UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

RELAZIONE

Creazione di formari mono-lingua a partire da lessici di frequenza

Candidato: *Giulia Guidi*

Relatore: *Prof. Francesco Romani*

Correlatore: *Prof. Alessandro Lenci*

Anno Accademico 2015-2016

Indice

Introduzione

Per la sperimentazione si sono usati i dump di Wikipedia che permettono di disporre di svariati gigabyte di testo solo apparentemente mono-lingue in quanto, essendo Wikipedia un'enciclopedia, è molto frequente la presenza di parole straniere, in particolare nomi di luoghi e persone.

Il lavoro di tirocinio svolto presso il dipartimento di Informatica dell'Università di Pisa ha costituito la base per la preparazione del materiale utilizzato ed è consistito nell'estrazione delle parole dai dump di Wikipedia per le lingue italiana, inglese, francese, tedesca e spagnola.

Il sistema di estrazione è indipendente dalla lingue richiedendo solo la definizione dell'alfabeto di riferimento e può essere facilmente esteso per trattare altre lingue con alfabeto latino o lingue con alfabeti diversi.

Nella tesi è descritto dapprima il lavoro di estrazione delle parole poi dopo alcuni richiami di linguistica computazionale e di teoria dell'informazione viene definito un indice di tipicità legato alla probabilità di occorrenza di una parola di lunghezza fissata nel quadro di un modello markoviano. L'uso congiunto della frequenza delle parole nel campione (che ha una distribuzione di tipo zipfiano) e dell'indice di tipicità nel modello (che ha una distribuzione a campana) ha permesso di rilevare (pur in presenza di falsi positivi e falsi negativi) la presenza di errori e di parole straniere nei formari delle varie lingue.

Vengono anche dati cenni di come utilizzare queste informazioni allo scopo di individuare formari puliti mono-lingua.

1. I dati della lingua

La raccolta dei dati linguistici e la loro rappresentazione svolgono un ruolo cruciale per la conduzione di analisi, anche in vista del trattamento automatico del linguaggio. L'impiego di metodi statistici ha reso indispensabile una maggiore attenzione alla composizione del materiale linguistico e conseguentemente ha influito sull'attenzione rivolta alla raccolta dei dati stessi.

1.1 I Corpora

"Un corpus è una collezione di testi selezionati e organizzati secondo criteri specifici in modo da renderli funzionali a precise analisi linguistiche." (A. Lenci et al., Testo e Computer)

In linguistica computazionale i corpora rappresentano la principale fonte di dati. Il fattore che ha promosso la creazione e l'utilizzo di queste collezioni testuali è senza dubbio lo sviluppo della tecnologia, anche se la nozione di corpus risulta essere precedente l'avvento del computer. Tuttavia l'era informatica ha rivoluzionato la natura e il ruolo dei corpora, in quanto permette di immagazzinare dati in grandi quantità, permette di ottimizzarne l'analisi e la ricerca e permette di sviluppare modelli computazionali della lingua.

Il ruolo del computer è ad oggi di cruciale rilevanza nell'ambito della linguistica computazionale, tanto che il termine corpus è diventato di fatto sinonimo di "corpus elettronico", ovvero una raccolta di testi in formato elettronico.

1.2 Corpora: una fonte di dati ecologici

Parole, frasi ed enunciati sono i prodotti del linguaggio che vengono sottoposti ad analisi, questi elementi si definiscono dati linguistici, e in particolare si definiscono dati linguistici "ecologici" qualora si tratti di testi prodotti dai parlanti di una lingua.

I testi che compongono i corpora sono un insieme di prodotti del linguaggio e in qualità di dati ecologici evidenziano una naturalezza del contesto in cui sono ottenuti, poiché osservati e raccolti nel proprio "ambiente", il testo.

1.3 Il corpus in qualità di campione

Ogni corpus è il frutto di una precisa selezione e per sua natura si differenzia nella tipologia a seconda dei testi che vi sono contenuti. Esistono una serie di parametri per cui è possibile definire la tipologia dei corpora: generalità, modalità, cronologia, lingua, integrità dei testi e livello di codifica digitale sono gli elementi su cui si basano i linguisti per delinearne le caratteristiche.
Il grado di adeguatezza di un corpus in qualità di fonte di dati viene misurato sulla base dell'interazione di due fattori fondamentali, la sua dimensione quantitativa e la sua composizione qualitativa, ovvero la sua tipologia. La validità in veste di campione di un corpus si misura secondo quella che in linguistica computazionale è la rappresentatività del campione stesso: un corpus è rappresentativo di una popolazione se tiene traccia dell'intero ambito di variabilità dei tratti e delle proprietà della lingua. Rappresentatività e variabilità però non sono i soli principi secondo cui viene valutata la validità di un campione, anche il bilanciamento è un fattore essenziale. La capacità di tenere traccia dello spettro di variabilità di una lingua è misurata dal bilanciamento del corpus. Un corpus si dice bilanciato se presenta una quantità consistente di testi selezionati per le diverse tipologie individuate nella popolazione. È importante tenere presente che la rappresentatività di un campione ha dei limiti intrinseci, in quanto un corpus è una collezione di testi finita che ha il compito di tenere traccia delle infinite situazioni in cui si possono registrare dati linguistici. L'affidabilità di un corpus come fonte di dati dipende dunque dalla sua capacità di offrire un modello fedele del lessico e della grammatica di una lingua.

1.4 Corpus di addestramento

"Un corpus di addestramento, detto anche "training-corpus", è un corpus che ci permette di osservare i fenomeni linguistici ed offre la possibilità di costruire modelli a partire dagli eventi osservati in esso" (A. Lenci et al., Testo e Computer); in altri termini un corpus di addestramento non è altro che un corpus bilanciato e rappresentativo della lingua che viene utilizzato a scopi di analisi.

2. Wikipedia

Wikipedia è un'enciclopedia elettronica basata sul principio del "peer-to-peer", il cui contributo non si deve ad un comitato di redazione al vertice dell'organizzazione, ma è dato dagli stessi utenti che possono creare, modificare e aggiornare le pagine di cui la raccolta dispone.

Questa peculiarità comporta un fattore di rischio riguardo l'oggettività dei contenuti, ma permette di disporre di argomenti vastissimi e rapidissimi aggiornamenti.

Wikipedia è l'enciclopedia più utilizzata al mondo, è completamente gratuita e dispone di innumerevoli voci in diverse lingue; per lo sviluppo di questo lavoro si è rivelata particolarmente interessante soprattutto per la sua qualità di essere una base di conoscenze e per la possibilità di essere utilizzata in qualità di corpus da cui si è potuto estrarre parole in più lingue.

Ecco come si presenta una pagina Wikipedia:

Figura 1, esempio di una pagina Wikipedia.

2.1 Wikipedia: un training-corpus elettronico

Per sua definizione un corpus è una collezione di testi, un'enciclopedia elettronica come quella di Wikipedia dunque non può che soddisfare questo requisito.
Non tutti i corpus presentano le caratteristiche appropriate per essere considerati in qualità di campione, ma osservando le peculiarità di Wikipedia ne possiamo comprendere le potenzialità.
Trattandosi di un'enciclopedia i testi che vi sono raccolti offrono, oltre alle varietà linguistiche con un basso grado di generalità, anche le varietà del linguaggio che fanno parte di un ambito tecnico-specialistico. Wikipedia in qualità di corpus risulta bilanciato e rappresentativo.
Inoltre Wikipedia è disponibile in numerose lingue sia europee che extra-europee e perfino in alcuni dialetti.
La scelta di Wikipedia in veste di training-corpus è dovuta all'insieme delle proprietà appena elencate, rivelatesi ottimali ai fini di ottenere formari con lessici di frequenza privi di parole appartenenti a lingue straniere.

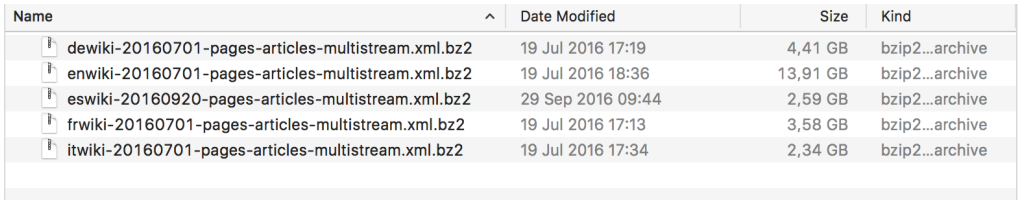
2.2 Struttura interna

Da un punto di vista tecnico Wikipedia è un sito web che poggia sul sistema "wiki", un software collaborativo, le cui pagine sono contenute all'interno di un database accessibile. Le informazioni relative alle pagine dell'enciclopedia sono fruibili e scaricabili dagli utenti nel formato XML (eXtensible Mark-Up Language), un metalinguaggio basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo.
Wikipedia mette a disposizione degli utenti del web le copie dei database realizzate a partire dagli aggiornamenti periodici dei propri archivi, i *dump*.
I dump Wikipedia sono disponibili sulla pagina della stessa enciclopedia alla voce "Wikipedia Database download" e possono essere scaricati dalla pagina "Wikimedia Download" (`https://dumps.wikimedia.org/backup-index.html`).
Per condurre gli studi relativi a questa tesi sono state trattate alcune delle lingue europee, in particolare l'inglese, l'italiano, il francese, il tedesco alle quali

successivamente si è deciso di aggiungere anche lo spagnolo.

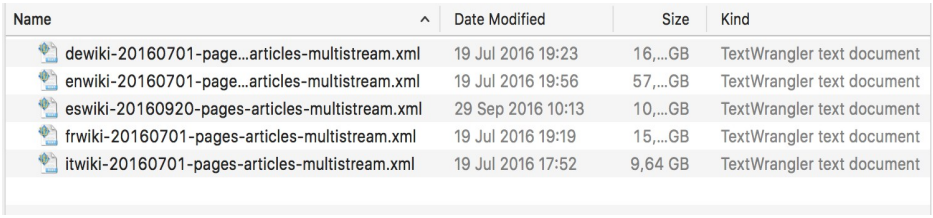
Per motivi di prestazione della rete i dump di Wikipedia sono stati scaricati presso il Dipartimento di Informatica dell'Università di Pisa.

