale selezionare le funzioni efficacemente, in modo che la rumorosità delle parole nel corpus siano rimossi prima del clustering.
Oltre alla selezione della funzione, un certo numero di metodi di trasformazione come Latent Semantic Indexing (LSI), Probabilistic Latent Semantic Analysis (PLSA), e Non-negative Matrix Factorization (NMF) sono disponibili per migliorare la qualità della rappresentazione del documento e renderlo più suscettibile di clustering.
In queste tecniche (spesso chiamato riduzione della dimensione), le correlazioni tra le parole del lessico vengono utilizzate al fine di creare le funzioni, che corrispondono ai concetti o componenti principali nei dati.

3.1.1 Metodi di selezione

La funzione di selezione funzionalità è più comune e facile da applicare al problema della classificazione testo [59] in cui supervisione è disponibile per il processo di selezione delle caratteristiche . Tuttavia, un certo numero

di semplici senza supervisione metodi possono anche essere utilizzati per selezionare le funzioni nel testo il clustering. Alcuni esempi di tali metodi sono descritti di seguito.

Scelta del documento basato sulla frequenza

Il metodo più semplice possibile per la funzione selezione nel documento del clustering è l'uso della frequenza per filtrare caratteristiche irrilevanti. Mentre l'uso della frequenza inversa riduce l'importanza di tali parole, questo non può da sola essere sufficiente a ridurre il rumore di parole molto frequenti.
In altre parole, parole che sono troppo frequenti nel corpus possono essere rimosse, perché sono tipicamente parole comuni come 'a', 'un', 'i', o 'di' che non sono discriminatorie in un clustering. Una varietà di metodi sono comunemente disponibili in letteratura [54].
Inoltre, parole che si verificano molto raramente possono anche essere rimosse dalla raccolta. Questo perché tali parole non aggiungono nulla alla somiglianza calcoli che vengono utilizzati nella maggior parte dei metodi di clustering. In alcuni casi , tali parole possono essere errori ortografici o errori tipografici nei documenti. Collezioni di testi rumorosi sono derivate dal web, blog o social network sono più suscettibili a contenere tali termini.
Si nota che alcune linee di ricerca definire documento basato sulla frequenza scelta esclusivamente sulla base di rado, perché questi termini meno contribuiscono alla somiglianza dei calcoli. Tuttavia, va sottolineato che sono molto frequenti le parole dovrebbero anche essere rimosso, soprattutto se non sono discriminatorie tra i cluster. Nota che il metodo TF-IDF può anche filtrare le parole comuni di un metodo 'soft'.

Forza del termine

Una tecnica molto più aggressiva è proposta in [56]. L'idea alla base di questo approccio è quello di estendere le tecniche che vengono utilizzate per l'apprendimento con la supervisione del supervisore del caso. Il termine è essenzialmente utilizzato per misurare l' informativa di una parola per l'individuazione di due documenti correlati. Ad esempio, per due documenti correlati x e y, il termine forza $s(t)$ di durata t è definito nei seguenti termini:

$$s\left(t\right) = P\left(t \in y | \, t \in x\right)$$

Chiaramente, il problema principale è come si potrebbe definire il documento x e y . Una possibilità è quella di un manuale d'uso (o utente) feedback per definire quando una coppia di documenti è correlata. Questo è essenzialmente equivalente all'utilizzo del controllo nel processo di selezione delle caratteristiche, e può essere pratica in situazioni in cui le categorie di documenti sono disponibili. D'altro canto, non è possibile creare manualmente le coppie in grandi raccolte in modo completo.
È stato dimostrato in [56] che è possibile utilizzare le funzioni automatizzate somiglianza come la funzione coseno [48] per definire la parentela di coppie di documenti. Una coppia di documenti è definita correlata se loro somiglianza coseno è al di sopra di una soglia definita dall'utente.
In tali casi, il termine forza $s(t)$ può essere definita in modo casuale campionamento di un numero di coppie di tali documenti come segue:

$$s\left(t\right) = \frac{Num_{pair \in t}}{Num_{pair \in t_1}}$$

Qui, il primo documento della coppia può essere semplicemente scelto a caso. Al fine di snellire le funzioni, il termine può essere confrontato con un termine che sia distribuito in modo casuale per i documenti di training con la stessa frequenza. Se il termine t non è almeno due deviazioni standard superiore a quello della parola casuale, viene rimosso dalla raccolta.
Uno dei vantaggi di questo approccio è che non richiede la supervisione iniziale o dati di addestramento per la selezione della funzione, che è un requisito fondamentale per la supervisione dello scenario. Naturalmente l'approccio può essere utilizzato anche per selezionare le funzioni di clustering supervisionato [11] o la classificazione [60], quando tale formazione dati è infatti disponibile.
Un'osservazione su questo tipo di approccio per la selezione di funzione è che risulta particolarmente adatto per somiglianza clustering basato sul perché la natura discriminativa del sottostante è definito sulla base di somiglianze nei documenti stessi.

Classificazione su Entropia

La classificazione su entropia è stato un approccio proposto in [15]. In questo caso, la qualità di questo termine è misurata mediante dalla riduzione dell'entropia quando viene rimossa. Qui l'entropia $E(t)$ del termine t in una

raccolta di documenti è definito come segue

$$E(t) = -\sum_{i=1}^{n}\sum_{j=1}^{n}\left(S_{ij}\cdot\log\left(S_{ij}\right)+\left(1-S_{ij}\right)\cdot\log\left(1-S_{ij}\right)\right)$$

Qui $S_{ij} \in (0,1)$Sij è la somiglianza tra l'i-esimo e j-esimo documento nella raccolta, dopo il termine t viene rimosso e viene definito come segue:

$$S_{ij} = 2^{-\frac{dist(i,j)}{\overline{dist}}}$$

Qui $dist\,(i,j)$ è la distanza tra i termini i e j dopo che il termine t è stato rimosso e $dist$ è la distanza media tra i documenti dopo il termine t è stato rimosso. Si nota che il calcolo di $E(t)$ per ogni termine t richiede $O\,(n^2)$ operazioni. Questo non è pratico di un corpus molto vasto contenente molti termini. È stato dimostrato in [15] come questo metodo può essere molto più efficace con l'uso di metodi di campionamento .

Contributo del termine

Questo concetto [33] si basa sul fatto che i risultati del clustering sul testo sono altamente dipendenti dalla somiglianza del documento. Pertanto, il contributo di un termine può essere considerato come il suo contributo di somiglianza al documento. Per esempio, nel caso di prodotto scalare in base somiglianza, la somiglianza tra i due documenti è definito come il prodotto scalare di frequenze normalizzate.
Pertanto, il contributo di un termine della somiglianza dei due documenti è il prodotto delle loro frequenze normalizzate in due documenti. Questa deve essere la somma di tutte le coppie di documenti al fine di determinare il termine contributo. Come nel caso precedente, questo metodo richiede $O\,(n^2)$ per ogni termine, e quindi i metodi di campionamento può essere necessaria per accelerare il contributo.
Una delle principali critiche mosse a questo metodo è che tende a favore molto frequenti parole senza riguardo alla specifica discriminativa potenza all'interno di un processo di clustering. Nella maggior parte di questi metodi, l'ottimizzazione del termine selezione si basa su alcune pre-suppone somiglianza funzione (ad es., coseno). Questa strategia rende questi metodi senza supervisione, vi è la preoccupazione che il termine selezione potrebbero essere spinti a causa del potenziale di polarizzazione dell'assunta somiglianza .

Quindi, la scelta di un appropriato somiglianza può essere importante per questi metodi.

3.1.2 Metodi basati su LSI

In selezione della funzione, si tenta di trovare in modo esplicito le caratteristiche del set di dati originale. La funzione trasformazione è un metodo diverso in cui le nuove funzioni sono definite come una rappresentazione funzionale delle funzioni del set di dati originale.
La più comune classe di metodi di riduzione di dimensionamento [31] in cui i documenti vengono trasformati in un nuovo spazio di funzioni di dimensionalità inferiore in cui le funzioni sono tipicamente una combinazione lineare delle funzioni di dati originali. Metodi come indexing semantico latente (LSI) [16] sono basati su questo principio comune. L'effetto complessivo è quello di rimuovere un sacco di dimensioni nei dati che sono rumorose per applicazioni basate su somiglianza come il clustering.
La rimozione di tali dimensioni aiuta anche ingrandire la semantica in effetti dei dati sottostanti. Da LSI è strettamente correlata al problema dell'analisi dei Componenti Principali (PCA) o decomposizione in valori singolari (SVD), dovremo prima discutere di questo metodo, e la sua relazione con LSI. Di una d-set di dati dimensionali, PCA costruisce il simmetrico $d \times d$ matrice di covarianza C dei dati, in cui il (i, j) è la covarianza tra dimensioni i e j.
Questa matrice semidefinita positiva, e può essere diagonalizzata come segue:

$$C = P \cdot D \cdot P^T$$

In questo caso P è una matrice le cui colonne contengono gli autovettori ortonormali di C e D è una matrice diagonale contenente i corrispondenti autovalori. Notiamo che gli autovettori ortonormali rappresentano un nuovo sistema di base lungo il quale i dati possono essere rappresentati. In questo contesto, gli autovalori corrispondono alla varianza quando i dati vengono proiettati lungo questo sistema di base.
Questo sistema di base è anche quello in cui il secondo ordine delle covarianze dei dati vengono rimosse, e la maggior parte della varianze dei dati viene catturato per preservare gli autovettori con autovalori più ampi. Pertanto, al fine di ridurre la dimensionalità dei dati, un approccio comune è quello di rappresentare i dati in questo nuovo sistema di base, che viene

ulteriormente troncato ignorando quegli autovettori per i quali i corrispondenti autovalori sono piccoli.
Questo perché le varianze lungo quelle dimensioni sono piccole, e il comportamento dei punti dati non è significativamente influenzata dall'esame. In effetti, può essere dimostrato che la distanza euclidea tra i punti non sono significativamente interessati da questa trasformazione e corrispondente troncamento. Il metodo del PCA è comunemente utilizzato per ricerca di somiglianza nel recupero delle applicazioni database.
LSI è molto simile alla PCA, ad eccezione del fatto che utilizziamo una approssimazione della matrice di covarianza C, che si adatta molto bene per la natura pluridimensionale dei dati di testo.
In collezioni, soltanto 300-400 autovettori sono necessari per la rappresentazione. Una ottima caratteristica di LSI [16] è che il troncamento di dimensioni rimuove gli effetti dei disturbi di sinonimia e polysemia, e la somiglianza calcoli sono maggiormente colpiti dalla semantica dei concetti nei dati. Questo è particolarmente utile per un applicazione semantica come testo il clustering.
Tuttavia, se una granularità del clustering, come spazio di rappresentazione di piccole dimensioni del testo non può essere sufficientemente discriminativa; nel recupero delle informazioni, il problema è spesso risolto miscelandolo con l'originale rappresentazione tridimensionale (vedi, ad esempio, [61]). Una tecnica simile per LSI, ma basata sulla modellazione probabilistica è Probabilistic Latent Semantic Analysis (PLSA) [27]. La somiglianza e l'equivalenza delle PLSA e LSI sono discusse in [27].

Nozioni di decomposizione utilizzando il Clustering

Una interessante osservazione è che, mentre funzione trasformazione viene spesso utilizzato come pre-elaborazione tecnica per il clustering, il raggruppamento stesso può essere utilizzato per una nuova riduzione di dimensionamento tecnica nota come concetto decomposizione [22, 17]. Questo naturalmente porta al problema della circolarità nell'uso di questa tecnica per il clustering, specialmente se il clustering è richiesto per eseguire la riduzione di dimensionamento.
Tuttavia, è possibile utilizzare questa tecnica efficace per la pre-elaborazione con l'uso di due fasi distinte di clustering. La tecnica di decomposizione standard utilizza una tecnica di clustering [22, 17] sulla rappresentazione

originale dei documenti. I frequenti termini i centroidi di questi gruppi sono utilizzati come vettori di base che sono quasi ortogonali tra loro. I documenti possono quindi essere rappresentati in un modo molto più breve in termini di questi vettori di base.
Notiamo che questo condensato rappresentazione concettuale permette di cluster avanzati nonché classificazione. Pertanto, una seconda fase di raggruppamento può essere applicata su questa rappresentazione ridotta al fine di raggruppare i documenti molto più efficace. Tale metodo è stato testato anche in [52] utilizzando word-cluster in modo da rappresentare i documenti. Descriveremo questo metodo in dettaglio pi¨u avanti in questo capitolo.

3.1.3 Fattorizzazione di matrici non negative

La fattorizzazione di matrici non negative (NMF) è particolarmente adatta per il clustering [58]. Come nel caso di LSI, lo schema NMF rappresenta i documenti in un nuovo sistema basato su un'analisi della matrice termine-documento. Tuttavia, questo metodo presenta un certo numero di differenze critiche da LSI da un punto di vista concettuale.
In particolare, NMF è un metodo di trasformazione che è particolarmente adatto per il clustering. Le principali caratteristiche concettuali del NMF, che sono molto diverse da LSI sono le seguenti:

- In LSI, il nuovo sistema di base consiste in un insieme di vettori ortonormali. Questo non è il caso di NMF.

- In NMF, i vettori nel sistema di base corrispondono direttamente al cluster argomenti. Pertanto, il cluster membership per un documento può essere determinato esaminando la componente principale del documento. Le coordinate di un qualsiasi documento lungo un vettore sono sempre non negative. L'espressione di ogni documento come una combinazione additiva della sottostante semantica ha senso intuitivo. Pertanto, la trasformazione NMF è particolarmente adatta per il clustering, e fornisce anche una comprensione intuitiva del sistema di base in termini di cluster.