Al momento del download i documenti ottenuti in formato bzip sono stati espansi e ne sono stati tratti documenti in formato XML.



Name	Date Modified	Size	Kind
dewiki-20160701-pages-articles-multistream.xml.bz2	19 Jul 2016 17:19	4,41 GB	bzip2…archive
enwiki-20160701-pages-articles-multistream.xml.bz2	19 Jul 2016 18:36	13,91 GB	bzip2…archive
eswiki-20160920-pages-articles-multistream.xml.bz2	29 Sep 2016 09:44	2,59 GB	bzip2…archive
frwiki-20160701-pages-articles-multistream.xml.bz2	19 Jul 2016 17:13	3,58 GB	bzip2…archive
itwiki-20160701-pages-articles-multistream.xml.bz2	19 Jul 2016 17:34	2,34 GB	bzip2…archive

Figura 2, file Wikipedia in formato bzip.



Name	Date Modified	Size	Kind
dewiki-20160701-page…articles-multistream.xml	19 Jul 2016 19:23	16,…GB	TextWrangler text document
enwiki-20160701-page…articles-multistream.xml	19 Jul 2016 19:56	57,…GB	TextWrangler text document
eswiki-20160920-pages-articles-multistream.xml	29 Sep 2016 10:13	10,…GB	TextWrangler text document
frwiki-20160701-pages-articles-multistream.xml	19 Jul 2016 19:19	15,…GB	TextWrangler text document
itwiki-20160701-pages-articles-multistream.xml	19 Jul 2016 17:52	9,64 GB	TextWrangler text document

Figura 3, espansione dei file Wikipedia in documenti XML.

Un esempio per la lingua italiana:

```
<page>

  <title>Armonium</title>
  <id>2</id>
  <timestamp>2008-06-22T21:48:55Z</timestamp>
  <username>Nemo bis</username>
  <comment>italiano</comment>
  <text xml:space="preserve">[[Immagine:Harmonium2.jpg|   thumb|right|300
px]]

  L''''armonium'''' (in francese, ''harmonium'') è uno [[strumenti
musicali|strumento musicale]] azionato con una [[tastiera (musica)|
tastiera]], detta manuale. Sono stati costruiti anche alcuni armonium con
due manuali.
```

```
  ==Armonium occidentale==
Come l'[[organo (musica)|organo]], l'armonium è utilizzato tipicamente in
[[chiesa (architettura)|chiesa]], per l'esecuzione di [[musica sacra]], ed
è fornito di pochi registri, quando addirittura in certi casi non ne
possiede nemmeno uno: il suo [[timbro (musica)|timbro]] è molto meno ricco
di quello organistico e così pure la sua estensione.

  ...

  ==Armonium indiano==
  {{S sezione}}

  == Voci correlate ==
  *[[Musica]]
  *[[Generi musicali]]</text>

</page>
```

2.3 L'estrazione del testo da Wikipedia

La procedura di estrazione del testo da Wikipedia è piuttosto complessa, specialmente per la presenza dei *template*, elementi che possono essere definiti come macro aspetti che permettono di referenziare gli spazi comuni del testo in una maniera parametrica efficiente. *"Il termine inglese template, letteralmente "sagoma" o "calco", in informatica indica un documento o un programma nel quale esistono spazi temporaneamente "bianchi" da riempire successivamente."* (Wikipedia, voce *Template*)

Il lavoro di estrazione per questa tesi è stato svolto grazie ad un tool pubblico già esistente, `WikiExtractor`, sviluppato presso il Dipartimento di Informatica dell'Università di Pisa dal Professor Giuseppe Attardi.

L'utilizzo del programma si è rivelato abbastanza agevole ed ha permesso di trasformare i file `XML` di Wikipedia in file `XML` molto più semplici, nei quali sono riportate le sole frasi del testo, frasi che hanno costituito il punto di partenza del lavoro di analisi condotto per la presente tesi.

WikiExtractor

Il tool in questione, `WikiExtractor`, è organizzato come una piattaforma che

utilizza l'enciclopedia di Wikipedia come fonte di documenti ed è disponibile tra i progetti presenti sul sito internet Medialab dell'Università di Pisa, alla voce *Wikipedia Extractor*. L'operazione messa in atto dal programma genera documenti di puro testo basati sui dump del database di Wikipedia. La funzione di estrazione è definita in Python e non richiede l'utilizzo di alcuna libreria aggiuntiva. Nel documento restituito in risultato è possibile identificare la suddivisione originale in pagine dell'enciclopedia; l'inizio di ogni unità è segnalato da un tag di apertura `<doc>`, e la fine dal rispettivo tag di chiusura `</doc>`.

Un esempio di testo estratto con `WikiExtractor`:

```
<doc id="2" url="http://it.wikipedia.org/wiki/Armonium">

Armonium.
L'armonium (in francese, "harmonium") è uno strumento musicale azionato con una tastiera, detta manuale. Sono stati costruiti anche alcuni armonium con due manuali.

Armonium occidentale.

Come l'organo, l'armonium è utilizzato tipicamente in chiesa, per l'esecuzione di musica sacra, ed è fornito di pochi registri, quando addirittura in certi
casi non ne possiede nemmeno uno: il suo timbro è molto meno ricco di quello organistico e così pure la sua estensione.

...

</doc>
```

Lo script produce in output una serie di documenti simili la cui dimensione massima è di circa 104 MB, ad eccezione dell'inglese, i cui file arrivano a dimensioni di 150 MB. Questa operazione ha avuto come scopo quello di presentare non più di 100 file per lingua.

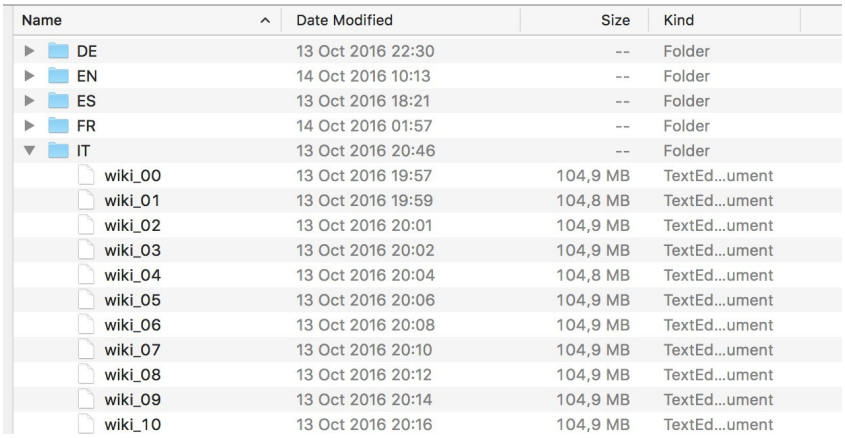


Figura 4, file per tutte le lingue organizzati in cartelle.

2.4 Lessici di frequenza

Per analizzare e confrontare nel dettaglio ciascuno dei lessici delle lingue di interesse è stato opportuno definire altri script, oggetto del percorso di tirocinio, mediante i quali i testi sono stati ulteriormente elaborati.

In seguito all'estrazione ottenuta con `WikiExtractor`, dagli stessi file, si è voluto ricavare un lessico di frequenza, pertanto è stato necessario eseguire una suddivisione del testo in token. A questo scopo un programma Python, `elab`, opera la normalizzazione del testo eliminando tutti i tag `XML` relativi alla gerarchia delle pagine dei documenti, tutti i caratteri non alfabetici presenti nel testo e le parole con lunghezza inferiore a tre caratteri.

All'interno del programma sono state utilizzate le *espressioni regolari*, sequenze di simboli che identificano particolari stringhe in un testo, e sono stati impiegati alcuni metodi appartenenti al *modulo* `string` di Python: `isalpha`, per controllare che le stringhe siano costituite solamente da caratteri alfabetici, `split`, per realizzare la suddivisione in token e `casefold`, che restituisce una copia della stringa dalla quale vengono eliminati tutti i caratteri maiuscoli e vengono rimosse tutte le distinzioni di casi, come ad esempio la lettera "ß" tedesca trasformata in "ss".

Il programma non conosce la lingua su cui lavora ed i file che ne risultano di conseguenza sono indipendenti da essa.

`elab` produce e salva due file: `raw.txt` e `raw.json`; entrambi contengono il lessico di frequenza del linguaggio esaminato, nel quale sono compresi tutti i caratteri alfabetici appartenenti a lingue straniere che si trovano nel corpus di partenza, come ad esempio l'arabo, il greco o il cinese.

Il formato `JSON`, acronimo di JavaScript Object Notation, è un formato indipendente dal linguaggio di programmazione utilizzato, ideale per lo scambio di dati. La versione `JSON` è utilizzata per le successive elaborazioni, mentre la versione `TXT` è impiegata al semplice scopo di debugging.

Ecco alcuni esempi per la lingua italiana:

Figura 5, parole italiane presenti in `raw.text`

Figura 6, parole arabe e coreane presenti in `raw.text`

Il programma `elab` è stato eseguito su una macchina iMac i7 @ 4 Ghz 32 Mb core memory 1600Mhz DDR3.

Nella tabella di seguito vengono illustrati, per ciascuna lingua, i tempi necessari ad `elab` per l'elaborazione, con le dimensioni dei relativi corpora.

Tempi di elaborazione

LINGUA	TEMPO	DIMENSIONE OUTPUT
IT	581 sec.	2.60 GB
ES	657 sec.	3.13 GB
FR	787 sec.	3.70 GB
DE	1188 sec.	5.37 GB
EN	2826 sec.	13.00 GB

Tabella 1, tempi di `elab`.

Ecco il cuore di `elab`[1]:

```
(...)

def elab(file, D):
    start = time()
    inp = codecs.open(file, "r", "utf8")
    for line in inp:
        line = line.strip()
        if len(line) > 3:
            line=re.sub(r"<.*>", " ", line)
            ba = list(line)
            for i in range(len(ba)):
                c = ba[i]
                if not c.isalpha():
                    ba[i] = ' '
            line = "".join(ba)
            words = line.split()
            for word in words:
                if len(word) > 3:
                    word = word.casefold()
                    D[word] = D.setdefault(word, 0) + 1
    t = int(time() - start + 0.5)
    print(file + "  " + str(t) + " sec.")

(...)
```

1 Per la versione completa del codice si consulti l'Appendice.

Dopo `elab` è stato definito un secondo programma, `filter`, anche questo scritto in Python con l'obiettivo di ottenere dai testi un corpus con cui lavorare per ottenere formari mono-lingua puliti. Il programma mira ad eliminare dalle liste di parole tutte quelle che agli scopi di analisi sono considerate inconsistenti: le parole troppo lunghe e quelle appartenenti ad alfabeti esotici e stranieri. Tutto ciò che viene eliminato dai testi di partenza viene comunque registrato su file separati che permettono di effettuare eventuali controlli anche a scopo di debugging.