3.2 Algoritmi per il clustering basati sulla distanza

Gli algoritmi per il clustering basati sulla distanza sono progettati utilizzando una funzione per misurare la vicinanza tra gli oggetti di testo. La più nota che viene comunemente usata nel testo dominio è la somiglinanza coseno. Sia $U = (f(u_1), \ldots, f(u_k))$ e $V = (f(v_1), \ldots, f(v_k))$ siano i vettori ammortizzati di frequenze normalizzate di due diversi documenti U e V .
I valori $u_1, \ldots, u_k$ e $v_1, \ldots, v_k$ rappresentano la frequenza (normalizzato) dei termini e la funzione $f(\cdot)$ rappresenta la funzione di ammortizzazione. Tipiche funzioni di smorzamento per $f(\cdot)$ potrebbe essere la radice quadrata o il logaritmo [13]. Quindi, la somiglianza coseno tra i due documenti è definito come segue:

$$\cos ine(U, V) = \frac{\sum_{i=1}^{k} f(u_i) \cdot f(v_i)}{\sqrt{\sum_{i=1}^{k} f(u_i)^2} \cdot \sqrt{\sum_{i=1}^{k} f(v_i)^2}}$$

Il calcolo della similarità del testo è un problema fondamentale nel recupero delle informazioni. Sebbene la maggior parte del lavoro di recupero delle informazioni si è concentrata su come valutare la similarità di una parola chiave, piuttosto che la somiglianza tra i due documenti, molte funzioni di ponderazione euristiche e somiglianza possono anche essere applicate ad ottimizzare la somiglianza per il clustering.
Modelli efficaci euristici per il recupero di informazioni generalmente sono tre, cioè ponderazione TF, ponderazione IDF e normalizzazione della lunghezza del documento [18]. Un modo efficace per assegnare i pesi ai termini quando rappresentano un documento come un vettore di termini pesati è il metodo di ponderazione BM25 [45].
Un documento può anche essere rappresentato con una distribuzione di probabilità sulle parole (cioè modelli di linguaggio unigram), e la somiglianza può essere misurata sulla base di informazioni teorico misura come entropia o la divergenza Kullback-Leibler [61]. Per il clustering, varianti simmetriche di tale funzione di somiglianza funzione potrebbero essere più appropriate. Una sfida per il raggruppamento di brevi segmenti di testo (ad

es., tweet o frasi) è quella esatta corrispondenza delle parole chiave che non può funzionare bene.
Una strategia generale per risolvere questo problema è di espandere la rappresentazione del testo sfruttando relativi documenti di testo, che sono legati alla levigatura del modello di una lingua per l'information retrieval [61]. Una specifica tecnica, che sfrutta un motore di ricerca per espandere la rappresentazione del testo, è stato proposto in [46]. Un confronto tra alcune semplici misure per somiglianza di brevi segmenti di testo può essere trovata in [36].
Tali funzioni di similarità possono essere usato in combinazione con una grande varietà di tradizionali algoritmi di clustering [29, 8].

3.2.1 Algoritmi agglomerativi e gerarchici di clustering

Algoritmi gerarchici sono stati studiati in modo approfondito in letteratura [29, 8] per i record di diverso tipo, tra cui i dati numerici multidimensionali e categorici e i dati di testo. Una panoramica delle tecniche tradizionali è fornito in [37, 38, 55, 57]. Un confronto sperimentale di diversi algoritmi di clustering gerarchico può essere trovata in [63].
Il metodo di clustering gerarchico agglomerativo è particolarmente utile per sostenere una varietà di metodi di ricerca, perché naturalmente crea una struttura ad albero gerarchico che può essere utilizzata per il processo di ricerca. In particolare, l'efficacia di questo metodo per migliorare l'efficienza della ricerca in una scansione sequenziale è stato mostrato in [30, 44]. Il concetto generale di clustering agglomerativo è unire successivamente documenti in cluster in base alla loro somiglianza con un altro.
Quasi tutti gli algoritmi di clustering gerarchico uniscono successivamente i gruppi sulla base del miglior somiglianza tra i gruppi di documenti. Le principali differenze tra le classi di metodi sono in termini di come questa similitudine viene calcolata tra i diversi gruppi di documenti. Per esempio, la somiglianza tra una coppia di gruppi può essere calcolata come il caso migliore somiglianza, somiglianza media, o peggiore somiglianza tra i documenti che provengono da queste coppie di gruppi.
Concettualmente, il processo di agglomerazione dei documenti in livelli in successione superiori di cluster crea un cluster (gerarchia o dendogramma per le quali i nodi corrispondono ai singoli documenti, e i nodi interni cor-

rispondono alla fusione dei gruppi di cluster. Quando due gruppi sono uniti, un nuovo nodo è creato in questa struttura, corrispondente a questa entità maggiore risultante dalla fusione. I due figli di questo nodo corrispondono ai due gruppi di documenti che sono stati fusi.
I diversi metodi di fusione dei gruppi di documenti per i diversi metodi agglomerativo sono come segue:

- Single Linkage Clustering: la somiglianza tra due gruppi di documenti è la piùù grande somiglianza tra qualsiasi coppia di documenti di questi due gruppi. In un unico link si uniscono i due gruppi che sono tali che la loro più alta coppia di documenti che hanno la massima somiglianza rispetto a qualsiasi altra coppia di gruppi. Il vantaggio principale del single linkage clustering è che è estremamente efficace da mettere in pratica. Questo è il motivo per cui può calcolare tutte somiglianza coppie e li ordina al fine di ridurre la somiglianza. Queste coppie sono state elaborate in questo pre-ordine definito e il merge viene eseguito successivamente se le coppie sono appartenenti a gruppi differenti. Questo è sostanzialmente equivalente ad un algoritmo spanning tree sul grafico completo di coppie di distanze. È stato dimostrato in [55] come l'algoritmo Prim dello spanning tree può essere adattato al singolo collegamento del clustering. Un altro metodo [12] progetta l'unico metodo di legame con l'indice inverso per evitare compuazioni di zero somiglianze. Il principale svantaggio di questo approccio è che esso può portare al fenomeno della concatenazione in cui una catena di documenti simili portano a documenti differenti raggruppati in uno stesso cluster. In altre parole, se A è simile a B e B è simile a C, non sempre significa che A è simile a C, a causa della mancanza di transitività nei calcoli. Single linkage clustering incoraggia il raggruppamento dei documenti attraverso la transitività di catene. Spesso questo può portare a uno scarso numero di cluster, specie ai livelli più elevati dell'agglomerazione. Metodi efficaci per implementare un collegamento per il clustering di dati del documento può essere trovata in [12, 55].

- Group-Average Linkage Clustering: la somiglianza tra due cluster è la media somiglianza tra le coppie di documenti in due raggruppamenti. È chiaro che il processo di clustering medio è leggermente più lento rispetto a un singolo collegamento clustering, perché abbiamo bisogno

di determinare la media tra un grande numero di coppie al fine di determinare la somiglianza media. D'altro canto, è molto più efficace in termini di qualità del clustering, perché non presentano il comportamento del concatenamento come single linkage clustering. È possibile accelerare l'algoritmo di clustering, assimilando la somiglianza media tra due cluster C_1 e C_2 calcolando la somiglianza tra la media di C_1 e la media di C_2. Mentre questo approccio non funziona ugualmente bene per tutti i domini dei dati, funziona particolarmente bene per il caso dei dati di testo. In questo caso, il tempo pu`ò essere ridotto a $O(n^2)$, dove n è il numero totale di nodi. Il metodo può essere implementato in modo abbastanza efficace nel caso di dati del documento, perché il baricentro di un cluster è semplicemente la concatenazione dei documenti in cluster.

- Complete Linkage Clustering: In questa tecnica, la somiglianza tra due cluster è la peggiore somiglianza tra qualsiasi coppia di documenti in due raggruppamenti. Completare il clustering si può anche evitare il concatenamento perché evita l'inserimento di una qualsiasi coppia di punti molto diversi nello stesso cluster. Tuttavia, come clustering, è computazionalmente più costoso di un singolo metodo di legame. Il metodo di clustering completo richiede $O(n^2)$ e $O(n^3)$. Il requisito di spazio può tuttavia essere significativamente inferiore nel caso dei dati nel dominio di testo, perché un gran numero di coppie di somiglianze sono pari a zero.

Algoritmi di clustering gerarchico sono stati progettati nell'ambito del testo flussi di dati. Un metodo di modellazione distributivo di clusterizzazione gerarchica di streaming è stato proposto in [47]. L'idea principale è quella di un modello della frequenza di presenza in documenti con l'uso di un multi-distribuzione di Poisson. I parametri di questo modello sono appresi al fine di assegnare i documenti in cluster. Il metodo estende gli algoritmi COBWEB e CLASSIT [19, 20] al caso di dati di testo. Il lavoro in [47] studi i vari tipi di ipotesi distribuzionali di parole nei documenti. Notiamo che queste ipotesi distribuzionali sono tenuti ad adattare questi algoritmi per il caso dei dati di testo. L'approccio essenzialmente modifiche distribuzionali assunzione in modo che il metodo può essere utilizzato efficacemente per i dati di testo.

3.2.2 Algoritmi di partizionamento basati sulla distanza

Algoritmi di partizionamento sono ampiamente utilizzati nel database letteratura al fine di creare cluster di oggetti. I due più diffusi distanza partizionamento basato su algoritmi [29, 8] sono i seguenti:

- algoritmi di clustering K-medoid: negli algoritmi di clustering K-medoid, utilizziamo una serie di punti da i dati originali come le ancore (o medoids) intorno alla quale i cluster sono costruiti. L'obiettivo chiave dell'algoritmo è di determinare un insieme ottimale di rappresentanti dei documenti del corpus originale su cui i grappoli vengono costruiti. Ogni documento viene assegnato al suo più vicino rappresentante della collezione. In questo modo si crea una serie di cluster dal corpus, che vengono successivamente perfezionati da un processo casuale. L'algoritmo lavora con un approccio iterativo in cui il set di k rappresentanti sono successivamente migliorati con l'uso di studio randomizzato di inter-modifiche. In particolare, la somiglianza media di ogni documento nel corpus del sua più vicino rappresentante come funzione obiettivo che deve essere migliorata durante questo scambio. In ogni iterazione, si sostituisce un rappresentante selezionate casualmente nell'attuale serie di medoids selezionati casualmente, se migliora la funzione obiettivo clustering . Questo approccio è applicato fino a realizzare la convergenza . Ci sono due principali svantaggi dell'uso di di algoritmi basati sul k-medoids, di cui uno specifico caso dei dati di testo. Uno svantaggio generale di k-medoids è che necessitano di un elevato numero di iterazioni per raggiungere una convergenza e sono quindi risulta piuttosto lento. Questo perché ogni iterazione richiede il calcolo di una funzione obiettivo il cui requisito di tempo è proporzionale alla dimensione del corpus sottostante. Il secondo svantaggio è che gli algoritmi k-medoid non funzionano molto bene per dati insufficienti come il testo. Ciò è dovuto al fatto che una grande frazione di coppie di documento coppie non hanno molte parole in comune, e le somiglianze tra queste coppie di documenti hanno valori piccoli (e rumoroso). Pertanto, un unico documento spesso non contiene tutti i concetti necessari per poter realizzare un cluster intorno ad esso. Questa caratteristica è specifica per il caso del recupero delle informazioni sul dominio, a causa della natura sparsa dei dati di testo.

- algoritmi di clustering K-means: anche il clustering k-means utilizza una serie di k rappresentanti attorno a cui i grappoli vengono costruiti. Tuttavia, questi rappresentanti non sono necessariamente ottenuti dai dati originali e raffinati in modo differente rispetto a k-medoids. La forma più semplice di approccio k-means è di iniziare con un set di k i semi del corpus originale, e assegnare i documenti di questi semi sulla base della più vicina somiglianza. Nella successiva iterazione, il baricentro di punti assegnati a ciascun seme viene utilizzato per sostituire il seme nell'ultima iterazione. In altre parole , il nuovo seme è definito, in modo che sia un migliore punto centrale di questo quadro. Questo approccio è continuato fino alla convergenza. Uno dei vantaggi del metodo k-means rispetto a k-medoids è che richiede un numero estremamente limitato di iterazioni per convergere. Osservazioni da [13, 50] sembrano suggerire che per molti grandi insiemi di dati, è sufficiente utilizzare 5 o meno iterazioni per un efficace clustering. Lo svantaggio principale della k-mean è che è ancora piuttosto sensibile all'impostazione iniziale di semi raccolti durante il clustering. In secondo luogo, il baricentro per un determinato cluster di documenti può contenere un gran numero di parole. Questo rallenterà la somiglianza dei calcoli nella iterazione successiva.

3.2.3 Un Approccio ibrido: Il metodo Scatter-Gather

Mentre i metodi di clustering gerarchico tendono ad essere più robusti a causa della loro tendenza a confrontare tutte le coppie di documenti, non sono generalmente molto efficienti, a causa della loro tendenza a richiedere almeno un tempo pari a $O(n^2)$. D'altro canto, gli algoritmi k-means sono più efficaci degli algoritmi gerarchici, ma a volte possono essere poco efficaci a causa della loro tendenza a fare affidamento su un piccolo numero di semi. Questo robusto insieme di semi viene utilizzato in combinazione con un normale k-means clustering algoritmo per determinare un buon clustering. La dimensione del campione nella fase iniziale è attentamente calibrata in modo da garantire la migliore efficacia possibile, senza questa fase diventa un collo di bottiglia per l'esecuzione dell'algoritmo.
Vi sono due metodi per la creazione della prima serie di semi, che sono indicati come buckshot e fractionization rispettivamente. Questi sono due metodi alternativi, e sono descritti come segue:

- Buckshot: Sia k il numero di cluster e n il numero di documenti del corpus. Invece di prelevare k semi casualmente dalla raccolta, il regime buckshot prende $\sqrt{k \cdot n}$ di semi, quindi li agglomera k semi. Gli algoritmi clustering gerarchico agglomerativo standard (che richiedono tempo quadratico) vengono applicati a questo campione iniziale di $\sqrt{k \cdot n}$ semi. Dato che usiamo gli algoritmi quadraticamente scalabili in questa fase, questo approccio richiede $O(k \cdot n)$ tempo. Notiamo che questo seme è molto piuàffidabile rispetto ad un altro che semplicemente campiona per k semi, a causa del riepilogo di un grande campionamento di documenti in una robusta serie di k semi.