Il programma crea tre liste in cui si inseriscono le parole a seconda delle condizioni che queste soddisfano: sono considerate *esotiche* le parole non appartenenti all'insieme prodotto dall'intersezione degli alfabeti italiano, inglese, francese, tedesco e spagnolo; sono classificate come *straniere* invece le parole che non appartengono all'insieme dell'alfabeto locale della lingua presa in esame ed infine sono considerate *lunghe* le parole che superano i 40 caratteri.

Più precisamente `filter` prende in ingresso i file `raw.json` prodotti da `elab` e a sua volta produce una serie di risultati: un file per le successive elaborazioni, `work.json`, un file a scopo di debugging, `work.txt`, un file contenente le parole che superano i quaranta caratteri di lunghezza, `long.txt`, un file per le parole esotiche, `esotic.txt`, ed un file per le parole straniere, `strange.txt`.

Ecco alcuni esempi di parole *esotiche*, *straniere* e *lunghe* rilevate nel file dell'italiano:

Figura 7, un esempio di termini esotici rilevati dalla versione italiana di Wikipedia

Figura 8, un esempio di termini stranieri,
sicuramente non italiani, ma contenuti nella versione italiana di Wikipedia

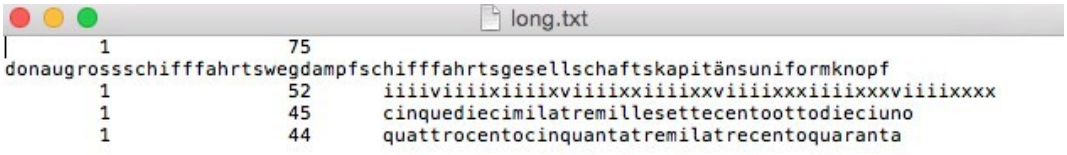


Figura 9, un esempio di parole molto lunghe contenute nella
versione italiana di Wikipedia

Da notare bene è che non tutte le parole *straniere* appartengono ad un linguaggio diverso dall'italiano, vi possono essere anche parole semplicemente errate.

Di seguito si riporta la parte centrale di `filter`[2] :

```
(...)

def filter(lang):
    start = time()
    long=[]
    esotic=[]
    strange=[]

    setAll = read(dir+"setAll.json")
    setc = read(dir+lang+"/setc.json")
    D = read(dir+lang+"/raw.json")

    for word in sorted(list(D.keys())):
        lw = len(word)
```

2 Per la versione completa del codice si consulti l'Appendice.

```
        fre = D[word]
        L[lw]=L.setdefault(lw, 0)+1
        F[fre]=F.setdefault(fre, 0)+1
        if not letter(word, setAll):
            esotic.append("{0:9d}\t{1:4d}".format(fre,lw)+"\t"+word)
            D.pop(word,None)
        elif lw>maxlen:
            long.append("{0:9d}\t{1:10d}".format(fre,lw)+"\t"+word)
            D.pop(word,None)
        elif not letter(word, setc):
            strange.append("{0:9d}\t{1:4d}".format(fre,lw)+"\t"+word)
            D.pop(word,None)


    writeMap(dir+lang+"/work.txt", D)
    dump(dir+lang+"/work.json", D)
    writeList(dir+lang+"/long.txt", long)
    writeList(dir+lang+"/esotic.txt", esotic)
    writeList(dir+lang+"/strange.txt", strange)
    writeMap(dir+lang+"/fre.txt", F)
    writeMap(dir+lang+"/len.txt", L)

(...)
```

Come si è detto precedentemente `filter` è un programma che lavora su intersezioni e disgiunzioni degli alfabeti, trattati in qualità di insiemi.

Pertanto è stato scritto un ulteriore script, `Chars`, che definisce liste di caratteri salvate come file JSON lette in ingresso dal programma `filter`.

Di seguito sono presentate le scelte per i vari linguaggi[3].

Italiano

Per l'italiano sono stati selezionati i seguenti caratteri facendo riferimento alla pagina *Alfabeto italiano* di *Wikipedia*:

```
setIT =
['a','à','b','c','d','e','è','é','f','g','h','i','ì','î','j','l','m','n','o'
,'ò','p','q','r','s','t','u','ù','v','z']
```

(https://it.wikipedia.org/wiki/Alfabeto_italiano).

Inglese

Per l'inglese si sono inseriti nell'insieme dei caratteri quelli elencati di seguito, in riferimento alle indicazioni ricavate da *Wikizionario* alla voce *Appendice alfabeti*:

3 Per la versione completa del codice si consulti l'Appendice.

```
setEN =
['a','b','c','d','e','f','g','h','k','i','j','l','m','n','o','p','q','r','s'
,'t','u','v','w','x','y','z']
```
(https://it.wiktionary.org/wiki/Appendice:Alfabeti).

Francese

Per il francese l'insieme dei caratteri è stato delineato in base alle informazioni riportate nella pagina *Alfabeto francese* di *Wikipedia*:

```
setFR=
['a','à','â','æ','b','c','ç','d','e','ë','è','é','ê','f','g','h','k','i','ï'
,'î','j','l','m','n','o','ô','œ','p','q','r','s','t','u','ù','ü','û','v','w'
,'x','y','ÿ','z']
```
(https://it.wikipedia.org/wiki/Alfabeto_francese).

Tedesco

Per il tedesco si è fatto riferimento al sistema ortografico descritto nella pagina *Sistema ortografico tedesco* di *Wikizionario*:

```
setDE=
['a','ä','b','c','d','e','f','g','h','i','j','k','l','m','n','o','ö','p','q'
,'r','s','t','u','ü','v','w','x','y','z']
```
(https://it.wiktionary.org/wiki/Appendice:Alfabeti).

Spagnolo

Per lo spagnolo si è definita una lista di caratteri sulla base del contenuto della pagina *Lingua spagnola* di *Wikipedia*:

```
setES=
['a','à','b','c','d','e','è','é','f','g','h','i','ì','j','l','m','n','ñ','o'
,'ò','p','q','r','s','t','u','ù','ü','v','w','x','y','z']
```
(https://it.wikipedia.org/wiki/Lingua_spagnola).

Alfabeto cumulativo

L'alfabeto cumulativo, contenente i caratteri di tutti gli alfabeti per le cinque lingue prese in esame è stato prodotto dall'intersezione degli insieme sopra elencati:

```
setAll=
['a','à','ä','â','á','æ','b','c','ç','d','e','ë','è','é','ê','f','g','h','k'
,'i','ì','í','ï','î','j','l','m','n','ñ','o','ò','ó','ö','ô','œ','p','q','r'
,'s','t','u','ù','ú','ü','û','v','w','x','y','ÿ','z']
```

3. Richiami di Teoria dell'informazione

"La teoria dell'informazione è una disciplina dell'informatica e delle telecomunicazioni che ha come scopo la quantificazione della quantità di dati (ossia di informazione) in relazione alla loro memorizzazione o trasmissione su un canale in modo affidabile". (Wikipedia, voce *Teoria dell'informazione*)
In questo capitolo vengono presentate alcune importanti nozioni teoriche connesse agli studi condotti per questa tesi.

3.1 Il concetto di Entropia

"L'entropia è una media calcolata sulla distribuzione di probabilità di un insieme di eventi aleatori mutuamente esclusivi". (A. Lenci et al., *Testo e Computer*,2005)

Dato un esperimento X con probabilità $p(X) = \{p_1, p_2, \ldots, p_k\}$, con k finito maggiore di 1, si definisce incertezza ovvero **Entropia** dell'esperimento la funzione

$$H(X) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

H(X) si misura in *bit*.

Una proprietà importante dell'entropia è che essa corrisponde al limite inferiore alla possibilità di comprimere l'informazione, ovvero se si effettua un esperimento al secondo non è possibile tarsmettere i risultati dell'esperimento con meno di $H(X)$ *bit/s*.

3.2 Il teorema dell'Equipartizione Asintotica

Nell'ambito della teoria dell'informazione il teorema dell'Equipartizione Asintotica stabilisce una forte proprietà dei risultati di un esperimento ripetuto un numero di volte che tende all'infinito.
È ben noto dalla Legge dei Grandi Numeri che se si ripete un esperimento finito un

numero di volte che tende all'infinito la frequenza relativa delle occorrenze delle varie possibili uscite tende alle probabilità delle stesse; questa legge può essere usata sia per prevedere (in senso statistico) i risultati dell'esperimento sia per calcolare empiricamente le probabilità a posteriori di un esperimento di cui non si conoscono le proprietà teoriche.

Il teorema dell'equipartizione asintotica, che si può dimostrare con tecniche standard di calcolo delle probabilità, lega i risultati dell'esperimento all'entropia dello stesso nel seguente modo.

Qualunque sia l'esperimento considerato, fissato un numero reale ε piccolo a piacere esiste un valore n_0 tale che se si ripete l'esperimento per n volte con $n>n_0$, le sequenze di risultati che si ottengono sono suddivisibili in due insiemi;

- un insieme detto di sequenze non **tipiche** con probabilità totale minore di ε
- un insieme detto di sequenze **tipiche** con probabilità totale maggiore di 1-ε.
- La probabilità di una sequenza tipica è dell'ordine di

$$2^{-nH(X)+O(\sqrt{n})}$$

- il numero delle sequenze tipiche è

$$2^{n\left(H(X)+\delta_n\right)} \quad \text{con} \quad \lim_{n\to\infty}\delta_n = 0$$

È interessante studiare cosa accade alle sequenze "tipiche" per valori di n abbastanza piccoli. (R. Ash, *Information and theory*,1965)

Noi non useremo il teorema dell'equipartizione asintotica nella sua forma originale, ma il richiamo di questo risultato dà un'idea di quello ci aspettiamo dalle uscite delle parole di una certa lingua se si immagina che la produzione di queste stesse parole sia un evento casuale. In particolare immaginiamo che data una lingua e fissato un n non grande, tipicamente compreso tra 4 e 30, le parole di lunghezza n si suddividono grosso modo in due categorie, una di parole tipiche che hanno "l'assonanza" propria di quella lingua e una categoria di parole non tipiche che hanno probabilità molto bassa di uscire. Poiché n non tende all'infinito ci aspettiamo che la distribuzione

delle sequenze “tipiche” rappresentata graficamente sia una curva a campana.

3.3 Un esempio

Supponiamo ad esempio di avere una moneta truccata che produce **testa** con probabilità p=0.7 e **croce** con probabilità 1–p=0.3; la probabilità di avere esattamente k teste in n tentativi vale

$$\binom{n}{k} p^k (1-p)^{n-k}$$

Questo valore è il risultato di due fattori:

$$\binom{n}{k}$$

che conta il numero di sequenze lunghe n con k teste;

$$p^k (1-p)^{n-k}$$

che dà la probabilità di ognuna di esse.

Il primo fattore è massimo quando k è circa $n/2$ (Figura 10), il secondo fattore è massimo quando k ed n sono uguali, ovvero quando la sequenza più probabile è quella formata da n volte il simbolo più probabile (Figura 11).

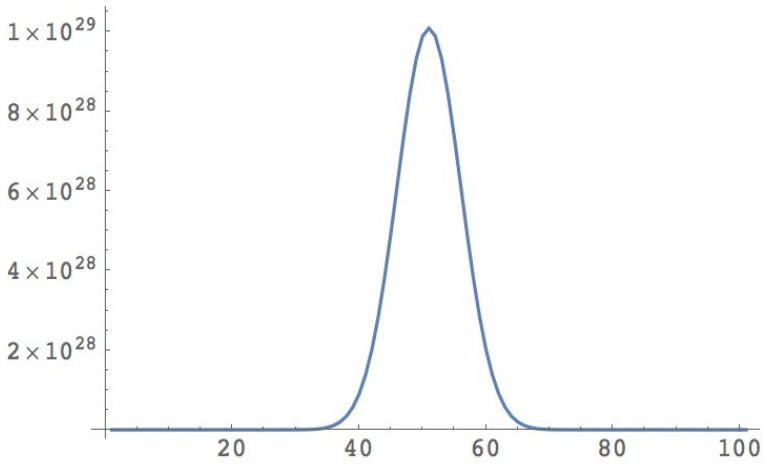


Figura 10, il grafico di $\binom{n}{k}$ in funzione di k per n=100.

Come si vede, il risultato prodotto dal grafico, è quello di una curva a campana.

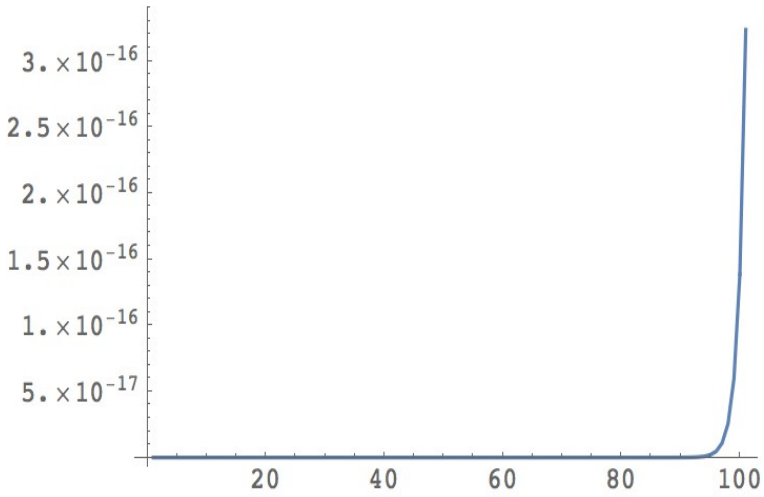


Figura 11, il grafico di $p^k(1-p)^{n-k}$ in funzione di k per P=0.7.

Di sequenze composta da sole teste ce n'è una soltanto, la tipologia di sequenze più probabile è quella con circa il 70% di teste ed il 30% di croci (Figura 12).

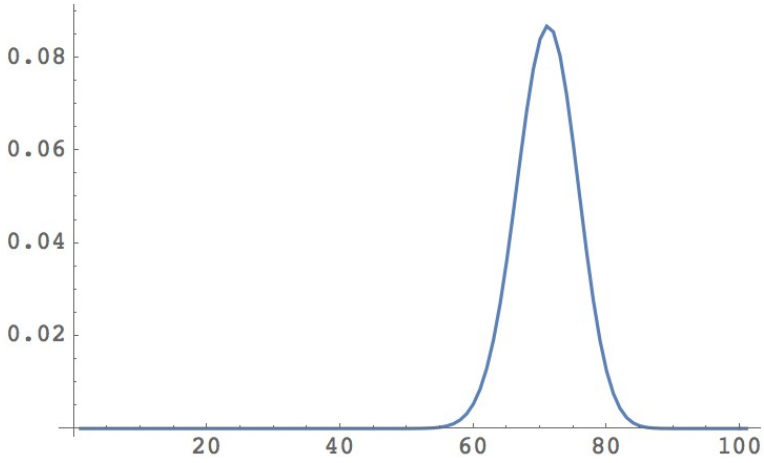


Figura 12, il grafico di $\binom{n}{k} p^k(1-p)^{n-k}$ in funzione di *k* per P=0.7.

Disponendo sulle ascisse la frequenza relativa delle teste e in ordinata i valori delle probabilità delle varie classi moltiplicate per *n* (Figura 11), le aree sottese alle curve risultanti sono tutte unitarie, ma con l'aumentare di *n* la campana si restringe e tende ad un picco di altezza infinita. Quindi asintoticamente, la probabilità che la sequenza che si presenta sia tipica tende ad 1.

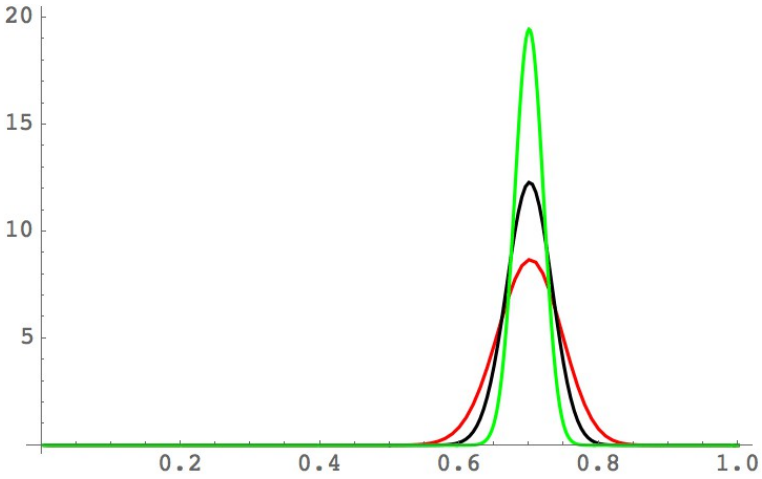


Figura 13, le curve relative a P=0.7, n=100, 200, 500 in scala normalizzata.

3.4 Sorgenti ed informazioni

"Lo scopo della teoria dell'informazione è quello di fornire metodi per comprimere al massimo l'informazione prodotta da una sorgente". (Wikipedia, voce *Teoria dell'informazione*)

Una sorgente, in informatica, è un generatore di informazioni che soddisfa determinate proprietà statistiche. Una sorgente si dice che non ha memoria se genera simboli tra loro indipendenti, se invece la generazione di un simbolo condiziona le generazioni successive, la sorgente si definisce con memoria.
Un linguaggio naturale può essere immaginato come una successione di informazioni emesse da una sorgente con memoria infinita. Ovviamente approssimare un linguaggio naturale con una sorgente dotata di proprietà semplici è un'operazione estremamente facile che fornisce risultati interessanti ma che non ha alcuna possibilità di catturare la pienezza di una lingua, in particolare un modello puramente sintattico non potrà in alcun modo generare frasi dotate di senso compiuto.
Di seguito si propongono alcuni esempi per la lingua inglese tratti da *Information theory and coding* di *Norman Abramson*, 1963.

Consideriamo un set di caratteri composto dalle ventisei lettere dell'alfabeto inglese e dal carattere di spaziatura. Il modello più semplice è quello a memoria zero con

simboli tutti equiprobabili.
L'entropia della sorgente risulta: $H(s) = \log_2 27 = 4.75$ *bit/simbolo*.
Ecco un esempio di sequenza di simboli emessa da questa sorgente:

ZEWRTZNSADXESYJRQY WGECIJJ
OBVKRBQPOZBYMBUAWVLBTQCNIKFMP KMVUUGBSAXHLHSIE

La frase non ha alcuna percettibile struttura e non è possibile nemmeno identificare un particolare linguaggio dal quale essa possa provenire.

Con una sorgente a memoria zero, i cui simboli vengono prodotti ciascuno in base alla distribuzione delle lettere dell'inglese, l'entropia della sorgente è: $H(s) = 4.03$ *bit/simbolo*.

AI NGAE ITF NNR ASAEV OIE BAINTHA HYROO POER
SETRYGAIETRWCO EDHUARU EU C FT NSREM DIY EESE F O SRIS R
UNNASHOR

Con una sorgente a memoria uno (ovvero le cui probabilità tengano conto del simbolo precedente) l'entropia scende a 3.32 *bit/simbolo* e un campione di emissione è il seguente:

URTESHETHING AD E AT FOULE ITHALIORT WACT D STE MINTSAN OLINS TWID OULY TE THIGHE CO YS TH HR UPAVIDE PAD CTAVED

Si noti già a questo punto come un po' di inglese inizi a farsi riconoscere, anche se potremmo riscontrare qualche problema con l'identificazione di queste sequenze come inglesi, piuttosto che francesi[4].