- Frazionamento: l'algoritmo di frazionamento inizialmente rompe il corpus in n/k buckets di dimensione $m > k$ ciascuno. Un algoritmo agglomerativo viene applicato a ciascuna di questi buckets per ridurre di un fattore di v. Così, al termine della fase, abbiamo un totale di k punti agglomerati. Il processo viene ripetuto da trattare ciascuno di questi punti agglomerati come un singolo record. Ciò si ottiene mediante la fusione dei diversi documenti all'interno di un cluster agglomerato in un unico documento. L'approccio termina quando un totale di k semi rimangono. Notiamo che il clustering agglomerativo per ciascun gruppo di m documenti alla prima iterazione del frazionamento algoritmo richiede $O(m^2)$ di tempo, con somme di $O(n \cdot m)$ oltre gli n/m gruppi diversi. Poiché il numero di individui riduce geometricamente di un fattore di v in ogni iterazione, il tempo di funzionamento totale su tutte le iterazioni è $O(n \cdot (1 + n2, +...))$. Per la costante $v < 1$, il tempo di esecuzione di tutte le iterazioni è ancora $O(n \cdot m)$.

Le procedure Buckshot e Frationation richiedono $O(k \cdot n)$ di tempo che equivale anche un tempo di esecuzione di una iterazione dell'algoritmo k-mean. Anche ogni iterazione dell'algoritmo K-means richiede $O(k \cdot n)$ di tempo perché abbiamo bisogno di calcolare la somiglianza dei documenti ai k semi diversi. Dobbiamo notare inoltre che la procedura di frazionamento può essere applicato in maniera casuale il raggruppamento dei documenti in n/m in diversi buckets. Naturalmente , si può anche sostituire il raggruppamento casuale con una procedura più accuratamente progettata per risultati più efficaci. Una tale procedura è quella di riordinare i documenti per l'indice j-esimo della più comune parola nel documento. Qui j è scelto

da un piccolo numero come 3 , che corrisponde alla media frequenza parole in dati. I documenti sono poi suddivisi in gruppi in base a questo ordinamento segmentando costantemente gruppi di m. Questo approccio assicura che i gruppi creati hanno almeno alcune parole di uso comune e non sono pertanto del tutto casuali. Questo a volte può fornire una migliore qualità dei centri che sono determinati dal algoritmo di frazionamento.
Una volta che il cluster iniziale dei centri sono stati determinati con l'uso di Buckshot o Fractionation si possono applicare gli algoritmi standard k-means di partizionamento. In particolare, ogni documento viene assegnato al più vicino dei centri. Il centroide di ciascun gruppo viene determinato come la concatenazione dei diversi documenti in un cluster. Tali centroidi sostituire il set di semi dalla ultima iterazione. Questo processo può essere ripetuto in un approccio iterativo per successivamente perfezionare i centri per i cluster. In genere, solo un piccolo numero di iterazioni necessarie, perché i più grandi miglioramenti si verificano solo nelle prime iterazioni.
È anche possibile utilizzare un numero di procedure per migliorare ulteriormente la qualità dei cluster. Queste procedure sono come segue: Operazione di divisione: Il processo di separazione può essere utilizzato al fine di perfezionare ulteriormente i cluster in gruppi di maggiore granularità. Questo risultato può essere ottenuto applicando la procedura buckshot ai singoli documenti in un cluster con $k = 2$, e quindi re-clustering intorno a questi centri. L'intera procedura richiede $O(k \cdot n_i)$ di tempo per un cluster contenenti n_i punti dati, e quindi la divisione tutti i gruppi richiede $O(k \cdot n)$ di tempo. Tuttavia, non è necessario dividere tutti i gruppi. Invece, solo un sottoinsieme dei gruppi può essere divisa . Questi sono i gruppi che non sono molto coerenti a contenere documenti di carattere eterogeneo. Al fine di misurare la coerenza di un gruppo, calcoliamo l'auto-somiglianza di un cluster. Questa somiglianza ci permette di capire la coerenza. Questa quantità può essere calcolata sia in termini di somiglianza dei documenti in un cluster per il suo centroide o in termini di somiglianza dei documenti raggruppati tra di loro. Il criterio può quindi essere applicato selettivamente solo i cluster che presentano una bassa auto-somiglianza. Questo contribuisce a creare cluster più coerenti.
Operazione di Join: L'operazione di unione tentano di fondere cluster simili in un unico cluster. Al fine di eseguire il merge, calcoliamo le parole topic di ciascun cluster, esaminando le più frequenti parole del centroide. Due gruppi sono considerati simili, se non vi è una significativa sovrapposizione

tra i problemi di attualità delle parole dei due gruppi.

È il caso di notare che il metodo è spesso indicato come Scatter-Gather, ma questo è a causa di come il metodo di clustering è stato presentato in termini di utilizzo per la navigazione di grandi raccolte nel documento originale [13]. Questo approccio può essere utilizzato per la navigazione di grandi raccolte di documenti, in quanto crea una gerarchia naturale di documenti analoghi.
In particolare, un utente può desiderare di esplorare la gerarchia di cluster in modo interattivo al fine di comprendere gli argomenti dei diversi livelli di granularità della raccolta. Una possibilità è quella di effettuare una clusterizzazione gerarchica a-priori; tuttavia, un tale approccio ha lo svantaggio di non essere in grado di unire e recluster le relative diramazioni della gerarchia ad albero on-the-fly quando un utente può aver bisogno .
Un metodo di interazione costante con il tempo con il metodo Scatter-Gather è stato presentato in [14]. Questo approccio presenta le parole chiave associate alle diverse parole chiave per un utente. L'utente può scegliere una o più di queste parole chiave, che corrisponde anche a uno o più gruppi. I documenti in questi cluster sono fusi e re-cluster per un miglior dettaglio in tempo reale. Questa granularità più fine di clustering viene presentata all'utente per ulteriori esplorazioni. L'insieme di documenti cui è prelevato dall'utente per l'esplorazione viene indicata come focus set. L'assunto fondamentale per consentire questo tipo di approccio è l'ipotesi di raffinatezza del cluster.

Capitolo 4

Analisi delle forme grafiche (parole)

Come già accennato in precedenza le forme grafiche delle parole sono composte da più parti, che a loro modo, ne compongono il significato.
La parte principale può essere considerata la radice, che determina il contesto di riferimento della forma grafica, parti secondarie sono i prefissi e i suffissi delle parole, che servono per modificare il significato della radice stessa. Infine si sono le desinenze che terminano le parole.

4.1 Radici

Si indica con radice, l'elemento base che esprime l'idea fondamentale della parola.
Nel linguaggio dei primitivi esistevano forse solo radici o radicali costituiti da voci imitative, senza nessuna determinazione. La radicale era una *parola-frase*. Oggi invece l'idea fondamentale della parola è specificata mediante suffissi che indicano anzitutto se si tratta di una cosa, persona o animale (sostantivo) o di un'azione(verbo) o di una qualità (aggettivo) o di un modo (avverbio), ecc. Es.: dalla radice *am-* si forma il sostantivo *amore*, il verbo *amare*, l'aggettivo *amabile*, l'avverbio *amabilmente*, ecc. Per ulteriori specificazioni si aggiungono a questi temi (*tema* è quanto resta di una parola togliendo la desinenza) le desinenze che indicano il genere e il numero dei nomi oppure la persona, il tempo e il modo dei verbi. In una parola si possono dunque distinguere: i *prefissi*, cioè le lettere o gruppi di lettere posti

prima della radice; la *radice*, che esprime l'idea originaria della parola nel modo più indeterminato; il *tema* (che nelle parole primitive è uguale alla radice); i *suffissi*, cioè le lettere o gruppi di lettere posti dopo la radice; la *desinenza*, che determina grammaticalmente la parola.
Ad esempio:

Parola	Radice	Suffisso	Desinenza	Tema
lodava	lod	av	a	lodav
lodevole	lod	evol	e	lodevol

Tabella 4.1: Composizione con suffisso

Le parole possono avere anche più prefissi o suffissi.

Parola	Prefisso	Radice	Suffisso	Desinenza	Tema
innominabile	in	nom	inabil	e	innominabil

Tabella 4.2: Composizione con prefisso e suffisso

Oppure

Parola	Prefisso	Radice	Suffisso	Desinenza	Tema
lodevolissimo	lod	evol	issim	o	lodevolissim

Tabella 4.3: Composizione con più suffissi

Le parole si raggruppano in famiglie derivanti tutte dalla stessa radice, che può essere comune a tutte le lingue dello stesso ceppo.

4.2 Desinenze

Prende il nome di desinenza la parte che compone la terminazione delle parole.
Nelle parole appartenenti alle parti variabili del discorso muta secondo le varie esigenze della flessione, che si divide in declinazione (per i nomi, gli aggettivi, i pronomi) e coniugazione (per i verbi). Le parole variabili hanno una parte che rimane sempre uguale, detta tema o radice, e una parte mobile, detta appunto desinenza. Nei nomi, negli aggettivi, nei pronomi e negli

articoli la desinenza indica il genere e il numero; nei verbi indica il modo, il tempo, il numero e la persona. Per mezzo della desinenza si può flettere il tema originario in tanti modi diversi, in relazione al bisogno di pensare un ogggetto o un fatto nei suoi diversi aspetti. Prendiamo, ad esempio, il tema lod-. Con desinenza -i (lodi) il nome diventa plurale, indica una molteplicità. Con la desinenza -o (lodo) esprimiamo un'azione, precisando la persona che la compie (io), e anche che l'azione è certa e si svolge ora mentre scrivo o parlo. See aggiungiamo la desinenza -erebbe (loderebbe) il significato muta profondamente: l'azione è compiuta dalla terza persona singolare (egli), ma non è più certa e reale, bensì condizionata o desiderata. Si noti però che mentre le parole primitive sono composte solo dal tema e dalla desinenza, le parole derivate sono composte anche con prefissi o suffissi. Dal tema originario si possono formare perciò ancor più numerose parole. Riprendiamo il tema lod-. Con l'aggiunta del suffisso -evol- si forma una nuova parola che indica la qualità di una cosa (un aggettivo qualificativo): lodevol-. A questa si aggiungono poi le desinenze indicante il genere e il numero: lodevole, lodevoli.

4.2.1 Suffissi

Indichiamo con suffisso la terminazione che si aggiunge al tema o alla radice di una parola (nome, aggettivo, verbo) per modificare il significato. Ecco un elenco di suffissi o terminazioni caratteristiche di nomi e aggettivi

-abile	-agno	-ario	-edine	-ese	-ico	-ile	-izio	-ore	-ugine
-acchio	-aio	-aro	-ela	-este	-icolo	-ime	-izio	-osmo	-ugio
-acco	-aldo	-asco	-ele	-esto	-idine	-ineo	-mento	-oso	-uglio
-aceo	-ale	-astro	-ello	-estre	-iere	-ingo	-monio	-otto	-ule
-aceo	-ame	-atico	-ena	-estro	-iero	-ino	-occhio	-ozzo	-ulo
-aco	-anda	-ato	-enda	-eto	-iggine	-io	-occio	-soio	-ume
-acolo	-ando	-ato	-endo	-evole	-igia	-ione	-occo	-sorio	-uno
-ado	-aneo	-atto	-ennza	-ezza	-igine	-ista	-oce	-sura	-uo
-aggine	-ano	-azzo	-ense	-ia	-igio	-ita	-ognolo	-toio	-uolo
-agine	-ante	-chio	-eo	-ibile	-iglia	-itico	-olo	-tore	-ura
-aglia	-anza	-colo	-erico	-icchio	-iglio	-ito	-ondo	-torio	-urno
-aglio	-ardo	-ecchio	-erno	-iccio	-iglio	-izia	-one	-tura	-uto
-aglio	-are	-eccio	-esco	-ice	-igno	-izie	-oneo	-uci	-zione

Tabella 4.4: Suffissi

A questi si devono poi aggiungere quelli che servono per formare le alterazioni dei nomi.
Suffissi usati per i verbi sono:

-acchiare	-eggiare	-ellare	-ezzare	-icchiare	-izzare	-ucchiare
-ecchiare	-eggiare	-ettare	-icare	-igare	-ottare	-ugliare

Tabella 4.5: Suffissi per i verbi

4.2.2 Prefissi

Particelle preposte a verbi, aggettivi o sostantivi per formare parole composte di significato diverso. Il significato della parola composta deriva in parte da quello del prefisso, che può essere una preposizione o un avverbio italiani o anche latini (post, super, circum) e greci (proto, pan, piro, teo) o particelle speciali come dis-, re-, ri-, ecc. I principali prefissi della nostra lingua sono:

a-	bis-	dis-	in-	mono-	para-	pro-	sopra-	teo-	vice-
ante-	circon-	e-	infra-	neo-	per-	proto-	sor-	termo-	zoo-
anti-	circum-	es-	inter-	ob-	peri-	pseudo-	sotto-	tra-	
arci-	con-	estra-	intra-	oltra-	piro-	re-	sovra-	trans-	
avam-	contra-	ex-	intro-	oltre-	po-	retro-	sta-	tras-	
avan-	contro-	extra-	lungi-	onni-	poli-	ri-	stra-	tri-	
bene-	de-	filo-	meta-	paleo-	pos-	s-	sub-	ultra-	
bi-	di-	fra-	mis-	pan-	pre-	semi-	sur-	uni-	

Tabella 4.6: Prefissi

4.2.3 Parti invariabili

La grammatica italiana distingue nove parti del discorso, di cui cinque *variabili* e quattro invariabili. Si dicono *invariabili* le parti del discorso le cui parole non mutano mai, cioè non si flettono, non si coniugano, non si declinano. Sono invariabili:

- l'avverbio;
- la preposizione;
- la congiunzione;
- l'interiezione.

L'avverbio infatti non ha genere, né numero, né tempo, né modo. Analogamente si comprende come siano invariabili le preposizioni semplici o le congiunzioni o le interiezioni. Nel seguito è stata aggiunta una piccola spiegazione di ognuna di queste parti.

Avverbio

Parte del discorso che determina, modifica o specifica il significato del verbo, e anche del nome, dell'aggettivo o di un altro avverbio, ai quali è riferito. Esso può indicare la qualità di un'azione o le sue circostanze di luogo, di tempo, di misura o anche l'affermazione, la negazione o il dubbio nei riguardi dell'azione stessa. Si distingue perciòò in varie categorie di avverbi:

di modo e maniera (o qualificativi)
di luogo
di tempo
di quantità
di affermazione, negazione dubbio
relativi
interrogativi

Tabella 4.7: Elenco avverbi

L'avverbio si pone prima del verbo, quando si vuol conferirgli risalto ed efficacia espressiva. Si pone all'inizio o alla dine di una frase (separato dal resto mediante una virgola) quando, indicando una probabilità o un giudizio, modifica il senso dell'intera proposizione. Riferito a un aggettivo o a un altro avverbio, si colloca davanti.
Nei casi in cui il avverbio è composto, l'avverbio si colloca tra l'ausiliare e il participio, oppure dopo il participio o anche prima dell'ausiliare. La prima forma dà tuttavia maggior risalto al valore dell'avverbio.
I comparativi si formano premettendo forme grafiche come

più	meno	tanto	quanto	così	come	altrettanto	...

Tabella 4.8: Avverbi comparativi

I superlativi aggiungendo il suffisso -mente al relativo dell'aggettivo o ripetendo il positivo. Le locuzioni avverbiali sono detti gli avverbi formati da più parole.

Preposizione

Parte del discorso che si prepone a un nome, a un pronome a un verbo all'infinito per formare i complementi, cioè per stabilire un rapporto tra le

parole.
Salvo il soggetto e il complemento oggetto, tutti i complementi sono introdotti da preposizioni. L'uso di queste ultime è quindi molto importante. Le proposizioni sono proprie e improprie. Le proprie sono quelle che nel discorso non hanno mai altro valore che quello di preposizione. Esse sono semplici e articolate, cioè risultati dall'unione i quelle semplici con gli articoli determinatici, improprie sono quelle costituite da altre parti del discorso che possono acquistare valore di preposizione.
Si noti che ciascuna preposizione può introdurre diversi complementi, i quali si riconoscono di volta in volta dal contesto del discorso.
A sua volta uno stesso complemento può essere retto da diverse preposizioni. Vi sono infine le locuzioni prepositive costituite dall'unione di due preposizioni, o di preposizioni con altre parole.

Congiunzione

Parte del discorso che serve per unire tra loro due o più parole oppure due o più preposizioni, si dice semplice quando è costituita da una parola semplice, composta quando risulta formata da una parloa composta. Si dice locuzione congiuntiva una congiunzione formata da due o più parole. Secondo la loro funzione le congiunzioni si distinguono in coordinate e subordinanti: le prime stabiliscono un legame tra i termini della relazione e danno luogo alla preposizione subordinata.

A loro volta le coordinate si suddividono in:

copulative	disgiuntive	avversative	dimostrative	conclusive

Tabella 4.9: Congiunzioni coordinate

Le subordinanti sono anch'esse di varia specie:

dichiarative	temporali	causali	finali	condizionali
concessive	modali	consecutive	eccettuative	

Tabella 4.10: Congiunzioni subrodinanti

Interiezione

Prende anche il nome di esclamazione. È costituita da un'espressione intercalata nel discorso, senza legami grammaticali col testo. È quindi un tipo di comunicazione spontanea, attraverso la quale si manifesta qualsiasi sentimento in forma immediata ed efficace.
L'interiezione può essere di:

ira	gloria	dolore	ammirazione
desiderio	rammarico	ironia	dubbio

Tabella 4.11: Interiezioni

Le interiezioni possono essere semplici, composte e improprie. Sono semplici quelle costituite da vocale seguita dalla lettera h, o da due vocali con in mezzo una h, sempre con il punto esclamativo, che può anche essere collocato alla fine della frase. Sono composte le interiezioni che risultano formate da parole composte. Locuzioni esclamative o interiezioni improprie sono quelle formate da più parole.

Capitolo 5

Processi per l'estrazione degli stem

Gli algoritmi che permettono di ridurre tutte le parole con la stessa radice a una forma comune, sono utile in molti settori della linguistica computazionale e nel recupero delle informazioni e risultano procedure molto simili alla lemmatizzazione. Mentre la forma dell'algoritmo varia con la sua applicazione, alcuni problemi linguistici sono comuni.
Vengono utilizzati due algoritmi per la lingua inglese che si differenziano fondamentalmente per una sola caratteristica fondamentale che viene messa a confronto:

- iterazione;
- massima verosimiglianza.

Iterazione è normalmente basata sul fatto che i suffissi sono attaccati allo stem in un 'certo ordine', che è, dove esiste, un ordine di classi di suffissi. Ogni ordine di classe può o non può essere rappresentato in ogni parola data. L'ultimo ordine contiene il suffisso di inflessione, come $-s.$ $-es.$ e $-ed.$ La classe precedente è derivazionale. Gli algoritmi iterativi sono semplicemente una procedura ricorsiva, che rimuove le stringhe in un determinato ordine ogni volta, partendo dalla fine della parola e lavorando verso l'inzio, non più di una corrispondenza è abilitata in una singola classe, per definizione. Si deve decidere quanti ordini di classe devono esserci, quali finali devono apparire e come i membri di ogni classe devono o non devono essere ordinati

per la scansione.
Il principio di massima verosimiglianza considera ogni terminazione di parola come una data classe, se più classi sono valide per un elemento, viene rimossa la classe più lunga. Per esempio se $-ion$ è rimossa, quando viene rilevato anche $-ation$, allora verrà rimosso $-ation$, in quanto la verosimiglianza è maggiore.
Un algoritmo basato solo sulla massima verosimiglianza usa una sola classe. Tutte le possibili combinazioni di affissi sono compilati e quindi ordinati per lunghezza. Se un occorrenza non è trovata, viene cercata un'occorrenza piùù corta. L'ovvio svantaggio di questo metodo è che devono essere generate tutte le possibili combinazioni di affissi. Un secondo svantaggio è che serve spazio per archiviare le combinazioni richieste.
Il primo svantaggio può anche essere presente in grande misura quando si è nell'impostazione di un algoritmo iterativo con ordine di classi. Per impostare l'ordine di classi, si deve esaminare un grande numero di desinenze. Inoltre, non è sempre chiaro a quale classe una determinata stringa deve appartenere per ottenere la massima efficienza.
In breve, mentre un algoritmo iterativo richiede un breve elenco di desinenze, introduce una serie di complicazioni nella preparazione della lista e di programmazione della routine.
Vengono presentato quindi due algoritmi per eliminare automaticamente i suffissi dalle parole di un corpus. Come afferma il professor Bolasco si deve stabilire una serie di operazioni atte a estrarre le informazioni che rappresentano il testo. Queste informazioni avranno lo scopo di facilitare il clustering del nostro corpus.

5.1 Svantaggi nell'automatizzazione dello stemming

Due principi sono utilizzati nella costruzione di un algoritmo di stemming: iterazione e di massima verosimiglianza. Un algoritmo basato esclusivamente su uno di questi metodi ha spesso inconvenienti che possono essere compensate impiegando una combinazione dei due principi.
Il metodo di iterazione è generalmente basato sul fatto che suffissi sono attaccati a radici in un 'certo ordine' (vedi, ad esempio, Lejnieks [32]). Og-

ni ordine di classe può o non può essere rappresentati in una parola data. L'ultimo ordine di classe si verifica alla fine di una parola e contiene suffissi flessivi come $-s$, $-es$, $and-ed$. Ordini di classi precedenti sono di tipo derivazionale. Come sottolineato da JL Dolby [comunicazione personale], ci sono diversi casi noti in cui un suffisso derivazionale ($-ness$) ne segue uno flessivo ($-ed$ o $-ing$). Ciò si verifica con alcuni aggettivi nominalizzati derivati da verbi utilizzando con uno di questi due flessiva finali, per esempio, *relatedness*, *disinterestedness*, *willingness*). Un esempio del più basso ordine di classe in una parola può essere ciò che è tecnicamente parte della radice (vedi $-ate$ esempio sopra), ma ai fini del calcolo si ritiene parte del finale. Un algoritmo iterativo derivante è semplicemente una procedura ricorsiva, come implica il nome, che rimuove stringhe in ogni ordine di classe uno alla volta, a partire dalla fine di una parola verso il suo inizio. Non più di un match è consentito all'interno di un singolo ordine di classe, per definizione. Bisogna decidere quanti ordini di classi ci dovrebbero essere, che terminazioni dovrebbero avvenirci, e se i membri di ogni classe devono essere internamente ordinato per la scansione.
Il principio di massima verosimiglianza afferma che all'interno di una determinata classe di terminazioni, se più di una finale prevede un match, deve essere rimossa la più lunga. Questo principio è stato attuato attraverso la scansione delle terminazioni di qualsiasi classe in ordine decrescente di lunghezza.

Ad esempio, se $-ion$ viene rimosso quando vi è anche una corrispondenza $-ation$, si dovrebbe rimuovere anche $-at$, che è per un altro ordine di classe. Per evitare questo ulteriore ordine di classe, $-ation$ dovrebbe precedere $-ion$ sulla lista.
Un algoritmo basato unicamente su una più lunga match utilizza un solo ordine di classe. Tutte le combinazioni possibili di affissi sono compilati e poi ordinati per lunghezza. Se non si trovano le terminazioni più lunghe, allora vengono scansionate le terminazioni più corte. L'ovvio svantaggio di questo metodo è che richiede la generazione di tutte le combinazioni possibili di suffissi. Un secondo inconveniente è la quantità di spazio di memorizzazione che le terminazioni richiedono.
Il primo svantaggio può anche essere presente in larga misura è la creazione di un algoritmo iterativo con molti classi. Per impostare le classi di ordine, si deve esaminare un gran numero di terminazioni. Inoltre, non è sempre chiaro a quale classe di una data stringa dovrebbe appartenere per

la massima efficienza. È anche possibile che il verificarsi di alcune classi dipenda dal contesto. In breve, mentre un algoritmo iterativo richiede un elenco più breve di terminazioni, introduce una serie di complicazioni nella preparazione della lista e sulla programmazione della routine.
Qualche idea della vastità di queste complicanze si ottiene attraverso un esame di un altro attributo di base di un algoritmo di derivazione: è libero dal contesto o sensibile al contesto. Dal momento che 'contesto' viene qui utilizzato per indicare qualsiasi attributo dello stem rimanente, 'context free' implica restrizioni qualitative e quantitative alla rimozione di terminazioni. In un algoritmo libero dal contesto, viene accettato il primo finale in ogni classe. Ma ci dovrebbe presumibilmente essere almeno qualche restrizione quantitativa, nel senso che lo stem restante non deve essere di lunghezza zero. Un esempio di questo caso estremo è la corrispondenza di $-$*ability* a *ability* nonché *computability*. Infatti, ogni stem utile consiste solitamente di almeno due lettere, e spesso tre o quattro costituiscono un minimo necessario. La limitazione della lunghezza dello stem varia con la desinenza; come cambia nuovamente può essere determinato solo in relazione al sistema totale. L'algoritmo sviluppato dal professor John W. Tukey dell'Università di Princeton associa un limite inferiore a ogni finale. Alcuni dei suoi limiti sono piuttosto elevati (ad esempio, sette lettere). Lovins è stata meno conservatrice e ha proposto una lunghezza minima dello stem pari a due; terminazioni alcuni hanno un ulteriore limite in quanto la loro lunghezza minima dello stelo è tre, quattro, o cinque lettere.
Il tipo di restrizioni qualitative contestuali che deve essere imposto è una domanda un po' aperta. Al fine di ottenere i migliori risultati, alcuni finali non deve essere rimosso in presenza di certe lettere nello stem risultante, di solito quelle lettere che precedono immediatamente il finale. La forma più desiderabile di regola *context-sensitive* è di carattere generale che può essere applicato al numero di terminazioni, ma tali regole sono poche. Un esempio è 'non rimuovere un finale che inizia con $-en-$, e seguenti'. La violazione di questa regola potrebbe cambiare *seen* in *se*$-$, uno stem potenzialmente ambiguo (cf. *sea* meno $-a$, *seize* meno $-ize$, ecc.). Ma un certo numero di regole deve essere creato per terminazioni dei singoli al fine di evitare determinati casi speciali proprie di quei finali. Si può andare di tutto in questa direzione, con rendimenti sempre più piccoli.
In realtà tutte queste problematiche sono state risolte per l'algoritmo di Lovins utilizzando come dizionario di confronto il dataset Wordnet, quindi

se una parola alla quale viene tolta la terminazione è presente in Wordnet, allora la parola viene modificata, quindi rimane comunque un certo grado di ambiguità.
Nell'applicazione dell'algoritmo di Porter non è stato considerato l'utilizzo del dizionario Wordnet.

5.2 Algoritmo di Porter

Per presentare l'algoritmo iterativo di stripping nella sua interezza abbiamo bisogno di alcuni definizioni.

Una 'consonante' in una parola è una lettera diversa da A, E, I, O o U, e la Y preceduta da una consonante (Il fatto che il termine 'consonante' è definito in questo modo non lo rende ambiguo.) Quindi, in TOY le consonanti sono T e Y, e in Syzygy sono S, Z e G. Se una lettera non è una consonante è una 'vocale'.

Una consonante sarà indicata con c, una vocale da v. Un elenco ccc ... di lunghezza maggiore di 0 verrà indicato con C, e un elenco vvv... di lunghezza maggiore di 0 indicheremo con V. Qualsiasi parola, o parte di una parola, ha quindi una delle quattro forme:

- CVCV ... C
- CVCV ... V
- VCVC ... C
- VCVC ... V

Questi possono tutti essere rappresentata dalla forma singola

$$[C]\,VCVC\ldots[V]$$

dove le parentesi quadre denotano presenza arbitraria del loro contenuto. Utilizzando $(VC)^m$ per indicare VC ripetuto m volte, questo può ancora essere scritto come

$$[C]\,(VC)^m\,[V]$$

m sarà chiamato 'misura' di qualsiasi parola o parte di parola quando viene rappresentato in questo modulo. Il caso $m = 0$ copre la parola nulla. Ecco alcuni esempi:

- m = 0 TR, EE, TREE, Y, BY.
- m = 1 TROUBLE, OATS, TREES, IVY.
- m = 2 TROUBLES, PRIVATE, OATEN, ORRERY.

Le 'regole' per la rimozione di un suffisso saranno date sotto la forma

$$(\text{Condizione})\text{S}_1 \rightarrow \text{S}_2$$

Ciò significa che se una parola termina con il suffisso S_1, e che lo stem prima di S_1 soddisfa la condizione data, allora S_1 viene sostituito da S_2. La condizione è di solito dato in termini di m, ad esempio:

$$(\text{M} > 1)\text{EMENT} - >$$

Qui S_1 è 'EMENT' e S_2 è nullo. Civuole sostiturire REPLACEMENT in REPLAC, visto che REPLAC è una parte della parola per cui $m = 2$.