Con una sorgente a memoria due (entropia circa 3.1 *bit/simbolo*[5]) possiamo ottenere una approssimazione ancora più accurata, dalla quale iniziano ad emergere brevi sequenze appartenenti alla lingua inglese:

4 Si presti particolare attenzione alle stringhe *foule* e *upavide.*
5 Dalla Teoria dell'Informazione segue che l'Entropia di una sorgente è tanto più bassa quanto più ricca di "struttura" è la sorgente stessa. Da questi semplici esperimenti si vede già come arricchendo la sorgente l'Entropia diminuisce.

IANKS CAN OU ANG RLER THATTED OF TO SHOR OF TO
HAVEMEM A I MAND AND BUT WHISSITABLY THERVEREER
EIGHTS TAKILLIS TA

I risultati ottenuti non possono essere ancora qualificati come frasi di "lingua inglese", le parole tuttavia iniziano ad avere una lunghezza ragionevole ed una corretta proporzionalità tra il numero di vocali e quello di consonanti.

Questo meccanismo di approssimazione ovviamente può essere applicato per tutti i linguaggi. Di seguito si propongono alcuni esempi per il francese, il tedesco lo spagnolo ed il latino, tratti dal medesimo libro *Information theory and coding*.

Approssimazione del **francese** data da una sorgente a memoria due:

JOU MOUPLAS DE MONNERNAISSAINS DEME US VREH BRE TU DE TOUCHEUR DIMMERE LLES MAR ELAME RE A VER IL DOUVIENTS SO

Approssimazione del **tedesco** data da una sorgente a memoria due:

BET EREINER SOMMEIT SINACH GAN TURHATTER AUM WIE BEST ALLIENDER TAUSSICHELLE LAUFURCHT ER BLEINDESEIT UBER KONN

Approssimazione dello **spagnolo** data da una sorgente a memoria due:

RAMA DE LLA EL GUIA IMO SUS CONDIAS SU E UNCONDANDADO DEA MARE TO BUERBALIA NUE Y HERARSIN DE SE SUS SUPAROCEDA

Approssimazione del **latino** data da una sorgente a memoria due:

ET LIGERCUM SITECI LIBEMUS ACERELEN TE VICAESCERUM PE NON SUM MINUS UTERNE UT IN ARION POPOMIN SE INQUENEQUE IRA

È interessante notare come già con una approssimazione con memoria due, ovvero in cui ogni lettera dipende dalle due precedenti, sia possibile ottenere stringhe che pur essendo senza senso in qualche modo catturano un germe di struttura lessicale sufficiente a riconoscere la lingua che ha fornito i dati statistici da parte di uno

studioso che abbia una minima conoscenza di quella lingua. In altre parole è altamente probabile che un europeo sappia riconoscere, ad esempio, lo spagnolo da una stringa corrispondente, mentre ovviamente una persona non specializzata saprà riconoscere una traslitterazione dell'urdu[6] da una traslitterazione tagalog[7].

6 Lingua indoeuropea del gruppo delle lingue indo-iraniche.
7 Tra le principali delle lingue parlate nelle Filippine, è la più diffusa.

4. Indice di tipicità delle parole di un lessico di frequenza

Lo scopo di questa tesi è tentare di stabilire un procedimento automatico per selezionare in un elenco di parole l'appartenenza delle medesime alle varie lingue. L'idea è quella di sfruttare proprietà statistiche, come le catene di Markov, per stimare in qualche modo la "tipicità" di una parola.

Il nostro uso del concetto di sequenza tipica è estremamente approssimato: le parole prese in considerazione hanno una lunghezza piccola, il teorema sopra citato invece vale per lunghezze che tendono all'infinito. Come abbiamo visto nel caso di lunghezze piccole la distribuzione tipica tende ad essere una curva allargata (detta anche curva a campana).

In base al teorema dell'equipartizione asintotica, per un n grande, se la lunghezza tende all'infinito la probabilità delle sequenze tende a $p_{typ} = 2^{-n\,H(x)}$, quindi il suo logaritmo tende ad essere $\log p_{typ} = -nH(x)$ e $\frac{-\log ptyp}{n} = H(x)$

Ci si aspetta dunque, per un n non grande, una distribuzione intorno ad $H(x)$ che si comporti come una curva con andamento tipo campana.

Questo valore, calcolato su una singola parola, può dare un'idea di quanto quella parola sia effettivamente tipica nei confronti della lingua data e può fornire un indice che segnala o meno l'appartenenza di quella parola alla lingua considerata.

Lavorando in questo modo contemporaneamente su tutti i dizionari per cui sono state calcolate le frequenze sarà possibile trovare ad esempio le parole italiane presenti nel francese o le parole spagnolo nel tedesco.

Il modello considerato tiene traccia delle coppie e delle triple, ma non si preoccupa delle proprietà statistiche delle parole generate. Il confronto per stabilire la lingua di appartenenza di una parola viene effettuato su stringhe con lunghezza uguale, poiché non interessa allo scopo di questa tesi il confronto di parole di lunghezza diversa. Per ogni valore di n fissato si evitano quindi problemi relativi alla modalità di stima e generazione della lunghezza o problemi riguardanti la lunghezza media delle stringhe e delle possibili distribuzioni all'interno di ciascuna lingua.

Una volta fissata la dimensione di n non sarà necessario neppure il confronto in merito alla tipicità tra parole di lunghezze diverse e questo aspetto semplifica l'analisi anche dal punto di vista computazionale.

4.1 Probabilità dell'appartenenza al modello di una stringa di lunghezza *n*

Sfruttando la conoscenza della probabilità delle occorrenze delle coppie e delle triple ottenuta da un'analisi statistica da un grosso formario di frequenza, è possibile tentare di stimare la probabilità che una certa sequenza di n caratteri possa essere generata da una sorgente che ha quelle probabilità delle coppie e delle triple. Poiché noi ci restringiamo a considerare parole di lunghezza n, detto A l'alfabeto considerato e detto L_n l'insieme delle stringhe di lunghezza n che appartengono al linguaggio generato dalla sorgente associata al nostro modello, ci interessa studiare la probabilità che una stringa di lunghezza n appartenga ad L_n. Si ha ovviamente:

$$\sum_{s \in A^n} p(s \in L_n) = 1$$

Indicando con "_" il separatore tra le parole generate dal modello definiamo le seguenti quantità:

la probabilità che la coppia $x_1\ x_2$ stia all'inizio di una parola di L

$$p_{CI}(x_1, x_2) = p(\_\ x_1, x_2)$$

la probabilità che la coppia $x_1\ x_2$ stia alla fine di una parola L

$$p_{CF}(x_1, x_2) = p(\ x_1, x_2, \_)$$

la probabilità che la coppia $x_1\ x_2$ sia in una parola di L

$$p_C(x_1, x_2) = p(\ x_1, x_2)$$

la probabilità che la tripla $x_1\ x_2\ x_3$ sia in una parola di L

$$p_T(x_1, x_2, x_3) = p(x_1, x_2, x_3)$$

A questo punto la probabilità che una stringa $x_1,.., x_n$ sia generata dal nostro modello si può scrivere

$$p(x_1,...,x_n) = p(\_,x_1,x_2)p(x_1,x_2,x_3/x_1,x_2)p(x_2,x_3,x_4/x_2,x_3)...p(x_{n-1},x_n,\_/x_{n-1},x_n)$$

ovvero, più precisamente

$$p(x_1,x_2,...,x_n) = p(\_,x_1,x_2)\left(\prod_{i=3}^{n} p(x_i/x_{i-2},x_{i-1})\right)p(\_/x_{n-1},x_n)$$

Ricordando che dal teorema della probabilità condizionata discende (Wikipedia, voce *Probabilità condizionata*)

$$p(x|y)=\frac{p(x,y)}{p(y)}$$

applicando le definizioni viste sopra si ottiene

$$p(x_1,x_2,...,x_n) = p_{CI}(x_1,x_2)\left(\prod_{i=3}^{n} \frac{p_T(x_{i-2},x_{i-1},x_i)}{p_C(x_{i-2},x_{i-1})}\right)\frac{p_{CF}(x_{n-1},x_n)}{p_C(x_{n-1},x_n)}$$

Una verifica superficiale della validità di questa formula può essere il fatto che qualora tutte le probabilità siano valori costanti ogni coppia avrebbe una probabilità $p(k^{-2})$, ogni tripla probabilità $p(k^{-3})$, e ogni stringa probabilità $p(k^{-n})$ e otterremmo la relazione:

$$p(x_1,x_2,...,x_n) = k^{-2}\left(\prod_{i=3}^{n} \frac{k^{-3}}{k^{-2}}\right) = k^{-n}$$

Questo indica che non sono stati commessi errori grossolani di normalizzazione.

4.2 Calcolo delle frequenze e delle probabilità

Le probabilità delle coppie e delle triple, ovviamente, non sono note a priori e per ciascuno dei linguaggi considerati sono state calcolate su base statistica attraverso il programma Python, `trainer`.

Questo programma legge il file `work.json` contenente il formario con tutte le parole e le relative frequenze e tiene traccia della somma delle frequenze delle

coppie, delle coppie iniziali, delle coppie finali e delle triple. I valori delle frequenze vengono convertiti da interi a reali tramite la funzione `float` e sono utilizzate per il calcolo della probabilità delle rispettive coppie, coppie iniziali, coppie finali e triple.