La 'condizione' può anche contenere i seguenti elementi:

- *S - la radice termina con S (e analogamente per le altre lettere).
- *V* - la radice contiene una vocale.
- *D - la radice termina con una doppia consonante (ad es-TT,-SS).
- *O - la radice termina CVC, in cui il secondo non c è W, X o Y (ad esempio, -WIL,-HOP).

E la parte condizione può anche contenere espressioni con AND, OR e NOT, in modo che

$$(\text{m} > 1\text{and}(*\text{Sor} * \text{T}))$$

Testi se uno stem con $m > 1$ finisca in S o T, mentre

$$(*\text{dandnot}(*\text{Lor} * \text{Sor} * \text{Z}))$$

test per uno stem che finisca con una doppia consonante diversa da L, S o Z. Condizioni elaborate come queste sono richieste solo raramente.

In una serie di regole scritte in sequenza, solo una è obbedita, e questa sarà quella con la più lunga S1 corrispondente per la parola data. Per esempio, con

- $SSES \rightarrow SS$
- $IES \rightarrow I$
- $SS \rightarrow SS$
- $S \rightarrow$

(Qui le condizioni sono tutte nulli) CARESSES si trasforma in CARESS mentre SSES è la più lunga corrispondenza per S1. Equivalentemente CARESS si trasforma in CARESS (S1=‘SS’) e CARES in CARE (S1=‘S’). Nelle regole qui di seguito, alcuni esempi della loro applicazione, di successo o meno, sono riportati a destra in minuscolo. L’algoritmo ora segue:
Fase 1a:

$SSES$	$\rightarrow$	SS	caresses	$\rightarrow$	caress
			ponies	$\rightarrow$	poni
IES	$\rightarrow$	I	ties	$\rightarrow$	ti
SS	$\rightarrow$	SS	caress	$\rightarrow$	caress
S	$\rightarrow$		cats	$\rightarrow$	cat

Fase 1b:

(m $>$ 0)	EED	$\rightarrow$	EE	feed	$\rightarrow$	feed
				agreed	$\rightarrow$	agree
($*$v$*$)	ED	$\rightarrow$		plastered	$\rightarrow$	plaster
				bled	$\rightarrow$	bled
($*$v$*$)	ING	$\rightarrow$		motoring	$\rightarrow$	motor
				sing	$\rightarrow$	sing

Se la seconda o la terza regola della Fase 1b ha successo, allora:

AT	$\rightarrow$	ATE	conflat(ed)	$\rightarrow$	conflate
BL	$\rightarrow$	BLE	troubl(ed)	$\rightarrow$	trouble
IZ	$\rightarrow$	IZE	siz(ed)	$\rightarrow$	size
($*$dandnot($*$Lor $*$ Sor $*$ Z))	$\rightarrow$	singleletter	hopp(ing)	$\rightarrow$	hop
			tann(ed)	$\rightarrow$	tan
			fall(ing)	$\rightarrow$	fall
			hiss(ing)	$\rightarrow$	hiss
			fizz(ed)	$\rightarrow$	fizz
(m = 1and $*$ o)	$\rightarrow$	E	fail(ing)	$\rightarrow$	fail
			fil(ing)	$\rightarrow$	file

La regola per mappare una lettera singola causa la rimozione di una delle due lettere doppie. La $-E$ è rimosso in $-AT$, $-BL$ e $-IZ$, in modo che i suffissi $-ATE$, $-BLE$ e $-IZE$ possano essere riconosciuti in seguito. E questi possano essere rimossi nella fase 4.
Fase 1c:

$(*v*)$	Y	$\rightarrow$	I	happy	$\rightarrow$	happi
				sky	$\rightarrow$	sky

Fase 1 si occupa di plurali e dei participi passati. Le fasi successive sono molto più semplici.

Fase 2

(m > 0)	ATIONAL	$\rightarrow$	ATE	relational	$\rightarrow$	relate
(m > 0)	TIONAL	$\rightarrow$	TION	conditional	$\rightarrow$	condition
				rational	$\rightarrow$	rational
(m > 0)	ENCI	$\rightarrow$	ENCE	valenci	$\rightarrow$	valence
(m > 0)	ANCI	$\rightarrow$	ANCE	hesitanci	$\rightarrow$	hesitance
(m > 0)	IZER	$\rightarrow$	IZE	digitizer	$\rightarrow$	digitize
(m > 0)	ABLI	$\rightarrow$	ABLE	conformabli	$\rightarrow$	conformable
(m > 0)	ALLI	$\rightarrow$	AL	radicalli	$\rightarrow$	radical
(m > 0)	ENTLI	$\rightarrow$	ENT	differentli	$\rightarrow$	different
(m > 0)	ELI	$\rightarrow$	E	vileli	$\rightarrow$	vile
(m > 0)	OUSLI	$\rightarrow$	OUS	analogousli	$\rightarrow$	analogous
(m > 0)	IZATION	$\rightarrow$	IZE	vietnamization	$\rightarrow$	vietnamize
(m > 0)	ATION	$\rightarrow$	ATE	predication	$\rightarrow$	predicate
(m > 0)	ATOR	$\rightarrow$	ATE	operator	$\rightarrow$	operate
(m > 0)	ALISM	$\rightarrow$	AL	feudalism	$\rightarrow$	feudal
(m > 0)	IVENESS	$\rightarrow$	IVE	decisiveness	$\rightarrow$	decisive
(m > 0)	FULNESS	$\rightarrow$	FUL	hopefulness	$\rightarrow$	hopeful
(m > 0)	OUSNESS	$\rightarrow$	OUS	callousness	$\rightarrow$	callous
(m > 0)	ALITI	$\rightarrow$	AL	formaliti	$\rightarrow$	formal
(m > 0)	IVITI	$\rightarrow$	IVE	sensitiviti	$\rightarrow$	sensitive
(m > 0)	BILITI	$\rightarrow$	BLE	sensibiliti	$\rightarrow$	sensible

Fase 3

(m > 0)	ICATE	→	IC	triplicate	→	triplic
(m > 0)	ATIVE	→		formative	→	form
(m > 0)	ALIZE	→	AL	formalize	→	formal
(m > 0)	ICITI	→	IC	electriciti	→	electric
(m > 0)	ICAL	→	IC	electrical	→	electric
(m > 0)	FUL	→		hopeful	→	hope
(m > 0)	NESS	→		goodness	→	good

Fase 4

(m > 1)	AL	→	revival	→	reviv
(m > 1)	ANCE	→	allowance	→	allow
(m > 1)	ENCE	→	inference	→	infer
(m > 1)	ER	→	airliner	→	airlin
(m > 1)	IC	→	gyroscopic	→	gyroscop
(m > 1)	ABLE	→	adjustable	→	adjust
(m > 1)	IBLE	→	defensible	→	defens
(m > 1)	ANT	→	irritant	→	irrit
(m > 1)	EMENT	→	replacement	→	replac
(m > 1)	MENT	→	adjustment	→	adjust
(m > 1)	ENT	→	dependent	→	depend
(m > 1and(∗Sor ∗ T))	ION	→	adoption	→	adopt
(m > 1)	OU	→	homologou	→	homolog
(m > 1)	ISM	→	communism	→	commun
(m > 1)	ATE	→	activate	→	activ
(m > 1)	ITI	→	angulariti	→	angular
(m > 1)	OUS	→	homologous	→	homolog
(m > 1)	IVE	→	effective	→	effect
(m > 1)	IZE	→	bowdlerize	→	bowdler

I suffissi sono ora rimossi. Tutto ciò che rimane è un po' di riordino.
Fase 5a

(m > 1)	E	→	probate	→	probat
			rate	→	rate
(m = 1andnot ∗ o)	E	→	cease	→	ceas

Fase 5b

(m > 1and ∗ dand ∗ L)	→	singleletter	control
		controll	→
		roll	→ roll

L'algoritmo è attento a non rimuovere un suffisso quando lo stem è troppo breve, la lunghezza dello stem essendo data dalla sua misura, m. Non è questo un approccio linguistico. Ci si è limitati ad osservare che m può essere utilizzato molto in modo efficace per aiutare per decidere se sia o no stato saggio eliminare un suffisso.
Per esempio, nei seguenti due liste:

listA	listB
RELATE	DERIVATE
PROBATE	ACTIVATE
CONFLATE	DEMONSTRATE
PIRATE	NECESSITATE
PRELATE	RENOVATE

-ATE viene rimosso dalle parole della lista B, ma non dalla lista A. Questo significa che le coppie DERIVATE/DERIVE, ACTIVATE/ACTIVE, DEMONSTRATE/DEMONSTRABLE, NECESSITATE/NECESSITOUS, si confondono insieme. Il fatto che tentatare di identificare questi prefissi può far ottenere i risultati piuttosto incoerente. Così PRELATE non perde l'-ATE, ma ARCHPRELATE diventa ARCHPREL. In pratica questo non importa troppo, perché la presenza del prefisso diminuisce la probabilità di una errata eliminazione.

Suffissi complessi vengono rimossi bit per bit in diverse fasi. Così GENERALIZATIONS è ridotto a GENERALIZATION (Fase 1), poi a GENERALIZE (Fase 2), poi a GENERAL (Fase 3), e quindi a GENER (Passaggio 4). OSCILLATORS è ridotto a OSCILLATOR (Fase 1), poi ad OSCILLATE (Fase 2), poi a OSCILL (punto 4), e poi a OSCIL (punto 5). In un vocabolario di 10.000 parole, la riduzione delle dimensioni dello stem è stato distribuito tra i passaggi come segue:

Suffix stripping of a vocabulary of 10,000 words	
Number of words reduced in step 1:	3597
Number of words reduced in step 2:	766
Number of words reduced in step 3:	327
Number of words reduced in step 4:	2424
Number of words reduced in step 5:	1373
Number of words not reduced:	3650

Il vocabolario conseguente contiene 6.370 voci distinte. Così il processo di stripping dei suffissi ha ridotto la dimensione del vocabolario di circa un terzo.

5.3 Algoritmo di Lovins

L'algoritmo di Lovins è basato sul suo articolo [34] in cui viene proposto un elenco dei suffissi presenti nella lingua inglese. Da questo studio è stato sviluppato l'algoritmo sull'applicazione a forza bruta della rimozione dei suffissi.
Un elenco delle classi finali (concatenazioni di suffissi) è stato compilato nel modo seguente: un elenco preliminare è stato basato su desinenze trovate in una piccola porzione di catalogo in fase di sviluppo del progetto Intrex e finali nell'elenco utilizzato presso l'università di Harvard. La lista preliminare è stata valutata applicando le terminazioni su questo elenco di una parte dell'uscita dalla routine tailcropping Tukey, livelli 1-3, e dei volumi 5 - 7 del normale e la parola inglese elenco [9] (volumi 5-7 contengono ininterrottamente parole in ordine alfabetico quando scritte da destra a sinistra).
a questo punto l'algoritmo è applicato semplicemente verificando la presenza dei suffissi, ordinandoli per ordine di lunghezza, dal più lungo al più corto. Le operazioni risultano:

1. seleziona prossimo suffisso;
2. verifica la presenza del suffisso;
3. elimina il suffisso;
4. verifica la presenza della nuova forma grafica all'interno del dizionario di controllo;
5. torna a (1) fino alla terminazione.

Questo algoritmo è computazionalmente oneroso, in quanto deve eseguire un numero di iterazioni comprese fra:

$$num\_parole \div num\_regole \times num\_parole$$

ma è chiaro che il caso medio è molto vicino al limite superiore, in quanto non necessariamente una parola contiene alcun suffisso, ma per verificarlo devono essere eseguiti tutti i controlli.
L'elenco dei suffissi risulta:

-alistically	-arizability	-izationally

Questi sono i suffissi più lunghi, quindi i meno verificabili, si può notare che al loro interno contengono a loro volta dei suffissi composti da un numero inferiore di parole.
Altri suffissi sono:

-arisations	-arizations	-entialness	-antialness

Si nota che accorciando la lunghezza dei suffissi, aumenta il loro numero, si osserva inoltre che questi suffissi sono in realtà composti da suffissi composti tra loro.

-antaneous	-ationally	-entiality	-istically
-antiality	-ativeness	-entialize	-itousness
-arisation	-eableness	-entiation	-izability
-allically	-arization	-entations	-ionalness
-izational			

In effetti, come viene spiegato nell'algoritmo di Porter, questi suffissi potevano essere eliminati con iterazioni successive, però oltre a rendere più complesso l'algoritmo a forza bruta, potevano anche non essere considerate parole che esistono eliminando tutto il suffisso, ma non esistono togliendo le singole parti che compongono l'intero suffisso.

-ableness	-eousness	-ionality	-lessness
-arizable	-ibleness	-ionalize	-entation
-icalness	-iousness	-entially	-ionalism
-izations			

In realtà questo grande numero di suffissi incide su una piccola quantità di parole, infatti nei 3 gruppi di suffissi, che sono stati suddivisi per quantità di parole modificate, i primi due gruppi hanno dimensione simile (136 e 280) mentre il solo suffisso $-s$ modifica da solo un intero terzo di tutte le parole modificate.