Ecco il "cuore" del programma[8]:

```
(...)

minfreq=10

def trainer(lang):
    start = time()

    CI = {}
    C  = {}
    CF = {}
    T  = {}

    sCI = 0
    sCF = 0
    sC  = 0
    sT  = 0

    D = read(dir+lang+"/work.json")

    for word in D.keys():
        fre = D[word]
        if fre >= minfreq:
            lw = len(word)
            sub = word[0:2]
            CI[sub]=CI.setdefault(sub, 0)+fre
            sCI+=fre
            for l in range(lw-2):
                sub = word[l:l+2]
                C[sub]=C.setdefault(sub, 0)+fre
                sC+=fre
                sub = word[l:l+3]
                T[sub]=T.setdefault(sub, 0)+fre
                sT+=fre
            sub = word[-2:]
            CF[sub]=CF.setdefault(sub, 0)+fre
            sCF+=fre
            C[sub]=C.setdefault(sub, 0)+fre
            sC+=fre

    sC=float(sC)
    sCI=float(sCI)
    sCF=float(sCF)
    sT=float(sT)

    for s in CI.keys():
        CI[s]=CI[s]/sCI
    for s in CF.keys():
        CF[s]=CF[s]/sCF
    for s in C.keys():
        C[s]=C[s]/sC
    for s in T.keys():
        T[s]=T[s]/sT

    dump(dir+lang+"/CI.json", CI)
    dump(dir+lang+"/CF.json", CF)
    dump(dir+lang+"/T.json", T)
    dump(dir+lang+"/C.json", C)
    writeMap(dir+lang+"/CI.txt", CI)
    writeMap(dir+lang+"/CF.txt", CF)
    writeMap(dir+lang+"/T.txt", T)
```

8 Per la versione completa di `trainer` si consulti l'Appendice

```
    writeMap(dir+lang+"/C.txt", C)

    t = int(time()-start+0.5)
    print(lang+"  "+str(t)+" sec.")

trainer("IT")
trainer("ES")
trainer("FR")
trainer("DE")
trainer("EN")
```

`trainer` produce e salva file in due diversi formati, `JSON`, utilizzato per le successive elaborazioni, e `TXT`, impiegato a scopo di debugging.

Ecco alcuni esempi per la lingua italiana:

Figura 14, probabilità delle coppie per l'italiano

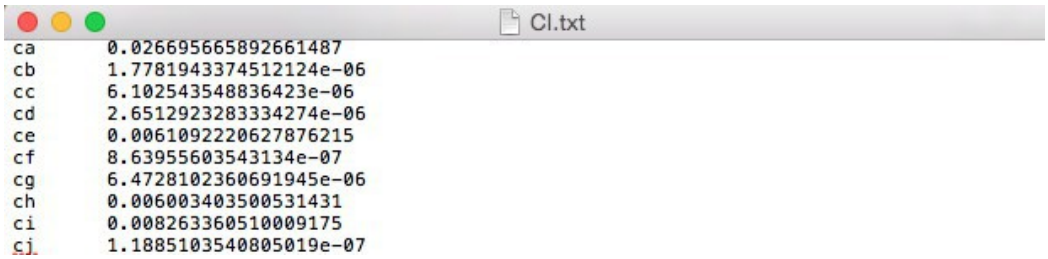


Figura 15, probabilità delle coppie iniziali per l'italiano

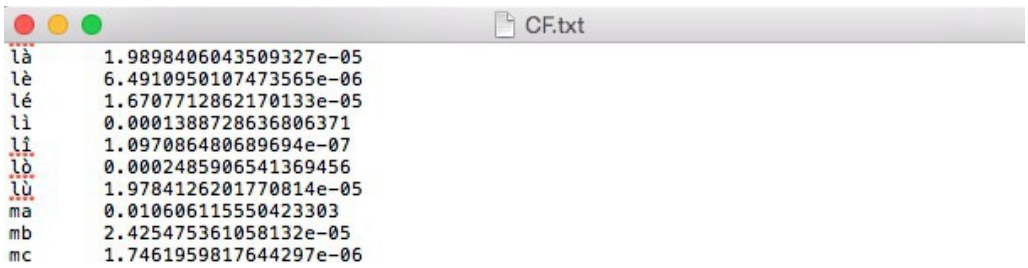


Figura 16, probabilità delle coppie finali per l'italiano

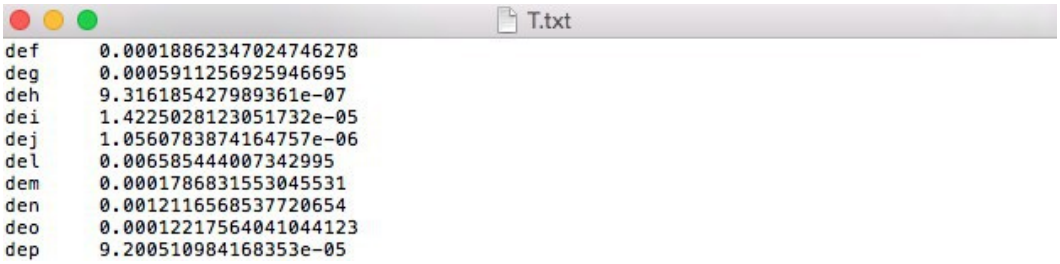


Figura 17, probabilità delle triple per l'italiano

Successivamente è stato scritto un programma Python `work` che legge i dizionari delle coppie, delle coppie iniziali, delle coppie finali e delle triple ed il formario `work.json`, conta le parole contenute all'interno del formario e ne calcola la frequenza relativa nel formario e la probabilità nel modello. Il programma restituisce due file: `P`, che contiene la lista delle parole con probabilità nel modello maggiore di zero, e `R`, che contiene la lista delle parole rifiutate perché con probabilità zero. Nel file `P`, allo scopo di rendere più agevoli le rappresentazioni grafiche, sia delle frequenze relative che delle probabilità viene salvato il logaritmo in base dieci cambiato di segno.

Affinché sia possibile lavorare separatamente con parole della stessa lunghezza il programma restituisce anche una cartella `SPLIT` nella quale il file `P` è stato suddiviso in base alla lunghezza delle parole.

Di seguito si riporta la parte centrale di `work`[9]:

```
(...)

def work(lang):
    start = time()

    CI = read(dir+lang+"/CI.json")
    C  = read(dir+lang+"/C.json")
    CF = read(dir+lang+"/CF.json")
    T  = read(dir+lang+"/T.json")

    D = read(dir+lang+"/work.json")
    sD = 0
    for word in D.keys():
        sD += D[word]
    sD = float(sD)
    for word in D.keys():
        D[word] = -math.log10(D[word]/sD)

    P = {}
    for word in D.keys():
        P[word] = prob(word, C, CI, CF, T)
```

9 Per la versione completa del programma si consulti la sezione Appendice

```
    rej=[]
    with codecs.open(dir+lang+"/P.txt", "w", "utf-8") as out:
        for word in sorted(list(P.keys())):
            if P[word] > 0:
                P[word] = -math.log10(P[word])
                 out.write("{0:s}\t{1:10.5f}\t{2:10.5f}\n".format(word,P[wor
                           d],D[word]))
            else:
                rej.append("{0:10.5f}\t{1:s}".format(D[word],word))
                P.pop(word,None)

    with codecs.open(dir+lang+"/R.txt","w", "utf-8") as out:
        for line in sorted(rej):
            out.write(line+"\n")

    for l in range(4,31):
         with codecs.open(dir+lang+"/SPLIT/"+lang+str(l)+".txt", "w", "utf-
              8") as out:
            for word in sorted(list(P.keys())):
                if len(word) is l:
                     out.write("{0:s}\t{1:10.5f}\t{2:10.5f}\n".format(word,P
                     [word],D[word]))

(...)

work("IT")
work("ES")
work("FR")
work("DE")
work("EN")
```

Di seguito si propongono alcuni esempi di risultati prodotti per l'italiano:

P.txt

```
abbandonare        8.95367        4.30583
abbandonarla      10.18528        5.87313
abbandonarle      10.82592        6.68604
abbandonarli      10.96701        6.68604
abbandonarlo      10.36649        5.81098
abbandonarne      11.23943        6.98707
abbandonarono     10.94741        4.85997
abbandonarsi      10.70891        6.20892
abbandonasse       9.96470        6.08398
```

Figura 18, logaritmo in base 10 cambiato di segno delle frequenze relative e delle probabilità delle parole italiane

R.txt

```
6.28810      mizler
6.28810      molisii
6.28810      montferrand
6.28810      monégasque
6.28810      mumtaz
6.28810      musique
6.28810      mvsn
6.28810      méie
6.28810      métro
6.28810      nagarjuna
```

Figura 19, lista di parole rifiutate perché con probabilità zero nel modello

5. Analisi dei risultati per una singola lingua

I file che compongono P, e cioè le liste di parole con probabilità nel modello maggiore di zero, suddivisi per lunghezza, possono essere analizzati per avere un'idea dei valori di frequenza relativa e probabilità. Nel seguito tratteremo sempre questi valori come logaritmo in base dieci.

Riportando i dati di frequenza relativa e probabilità in un istogramma è possibile condurre una prima analisi dei risultati. Per confrontare le stringhe di lunghezza diversa è stata fatta una ulteriore normalizzazione che consiste nel mettere i valori di frequenza relativa nel campione e probabilità nel modello a media pari a zero e varianza pari a uno[10].
I valori di frequenza relativa e probabilità sono anche stati cambiati di segno di modo che a valori più elevati corrispondano stringhe più probabili.
Per ogni *n* è stato calcolato sia l'istogramma delle frequenze relative del campione (blu), sia l'istogramma delle probabilità nel modello (ocra), sia il grafico cartesiano delle coppie "frequenza nel campione, probabilità nel modello".

Di seguito si riportano alcuni esempi[11]:

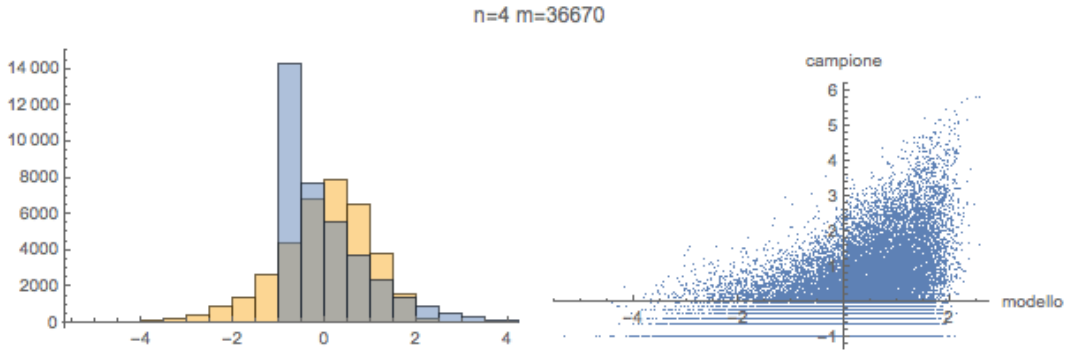


Figura 20, grafici per parole di lunghezza 4 dell'italiano

10 La varianza di una variabile è una funzione che fornisce una misura della variabilità dei valori assunti dalla variabile, nello specifico, di quanto essi si discostino quadraticamente dalla media aritmetica o dal valore atteso. (Wikipedia, voce *Varianza*)

11 Nei grafici che seguono "*n*" indica la lunghezza delle parole ed "*m*" il numero di parole appartenenti alla lingua considerata che hanno lunghezza uguale ad *n*.

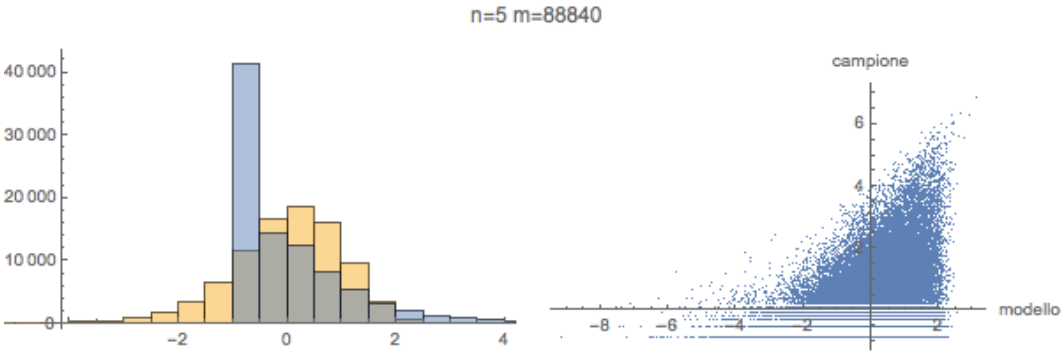


Figura 21, grafici per parole di lunghezza 5 dell'italiano

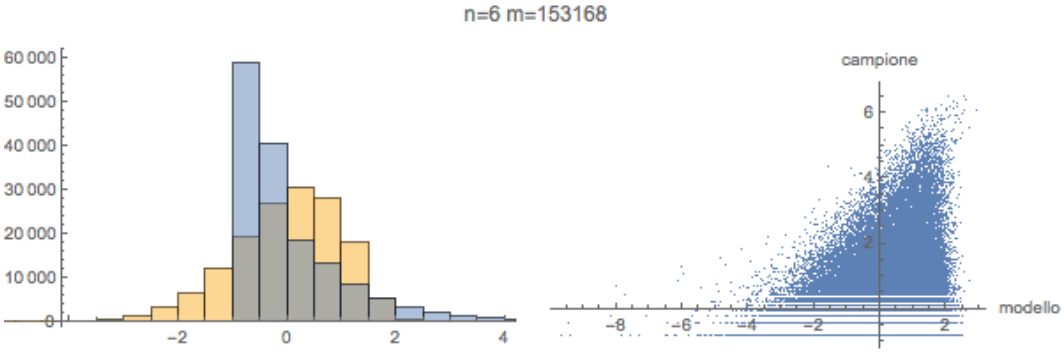


Figura 22, grafici per parole di lunghezza 6 dell'italiano

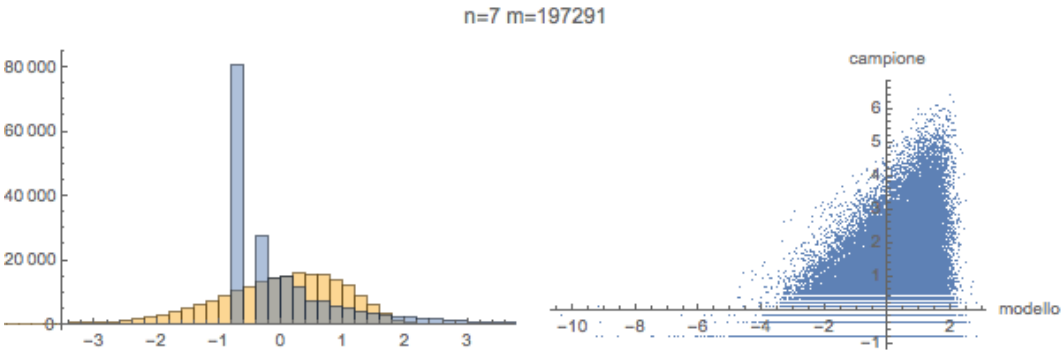


Figura 23, grafici per parole di lunghezza 7 dell'italiano

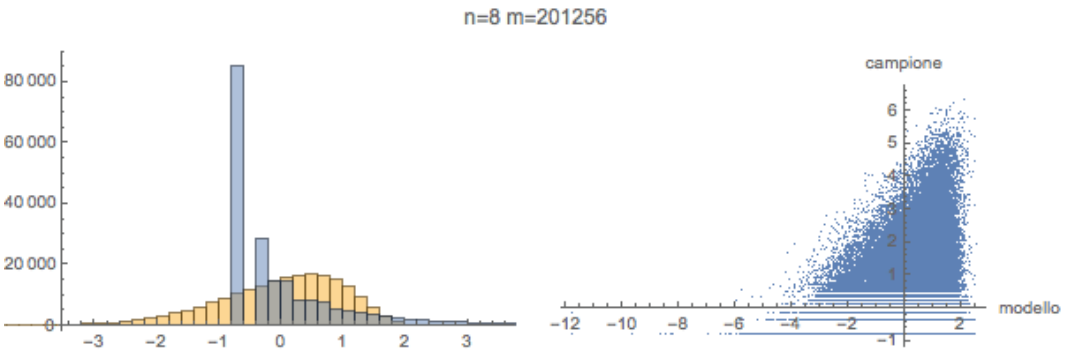


Figura 24, grafici per parole di lunghezza 8 dell'italiano

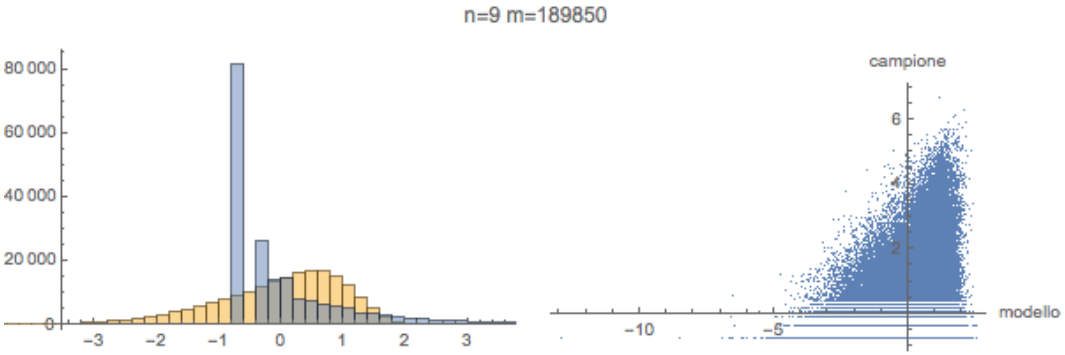


Figura 25, grafici per parole di lunghezza 9 dell'italiano

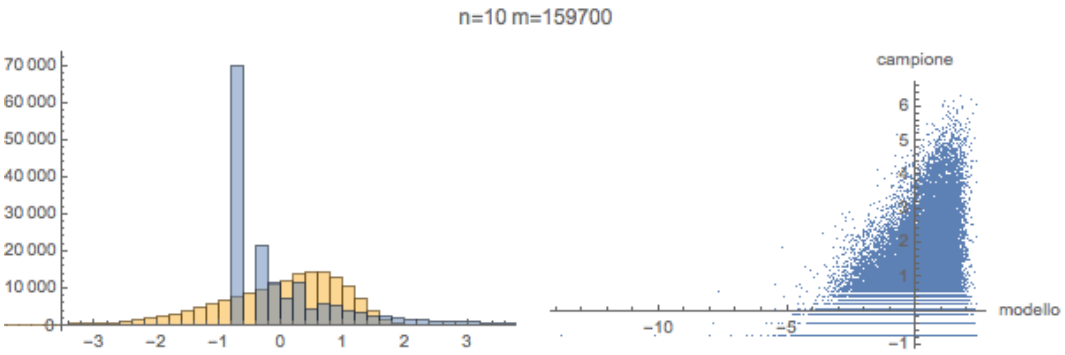


Figura 26, grafici per parole di lunghezza 10 dell'italiano

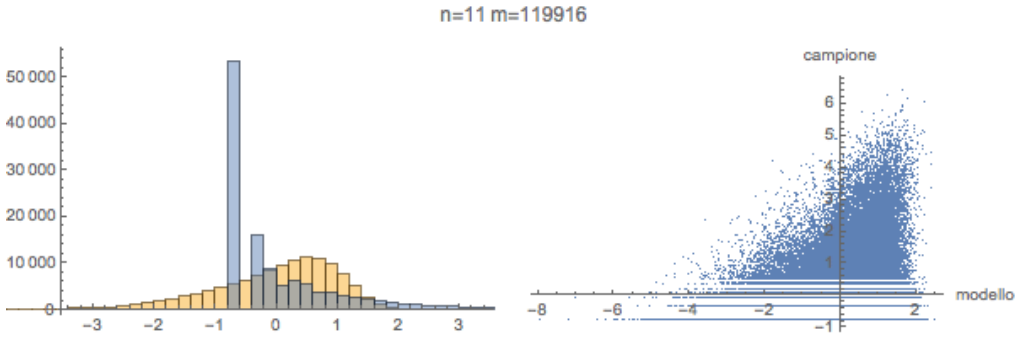


Figura 27, grafici per parole di lunghezza 11 dell'italiano

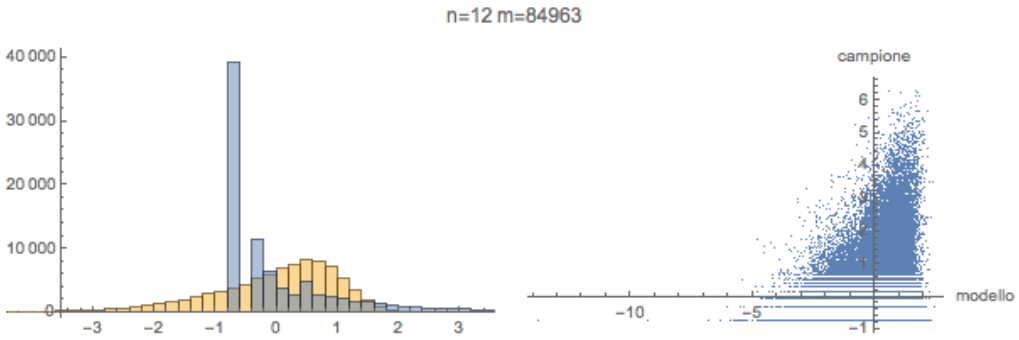


Figura 28, grafici per parole di lunghezza 12 dell'italiano

Come ci aspettavamo, dai grafici emerge che la distribuzione della probabilità nel modello ha un andamento a campana leggermente asimmetrico. Non si tratta di una gaussiana in quanto è evidente una leggera asimmetria. D'altra parte le frequenze relative hanno una distribuzione zipfiana, ovvero si presentano molti casi di parole rare e pochi casi di parole con elevata frequenza. I risultati ottenuti quindi sono in buon accordo con la teoria e mostrano come probabilità nel modello e probabilità nel campione abbiano due comportamenti profondamente diversi tra loro.