-alities	-ateness	-ativism	-entials
-ibility	-icalize	-ingness	-istical
-ivities	-ousness	-ability	-ariness
-atingly	-elihood	-entiate	-icalism
-ication	-iolally	-iteness	-ization
-aically	-aristic	-ational	-encible
-entness	-icalist	-icianry	-isation
-iveness	-izement	-alistic	-arizing
-atively	-entally	-fulness	-icality
-ination	-ishness	-ivistic	-oidally

-alness	-arized	-atives	-encies
-entist	-ialize	-icists	-ioning
-lessly	-aceous	-ancial	-arizer
-azable	-encing	-eously	-ically
-ifully	-ionist	-nesses	-acious
-ancies	-atable	-eature	-ential
-ialist	-icance	-ionasl	-iously
-oidism	-action	-ancing	-ations
-efully	-enting	-iality	-icians
-ionate	-istics		

-acity	-alist	-anced	-aries	-aroid	-ative
-early	-ement	-ening	-fully	-icide	-idine
-iness	-ional	-ities	-izers	-otide	-aging
-ality	-ances	-arily	-ately	-ators	-ehood
-enced	-ental	-ially	-icism	-iedly	-ingly
-ioned	-itous	-izing	-ously	-aical	-alize
-antic	-arity	-ating	-atory	-eless	-ences
-ented	-icant	-icist	-ihood	-inism	-ished
-ively	-oidal	-acies	-alism	-allic	-arial
-arize	-ation	-ature	-elily	-eness	-ently
-ician	-icity	-inate	-inity	-istic	-ivity

-ages	-ants	-ates	-edly	-ency	-hood
-ibly	-iful	-ious	-ized	-ness	-ying
-ally	-aric	-atic	-eful	-ened	-ials
-ical	-ines	-isms	-izer	-ogen	-yish
-able	-ance	-arly	-ator	-eity	-enly
-ians	-ides	-ings	-ists	-less	-ward
-ably	-ancy	-ated	-ealy	-ence	-eous
-ible	-iers	-ions	-itic	-lily	-wise

-aic	-ary	-ear	-ery	-ian
-ier	-ing	-ist	-ive	-ous
-als	-ata	-ely	-ese	-ics
-ies	-ion	-ite	-ize	-acy
-ant	-ate	-ene	-ful	-ide
-ily	-ish	-ity	-oid	-age
-ars	-eal	-ent	-ial	-ied
-ine	-ism	-ium	-one	

-as	-ia	-on	-yl
-ae	-ed	-ic	-or
-al	-en	-is	-um
-ar	-es	-ly	-us

-o	-a
-s	-e
-y	-i

Capitolo 6

Classificazione del testo

6.1 Definizione

La classificazione del testo (TC) è la procedura che associa un valore booleano a ogni coppia documento - categoria per definirne l'appartenenza (T(o meno (F) del documento alla categoria stessa. Più formalmente si vuole approssimare una funzione obiettivo $\breve{\Phi} : D \times C \rightarrow \{T, F\}$ che descrive come vanno classificati i documenti per mezzo di una funzione $\Phi : D \times C \rightarrow \{T, F\}$ chiamata classificatore, in modo tale da far coincidere il più possibile le funzioni $\breve{\Phi}$ e Φ Ipotiziamo inoltre che:

- le categorie siano etichette esclusivamente simboliche, che non danno alcuna conoscenza aggiuntiva;
- non è disponibile alcuna conoscenza esterna, per cui la classificazione deve essere realizzata sulla conoscenza estratta dagli stessi documenti.

I metodi di TC che si vedranno sono del tutto generali e non dipendono da risorse particolari, che potrebbero non essere disponibili o costose. Basandosi solo sulla conoscenza estratta, quindi basandosi esclusivamente sulla sua semantica, e visto che quest'ultima è un concetto soggettivo, risulterà che l'appartenenza di un documento a una categoria non sarà deciso in modo deterministico.

6.1.1 Classificazione mono/multi label

La classificazione di documenti può avere diversi obiettivi, a seconda della sua utilizzazione. Si può avere la necessità di assegnare solo 1 categoria a ogni documento (classificazione mono-label) e le categorie non si sovrapporranno mai. Oppure il caso di assegnare qualunque numero di categorie a un documento (classificazione multi-label), in cui le categorie si sovrappongono. Caso particolare del mono-label è il caso binario, in cui un documento appartiene, o no, a una determinata categoria.
Il caso binario è importante in quanto:

- le applicazioni come quelle di filtraggio sono problemi di classificazione binaria;
- risolvere un caso binario è anche risolvere un caso multi-label;
- la maggior parte della letteratura TC è in termini di caso binario;
- la classificazione binaria è un caso particolare di tecniche esistenti mono-label, e sono più semplici da utilizzare.

Quindi una classificazione $C = \{c_1, \dots c_{|C|}\}$ costituita da $|C|$ problemi indipendenti della classificazione dei documenti in D per una determinata categoria c_i , per $i = 1, \dots, |C|$. Un classificatore per c_i è quindi una funzione $\Phi : D \to \{T, F\}$.

6.1.2 Classificazione basata sulle categoria o sui documenti

Due sono le possibilità di utilizzo di un classificatore di testo. Dato $d_j \in D$ si potrebbero desiderare tutte le $c_j \in C$ in base alle quali deve essere presentata (classificazione basata sui documenti o DPC), in alternativa, dato $c_j \in C$, si possono cercare tutte le $d_j \in D$ che dovrebbero essere archiviate sotto di essa (classificazione basata sulle categorie o CPC). Questo è importante da ricordare perché sia la serie C che la serie D possono non essere completamente disponibili dall'inizio.
DPC è adatto quando i documenti saranno disponibili in momenti diversi, esempio posta elettronica, CPC è adatto quando sono necessarie più categorie di quante non sono state considerate inizialmente. Di fatto DPC

è usato molto piu spesso di CPC, in quanto è una situazione molto più comune.

6.2 Approccio Machine Learning nella classificazione del testo

Negli anni 80 l'approccio più popolare per la creazione di classificatori automatici era quello di costruire manualmente, con l'utilizzo delle tecniche di knowledge engineering (KE), un sistema esperto in grado di prendere decisioni TC. Normalmente si trattava di regole logiche definite manualmente, una per categoria, del tipo:

$$f\,\langle DNF formula\rangle\, then\,\langle category\rangle$$

Una formula DNF (disjunctive normal form)) è una disgiunzione di clausole, il documento viene classificato sotto una categoria solo se soddisfa la formula, ovvero una delle clausole.
Il grosso problema è l'acquisizione della conoscenza, ovvero le regole devono essere definite manualmente da un tecnico di conoscenze con l'aiuto di un esperto di domninio. Se i set delle categorie vengono aggiornati, allora queste due figure devono intervenire nuovamente, e se il classificatore viene applicato a un dominio diverso, un esperto di dominio diverso deve intervenire e ripetere il lavoro dall'inizio.
Dagli anni 90 l'approccio ML ha guadagnato popolarità ed è diventato quello dominante, almeno nella ricerca. In questo approccio il processo crea automanticamente un classificatore di una categoria osservando le caratteristiche di un insieme di documenti classificati manualmente da un esperto del settore e, da quelle caratteristiche, il processo induttivo rinviene le caratteristiche che un nuovo documento debba essere classificato nella opportuna categoria. Si parla di un problema di apprendimento supervisionato, poiché il processo è 'supervisionato' dalla conoscenza delle categorie e delle istanze di formazione che gli appartengono.
I vantaggi del metodo ML contro l'approccio KE sono evidenti, si cerca di costruire un classificatore automatico di classificatori così da poter aggiornare sia il classificatore, che la serie di categorie o il dominio al quale poterli applicare.

6.2.1 Training set, Test set e di validazione

L'approccio ML si basa basa sulla disponibilità di un corpus iniziale di documenti preclassificati in categorie. Data quindi la funzione $\Phi : D \times C \rightarrow \{T, F\}$ il documento deve essere un esempio positivo (T) se il documento appartiene alla categoria, oppure negativo (F) in caso contrario.
Nelle impostazioni di ricerca è opportuno valutare l'efficacia di un classificatore dopo che è stato costruito, e viene confrontata la sua validazione con i risultati delle decisioni degli esperti. In questo caso, prima di creare il classificatore costruiamo 2 corpus dividendo il corpus iniziale in due parti non necessariamente uguali:

- un training set (e validation set) TV in cui il classificatore 'impara' le caratteristiche dei documenti
- un test set TE che veridica l'efficacia dei classificatori ottenuti con il training set, i cui risultati vengono confrontati con i valori delle decisioni degli esperti.

I documenti TE non possono partecipare alla costruzione induttiva dei classificatori, altrimenti invaliderebbero i risultati sperimentali ottenuti, che sarebbero irrealisticamente buoni.
Questo approccio è chiamato di train-test. Un 'alternativa è l'approccio k-fold di validazione incrociata, in cui k classificatori diversi sono costruiti suddividendo il corpus iniziale in k insiemi disgiunti e applicando iterativamente l'approccio train-test sulle coppie di train e di test set.

6.2.2 Informazioni tecniche di recupero e classificazione del testo

La classificazione del testo si basa molto sull'IR. La ragione è che il TC è contenuto nelle attivita di gestione dei documenti e come tale ne condivide molte caratteristiche e altre attività di IR come la ricerca del testo. Tecniche IR vengono utilizzate in 3 fasi del ciclo di vita del classificatore di testo:

1. l'indicizzazione IR-style viene eseguita sui documenti del corpus iniziale e su quelli da classificare nella fase operativa;
2. tecniche IR-style sono usate per la costruzione intuitiva dei classificatori;

3. valutazione IR-style viene eseguita per verificare l'efficacia dei classificatori.

I vari approcci alla classificazione differiscono principalmente per come affrontare (2), anche se in alcuni casi non standard sono applicati anche in (1) e (3).

6.3 Metodo di classificazione classico

In questo metodo di classificazione viene trasformato ogni singolo documento d_j in un vettore di parole, e viene costruita una tabella sparsa che contiene tante righe quanti sono i documenti e tante colonne quante sono le parole diverse del dataset.
L'operazione di stemming non fa altro che diminuire il numero di parole diverse nell'intero dataset. Una volta costruita questa matrice, vengono utilizzati alcuni dei più noti metodi di classificazione di cui ne riportiammo una breve spiegazione nel seguito.

6.3.1 Algoritmi di classificazione

Presentiamo una veloce presentazione degli algoritmi di classificazione per il metodo classico, che sono stati scelti opportunamente per confrontare il lavoro con una grande parte di lavori precedenti, e inoltre sono tutti implementati nello strumento Weka. Gli algoritmi utilizzati sono i seguenti:

- C4.5;
- Kstar;
- BayesNet;
- NaiveBayesMultinomial;
- Random Forest;

Algoritmo C4.5 (J48)

Si tratta di un algoritmo usato per generare un albero decisionale sviluppato da Ross Quinlan [43] questo algoritmo è un'estensione del precedente

algoritmo ID3 sempre di Quinlan. L'albero decisionale può essere usato per la classificazione.
Viene costruito l'albero decisionale da un set di train usando il concetto di entropia dell'informazione. A ogni nodo dell'albero, l'algoritmo decide quale attributo meglio suddivide i campioni in sottogruppi di una classe o di altro. Ricorsivamente si procede nei sottoinsiemi.

Algoritmo K*

K* è un classificatore basato su un esempio, che è una classe di test basata su un'istanza di prova della classe dei casi di formazione simili ad esso, determinato da una qualche funzione di somiglianza. Differisce da altri riconoscitori in quanto utilizza una funzione entropica basata sulla distanza.

Algoritmo BayesNet

Le reti di apprendimento di Bayes utilizzano diversi algoritmi di ricerca e misure di qualità. Attraverso un insieme di variabili, viene costruita una tabella di probabilità e una rete B che rappresenta le distribuzioni di probabilitità della tabella stessa. Le strutture comuni alle reti di Bayes sono algoritmi di apprendimento come K2 e B.

Algoritmo Naïve Bayes Multinomiale

Il modello multinomiale cattura le informazioni di frequenza nei documenti. In questo modello, un documento è ordinato come sequenza di parole tratte dal vocabolario stesso. Date le stime di questi parametri calcolati dai documenti di formazione, la classificazione può essere eseguita sui documenti di prova calcolando la probabilità a posteriori di ciascuna classe data l'evidenza del documento di prova e selezione la classe con la più alta probabilità.

Algoritmo Random Forest

Questo metodo costruisce molti alberi decisionali e restituisce la modalità di uscita dei singoli alberi. La selezione di un sottoinsieme casuale di caratteristiche è un esempio del metodo sottospazio casuale, che è un modo per implementare discriminazione stocastico è stato introdotto indipendente-

mente sia da Ho. [25] [26] che da Amit e Geman,[1]. per costruire un insieme di alberi decisionali con la variazioni controllate.

6.3.2 Risultati ottenuti con il metodo classico

Ogniuno degli algoritmi visti in precedenza è stato applicato sia alle parole del dataset di partenza, sia alle parole alle quali era stato applicato lo stemming, sia con l'algoritmo di Porter che con l'algoritmo di Lovins. In più sono stati applicati anche gli stessi in modo 'parziale' ovvero limitando il numero di regole applicate, per Porter sono state applicate dapprima le prime 2, poi le prime due e le due seguenti, infine tutte quante. Per Lovins invece si sono trovati 3 gruppi di regole che producessero approssimativamente un ugual numero di modifiche, per cui sono state suddivise nelle prime 136 su 291, poi 290 su 291 infine tutte, si noti che la sola regola dell'eliminazione della 's' finale genera all'incirca un terzo di tutte le modifiche fatte. In ogniuno dei grafici è presente la serie relativa alla classificazione delle parole non trasformate (parole), le serie relative ai tre livelli di stemming con l'algoritmo di Porter (parole12, parole1234, parole123456) e quelle per i tre livelli di applicazione dell'algoritmo di Lovins (parole136, parole290, parole291).
I risultati sono i seguenti:

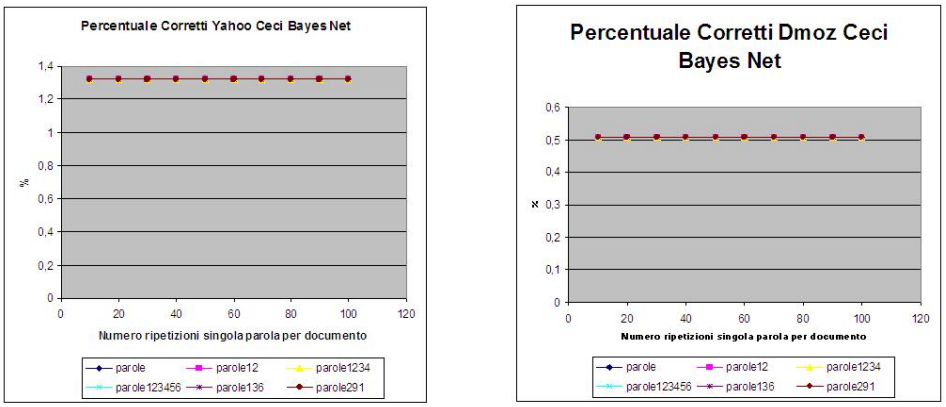

Figura 6.1: Risultati per il metodo BayesNet per Yahoo Ceci e Dmoz Ceci

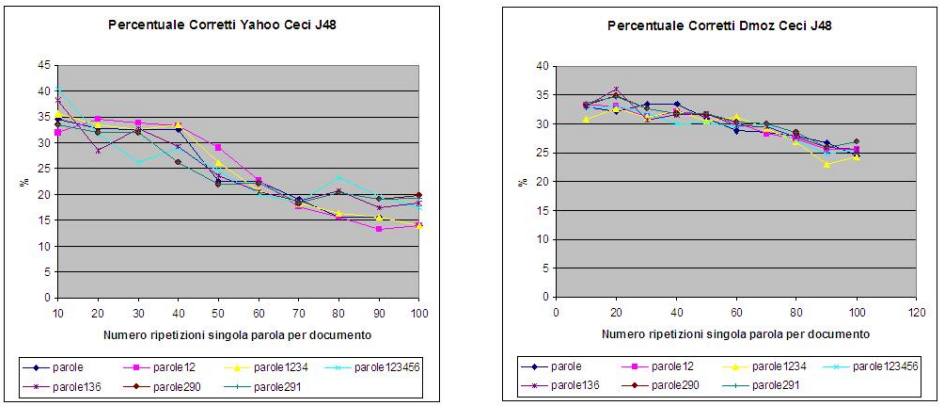

Figura 6.2: Risultati per il metodo J48 per Yahoo Ceci e Dmoz Ceci

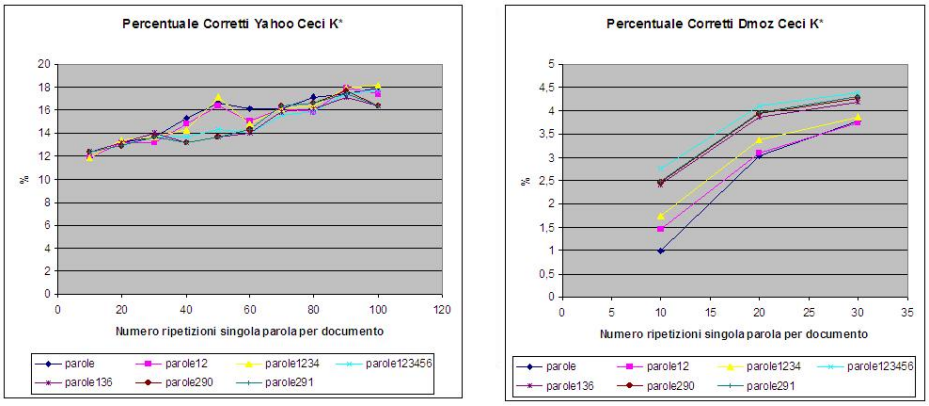

Figura 6.3: Risultati per il metodo K* per Yahoo Ceci e Dmoz Ceci

Per l'algoritmo K* sono state implementate meno istanze, in quanto il tempo per eseguire tutti gli esperimenti che sono stati implementati negli altri algoritmi, era troppo lungo, ma si evince comunque che i risultati non variano molto.

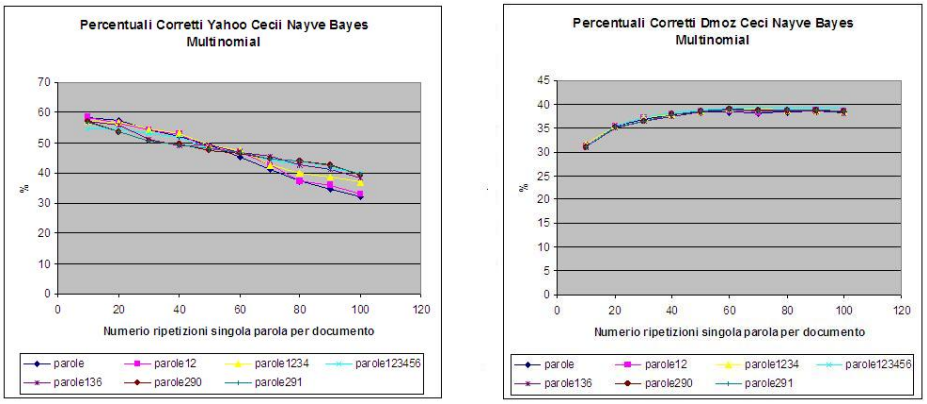

Figura 6.4: Risultati per il metodo Naive Bayes Multinomial per Yahoo Ceci e Dmoz Ceci

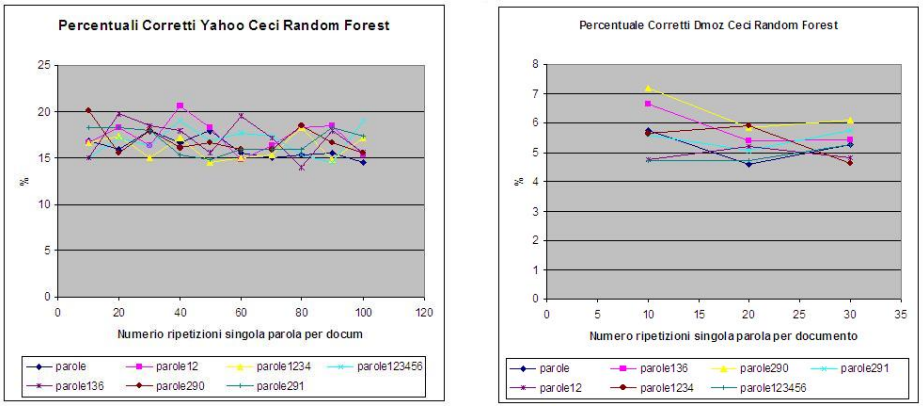

Figura 6.5: Risultati per il metodo Random Forest per Yahoo Ceci e Dmoz Ceci

Si nota che tutte le classificazioni seguono all'incirca lo stesso risultato, in altre parole, il solo stemming delle parole del dataset, non fornisce dei dati più facilmente classificabili.

Osserviamo ora i rapporti fra i risultati 'Falsi Positivi' e 'Falsi Negativi':

Si è voluto dare risalto al classificatore che forniva i risultati migliori, ovvero il Naive Bayes Multinomial. Osservando i grafici ottenuti si nota che la curva si comporta in maniera molto simile all'accuratezza del classificatore.

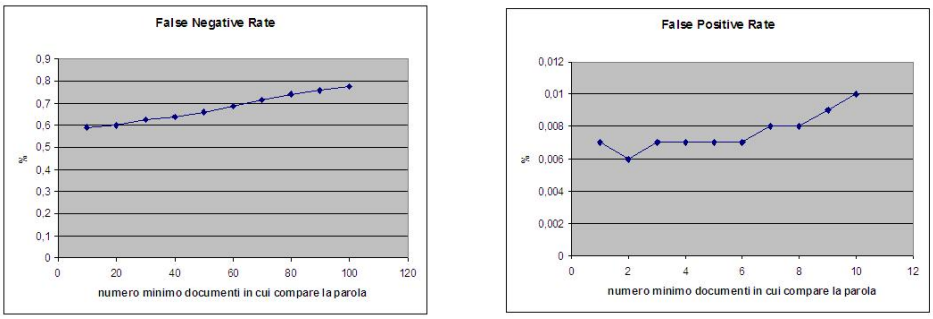

Figura 6.6: Falsi Positivi e Falsi Negativi

6.4 Un nuovo approccio: Metodo Statistico

A questo punto vediamo un nuovo approccio che chiameremo 'statistico', in cui ricercheremo le parole rappresentative sia delle categorie c_i , che dei documenti d_j.
Consideriamo tutte le parole p_k che compaiono in almeno X documenti d_j (dove X risulterà uno dei parametri modificabili per valutare la bontà di questo approccio) appartenenti alla categoria c_i.
Queste parole p_k le consideriamo le parole rappresentative della categoria c_i.
Consideriamo inoltre le parole p_l che compaiono in almeno Y documenti (ulteriore parametro di valutazione). Queste parole p_l le consideriamo rappresentative del documento d_j .
Attraverso queste due strutture dati costruite attraverso i documenti del set di train per ogni dataset, possiamo eseguire il controllo di corrispondenza con il set di test corrispondente.
Per la parte di train si tiene traccia di ciascuna di parole $p_k \in c_i$ e $p_l \in d_j$, e nel caso in cui il documento d_j corrisponde alla categoria c_i , allora il documento fa incrementare la capacità rappresentativa della coppia $(p_k - p_l)$ per la categoria c_i, in altre parole, se un documento d_j da classificare contiene questa coppia di parole allora verrà classificato come appartenente alla categoria c_i , in caso contrario, non viene classificato appartenente a quella categoria. Nella parte di test di fatto non si fa altro che verificare se le varie coppie abbiano più corrispondenze positive o negative.

6.4.1 Algoritmo

Il metodo statistico, utilizza un metodo piuttosto semplice per estrarre la struttura dati con cui fare la catalogazione: si devono estrarre i dati rilevanti, sia per le categorie che per i documenti (solo i documenti di train) e costruire un ADT composto dalla coppia di parole in analisi. Una volta costruita questa struttura il test non deve fare altro se non verificare se le coppie composte dalle parole delle categorie e le parole dei documenti di test, si siano presentate nel processo di train, per cui si tiene traccia delle coppie e se queste appartengono alla categoria giusta, viene incrementato un contatore per la classificazione giusta (classok), altrimenti viene incrementato il contatore per la classificazione errata (classno). Al termine di questo processo, le coppie di contatori (per ogni categoria e per ogni documento) che hanno il contatore classok maggiore di classno, allora sono classificati appartenenti alla categoria, altrimenti sono catalogati non appartenenti alla categoria. Presentiamo l'algoritmo schematizzato per il train:

for all $c_i \in C$ **do**
  for all $d_j \in D$ **do**
    for all $p_c \in c_i$ **do**
      for all $p_{dtrain} \in d_j$ **do**
        $train_k(c_i; d_j) + +$
      end for
    end for
  end for
end for

Questo algoritmo costruisce un ADT di dimensioni molto grandi, a seconda delle parole caricate. Si tratta di una matrice $Train$ tridimensionale:

- documenti
- categorie
- similarità

A ogni iterazione si aumenta un valore di questa matrice $Train(p_c; p_d; CHECK(c_i; d_j))$, dove $CHECK(c_i; d_j)$ rappresenta una funzione che restituisce il rapporto di similarità fra la categoria e il documento. Per problemi di capacità computazionale si è deciso di limitare le parole estratte alle sole parole che si presentavano almeno 7 volte, ma con il dataset

più piccolo è provato a costruire l'ADT più grande possibile, almeno per avere un'idea sulle dimensioni massime di questo ADT. I numeri osservati, nel dataset più piccolo sono stati:

- 69 categorie
- 528 documenti di train
- 2315 parole in media per ogni categoria
- 253 parole in media per ogni documento

Ottenendo un numero di cicli pari a:

$$cicli = 69 \times 528 \times \times 2315 \times 253 = 20 \times 10^9$$

Computazionalmente troppo grande per essere gestito in tempi ragionevoli e con risorse limitate. Quindi per tentativi sono stati cercati dei limiti tollerabili e si `è raggiunto un limite minimo di parole che sono ripetute almeno 5 volte in ogni documento.

L'algoritmo per il test è molto simile ma deve semplicemente verificare la presenza della coppia parola-categoria e parola-documento, generando due variabili per ogni categoria e per ogni documento, che rappresentano il numero di coppie di parole che corrispondono alla categoria e al documento, ovvero:

for all $c_i \in C$ **do**
  for all $d_j \in D$ **do**
    $OK(c_i; d_j) := 0$
    $NO(c_i; d_j) := 0$
    for all $p_c \in c_i$ **do**
      for all $p_{dtest} \in d_j$ **do**
        $OK+ = train(c_i; d_j, OK)$
        $NO+ = train(c_i; d_j, NO)$
      end for
    end for
    if $OK > NO$ **then**
      $prediction(c_i; d_j) = YES$
    else
      $prediction(c_i; d_j) = NO$
    end if

  end for
end for

Le stesse considerazioni sulla quantità di cicli da eseguire per questa parte di codice sono le medesime, ma risulterebbero comunque meno onerosa computazionalmente parlando, visto che si deve solo verificare la presenza della coppia $(p_c - p_{dtrain})$ che attraverso l'utilizzo di funzioni di hash risulta piuttosto veloce. Inoltre questi dati potranno essere direttamente inseriti in una tabella per mantenerli in maniera permanente.
Verranno quindi generate le coppie (c_i, d_j) a cui corrisponderanno 2 numeri (ok, no) che rappresenteranno quanti documenti d_j realmente appartengono alla categoria c_i e quanti documenti d_j non appartengono realmente alla categoria c_i e infine si determina la risposta $similar/dissimilar$ per la classificazione, ovvero un semplice controllo:

$$\begin{array}{lll} if\,(OK\,(c_i, d_j) > NO\,(c_i, d_j)) & then & prediction\,(c_i, d_j) := YES \\ & else & prediction\,(c_i, d_j) := NO \end{array}$$

questa risulta la predizione del nostro catalogatore statistico in base alle parole che compongono ogni documento e ogni categoria.

6.4.2 Risultati ottenuti con il metodo statistico

I risultati ottenuti con questo nuovo metodo possono essere considerati osservando la quantità di documenti classificati appartenenti a una determinata categoria, sul numero distinto di documenti diversi, nella migliore delle ipotesi il risultato ottimale sarebbe stato che un documento d_j appartenga a una sola categoria c_i.
Facendo alcuni esperimenti sul dataset più piccolo si ottiene che il risultato 'più interessante' risulta con un limite di minimo nell'intorno di 10 parole, ma comunque i risultati non sono particolarmente buoni:

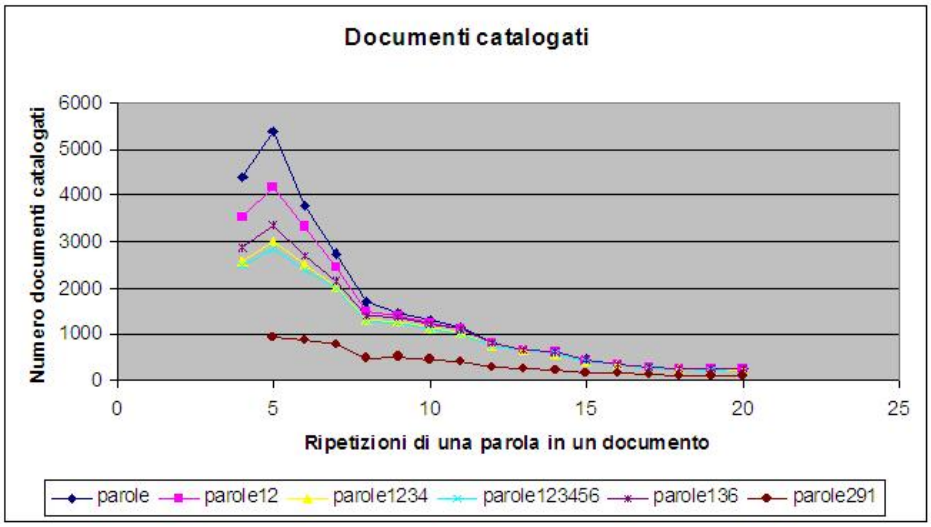

Figura 6.7: Documenti catalogati appartenenti a una certa categoria con diverse istanze di limite minimo di parole su Yahoo Ceci

si nota che il numero di documenti cala con l'aumentare del parametro di numero di ripetizioni in un docomento. Anche in questo caso si nota che lo stemming non apporta grandi migliorie.
Vogliamo vedere se comunque vengono catalogati documenti appartenenti a tutte le categorie, considerando le distinte categorie catalogate, i risultati sono:

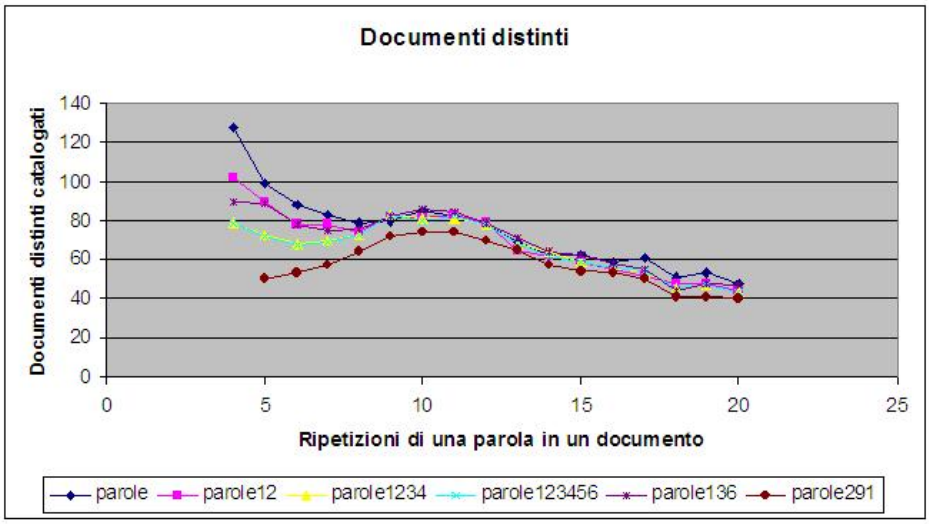

Figura 6.8: Documenti distinti catalogati appartenenti a una certa categoria

Notiamo che non vengono considerate tutte le categorie, ma se calcoliamo le percentuali di documenti catalogati bene, sul totale dei documenti considerati (visto che di fatto, limitando il numero minimo di ripetizioni di una parola in un documento possiamo togliere dei documenti dal dataset), si ottengono dei risultati sufficienti, ma che non variano profondamente con il variare dello stemming.

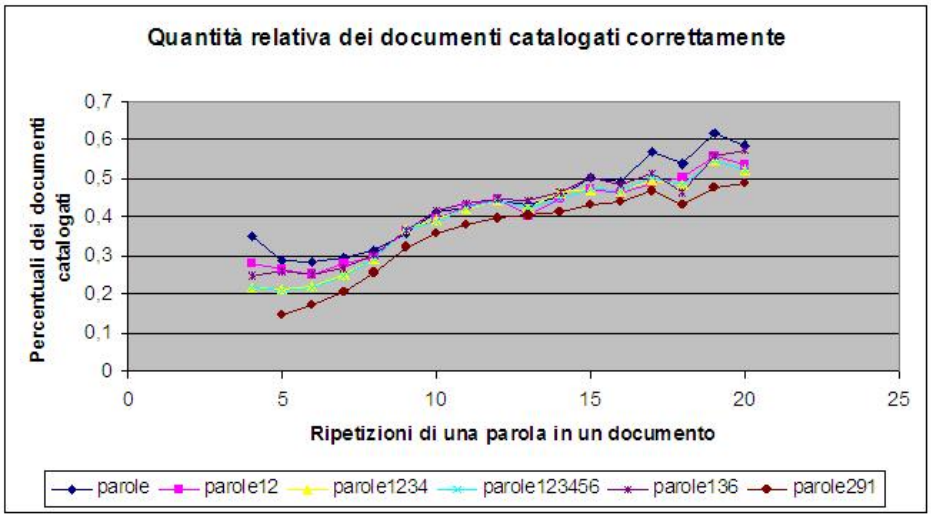

Figura 6.9: Percentuali di documenti distinti catalogati nelle categorie di appartenza

Chiaramente utilizzando dei limiti molto bassi per il numero minimo di parole, otteniamo dei buoni risultati, ma ricordiamoci che il dataset di Yahoo Ceci è molto piccolo e la grandezza del ADT per il train ha dimensioni molto grandi (come si abbiamo osservato nel precedente paragrafo).
Visto quindi il tempo necessario per i test, abbiamo applicato il metodo statistico anche ad altri dataset, più grandi, tenendo fissato il limite minimo delle parole a 10.
Dobbiamo tener presente che utilizzando un parametro stringente per rendere comunque le elaborazioni non eccessivamente pesanti, c'è la possibilità che dei documenti vengano eliminati, in quanto non contengono parole ripetute almeno 10 volte. Quindi vanno preliminarmente cercati tutti i distinti documenti di test che contengano almeno una parola ripetuta 10 volte. Inoltre si deve anche tener conto che i documenti possono essere catalogati in più categorie. I risultati ottenuti sono:

	dmozceci	ng20	reuters	ohsumed
parole	33,26	61,93	62,4	86,89
parole12	34,40	51,56	64,54	89,87
parole1234	34,01	50,85	63,82	88,88
parole123456	34,13	43,47	64,04	89,18
parole136	36,34	7,81	68,18	94,94
parole290	33,75	13,92	63,32	88,18
parole291	34,13	17,32	64,04	89,18

Si può notare che non in tutti i casi il metodo ha dei risultati validi, ma questo probabilmente deriva dal fatto che in metodo di estrazione dei dati di train non è completamente casuale. Osserviamo ora i rapporti fra i risultati 'Falsi Positivi' e 'Falsi Negativi', otteniamo, osservando Yahoo Ceci:

Capitolo 7

Conclusioni

In questa tesi sono stati applicati due algoritmi di stemming per la preelaborazione di documenti allo scopo di ridurne i termini alla propria radice sua radice, in generale diversa dalla radice morfologica.
Dagli esperimenti effettuati è emerso che gli algoritmi tradizionali di classificazione applicati a documenti trasformati mediante stemming parziale e completo non producono miglioramenti di accuratezza sui data set impiegati in questa tesi.
Nella tesi è stato sviluppato un nuovo metodo di classificazione basato sul calcolo di associazioni statistiche tra coppie di parole di documenti e prototipi di documenti rappresentativi di insiemi di documenti che trattano e che non trattano gli stessi temi. Il processo di stemming ha permesso di ridurre il numero di parole distinte e la loro varianza rispetto ai data set originali.
Dagli esperimenti emerge che il nuovo metodo di classificazione offre risultati promettenti, benché la versione prototipale implementata in questa tesi abbia margini di miglioramento anche in termini di efficienza computazionale e di storage. Il nuovo metodo produce i risultati migliori senza l'impiego degli algoritmi di stemming, questo significa che le parti eliminate nelle parole dal processo di stemming, pur essendo delle parti del testo 'invarianti', forniscono evidentemente un guadagno informativo nell'individuazione dell'argomento prevalente di ogni documento.
Il metodo sviluppato può essere esteso in diverse direzioni, per quanto riguarda l'efficacia una possibile estensione consiste nel generalizzare il matching esatto tra parole ad un matching mediante ontologie, mentre l'efficienza

computazionale pu ò essere migliorata introducendo indicizzazioni dei dati parzialmente gestite in memoria RAM invece che interamente in memoria secondaria su disco.

Bibliografia

[1] D. Amit, Yali; Geman. Shape quantization and recognition with randomized trees. Neural Computation, 1997.

[2] P. Andritsos. Limbo: Scalable clustering of categorical data. 2004.

[3] P. Anick. Exploiting clustering and phrases for context-based information retrieval. 1997.

[4] R. A. Baeza-Yates. Modern information retrieval - the concepts and technology behind search. second edition. 2011.

[5] L. Baker. Distributional clustering ofwords for text classification. 1998.

[6] R. Bekkerman. On feature distributional clustering for text categorization. 2001.

[7] S. Bolasco. Statistica testuale e text mining: alcuni paradigmi applicativi. In *Quaderni di Statistica*. Università di Roma, 2005.

[8] S. Bolasco. Statistica testuale e text mining: alcuni paradigmi applicativi. In *Quaderni di Statistica*. Università di Roma, 2005.

[9] A. F. Brown. Normal and reverse word list. In *AF 49*. Department of Linguistics, Philadelphia, 1963.

[10] R. Busa. Fondamenti di informatica linguistica,. In *Vita e Pensiero*. Istituto Geografico De Agostini, 1987.

[11] V. Ceppellini. Dizionario grammaticale. In *Dizionario Grammaticale*. Istituto Geografico De Agostini, 1978.

[12] W. B. Croft. Clustering large files of documents using the single-link method. 1977.

[13] D. Cutting. Scatter/gather: A cluster-based approach to browsing large document collections. 1992.

[14] D. Cutting. Constant interaction-time scatter/gather browsing of large document collections. 1993.

[15] M. Dash. Selection for clustering. 1997.

[16] S. Deerwester. Indexing by latent semantic analysis. 1990.

[17] I. Dhillon. Concept decompositions for large sparse data using clustering. 2001.

[18] H. Fang. A formal study of information retrieval heuristics. 2004.

[19] D. Fisher. Knowledge acquisition via incremental conceptual clustering. 1987.

[20] J. H. Gennari. Models of incremental concept formation. 1989.

[21] D. Gibson. Clustering categorical data: An approach based on dynamical systems. 1998.

[22] L. Giuliano. L'analisi automatica dei dati testuali. In *L'analisi automatica dei dati testuali*. Led on Line, 2004.

[23] S. Guha. Cure: An efficient clustering algorithm for large databases. 1998.

[24] S. Guha. Rock: a robust clustering algorithm for categorical attributes. 1999.

[25] T. K. Ho. Random decision forest. Proceedings of the 3rd International Conference on Document Analysis and Recognition, 1995.

[26] T. K. Ho. The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1995.

[27] T. Hofmann. Probabilistic latent semantic indexing. 1999.

[28] http://www.lemurproject.

[29] A. Jain. Algorithms for clustering data. In *Prentice Hall*. Englewood Cliffs, 1998.

[30] N. Jardine. The use of hierarchical clustering in information retrieval. 1971.

[31] I. T. Jolliffee. Principal component analysis. 2002.

[32] V. Lejnieks. The system of english suffixes. In *Linguistics 29*, 1967.

[33] T. Liu. An evaluation on feature selection for text clustering. 2003.

[34] J. B. Lovins. Development of a stemming algorithm*. In *Mechanical Translation and Computational Linguistics,*. Massachusetts Institute of Technology, 1968.

[35] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval classification and clustering. 1996.

[36] D. Metzler. Similarity measures for short segments of text. 2007.

[37] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. 1983.

[38] F. Murtagh. Complexities of hierarchical clustering algorithms: State of the art. 1984.

[39] R. Ng. Efficient and effective clustering methods for spatial data mining. 1994.

[40] K. Nigam. Learning to classify text from labeled and unlabeled documents. 1998.

[41] S. Nobile. L'approccio lessicometrico. In *L'analisi del contenuto*. Led on Line, 2006.

[42] M. F. Porter. An algorithm for suffix stripping. In *Prentice Hall*. Cambridge, 1998.

[43] J. R. Quinlan. C4.5: Programs for machine learning. Morgan Kaufmann Publishers, 1993.

[44] C. J. Rijsbergen. Document clustering: An evaluation of some experiments with the cranfield 1400 collection. 1975.

[45] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. 1994.

[46] M. Sahami. A web-based kernel function for measuring the similarity of short text snippets. 2006.

[47] N. Sahoo. Incremental hierarchical clustering of text documents. 2006.

[48] G. Salton. An introduction to modern information retrieval. 1983.

[49] G. Salton. Term weighting approaches in automatic text retrieval. 1988.

[50] H. Schutze. Projections for efficient document clustering. 1997.

[51] A. Singhal. Pivoted document length normalization. 1996.

[52] N. Slonim. Document clustering using word clusters via the information bottleneck method. 2000.

[53] M. Tartara. Elaborazione del linguaggio naturale. http://www.windizio.altervista.org/appunti/.

[54] C. J. van Rijsbergen. Information retrieval. 1975.

[55] E. M. Voorhees. Implementing agglomerative hierarchical clustering for use in information retrieval. July 1986.

[56] J. Wilbur. The automatic identification of stopwords. 1992.

[57] P. Willett. Recent trends in hierarchical document clustering: A critical review. 1988.

[58] W. Xu. Document clustering based on nonnegative matrix factorization. 2003.

[59] Y. Yang. A comparative study on feature selection in text categorization. 1995.

[60] Y. Yang. Noise reduction in a statistical approach to text categorization. 1995.

[61] C. Zhai. Statistical language models for information retrieval (synthesis lectures on human language technologies). 2008.

[62] T. Zhang. Birch: An efficient data clustering method for very large databases. 1996.

[63] Y. Zhao. Evaluation of hierarchical clustering algorithms for document data set. 2002.

Elenco delle figure

Elenco delle tabelle