Determinare l'appartenenza al linguaggio di una parola osservando solamente la sua frequenza sarebbe sbagliato perché ovviamente errori molto frequenti o parole straniere entrate nell'uso linguistico risulterebbero facenti parte della lingua considerata. Un evento di questo tipo si potrebbe verificare ad esempio con l'italiano per il termine *Wikipedia*, che è molto frequente all'interno della stessa enciclopedia

sebbene non sia una parola italiana (la *w* non fa parte dell'alfabeto italiano).

I risultati sono graficamente simili ma per tutte le lingue prese in esame. Di seguito vediamo un esempio per l'inglese:

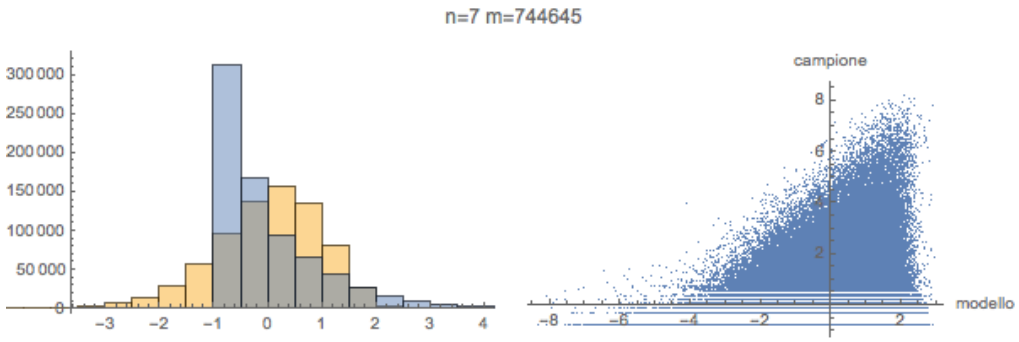


Figura 29, grafici per le parole di lunghezza 7 dell'inglese

Osservando i grafici triangolari sul piano cartesiano si può attribuire un significato alle varie posizioni. In corrispondenza della parte più alta del grafico si trovano parole per le quali si registra una elevata probabilità sia nel campione che nel modello, alla zona in basso a sinistra corrisponde invece una bassa probabilità nel campione e nel modello, mentre a destra abbiamo una probabilità mista, alta nel modello e bassa nel campione.

Per selezionare i casi estremi si è fatto un ulteriore riordinamento dei dati. Oltre alla probabilità si è calcolato anche il rango[12] delle parole. Alle parole, ordinate in base alla loro probabilità, sono stati assegnati valori tanto più alti quanto alta era la posizione della parola, normalizzati tra 0 ed 1. Nell'insieme delle coppie (*rango nel modello*, *rango nel campione*) si sono selezionati i quattro casi estremi (fig. 30) e tali insiemi sono stati riportati con gli stessi colori nel grafico delle probabilità (fig. 31).

12 Posizione che una parola occupa all'interno di una lista. (A. Lenci et A., *Testo e computer* 2005)

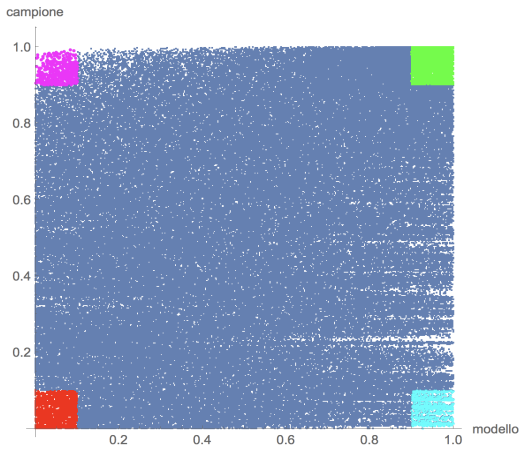


Figura 30, grafico relativo a parole di lunghezza 7, confronto tra i ranghi

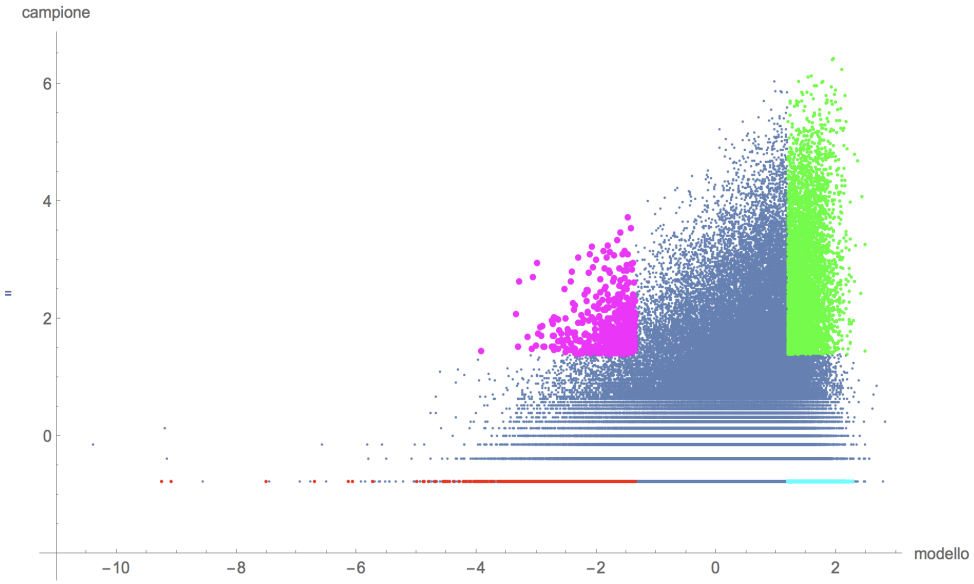


Figura 31, grafico relativo a parole di lunghezza 7 per l'italiano

Nei due ultimi grafici le parole sono state evidenziate con colori diversi, il verde corrisponde a parole appartenenti alla lingua presa in esame, il rosso corrisponde invece a parole inesistenti o sbagliate, il ciano a parole rare ma probabili ed in fine il magenta corrisponde e parole frequenti ed improbabili.

Una campionatura di parole collocate in queste zone è riportata come esempio nelle figure sottostanti, nel colore della posizione corrispondente.[13]

13 Le colonne rappresentano rispettivamente la parola, il logaritmo in base 10 della probabilità normalizzata nel modello, il logaritmo in base 10 della frequenza relativa normalizzata nel campione, il rango nel modello normalizzato tra 0 ed 1 e il rango nel campione normalizzato tra 0 ed 1.

appelli	1.3178	2.8467	0.929992	0.976958
avverta	1.56188	1.43698	0.971783	0.904826
camerte	1.59774	1.60649	0.975929	0.918871
campese	1.22399	2.21985	0.907345	0.955756
capendo	1.37925	3.23895	0.94281	0.985235
casette	1.43479	2.65802	0.952735	0.971793
cassita	1.8502	1.60649	0.994009	0.918851
cassiti	1.64221	1.46903	0.980623	0.907664
cellesi	1.21618	1.84193	0.905434	0.935446
estorta	1.67926	1.66801	0.983836	0.923286
mescita	1.26258	1.53739	0.917097	0.913179
pesista	1.40273	1.46903	0.947195	0.907081
posture	1.62405	2.3373	0.978808	0.96052
potendo	1.51339	4.01213	0.965077	0.994678
riparia	1.57434	2.41932	0.973207	0.963637
riporto	1.3497	3.31494	0.936885	0.986629
sicardo	1.23981	1.93065	0.911446	0.940611
sposina	1.35961	1.55548	0.938735	0.914401
trifore	1.2996	2.55578	0.926018	0.968508
vergato	1.57905	2.10216	0.973749	0.950033

Figura 32, parole sicuramente appartenenti alla lingua italiana

salihaj	-1.3544	-0.767882	0.0991834	0.0876117
shazdeh	-3.23861	-0.767882	0.00360888	0.0714832
shnautz	-1.81117	-0.767882	0.0490342	0.0698157
shogtun	-2.52215	-0.767882	0.014172	0.0697092
slumped	-1.47659	-0.767882	0.0829587	0.0643567
snoodle	-1.57419	-0.767882	0.0714731	0.0635964
szelenc	-2.66924	-0.767882	0.0106036	0.0490443
szemmel	-2.03994	-0.767882	0.0338789	0.049024
totanbi	-1.67706	-0.767882	0.0609151	0.0319427
triumpf	-1.62597	-0.767882	0.0658216	0.0285669
troizem	-2.57231	-0.767882	0.0127781	0.0281919
valadié	-1.93632	-0.767882	0.040159	0.0183435
valujev	-2.18065	-0.767882	0.0266104	0.0173804
vaudrec	-1.92785	-0.767882	0.0406151	0.0157787
vidvcas	-1.74216	-0.767882	0.0548631	0.0111764
vouillè	-1.76084	-0.767882	0.0532209	0.00661459
zaheeda	-2.16619	-0.767882	0.0273251	0.00537784
zbiralo	-2.40436	-0.767882	0.0176288	0.00412082
zihlala	-3.28121	-0.767882	0.00323887	0.00233665
zurulos	-2.04142	-0.767882	0.0338282	0.000136854

Figura 33, errori di battitura o parole inesistenti estratte dal formario dell'italiano

ritrato	1.56102	-0.767882	0.971686	0.0963044
rivegli	1.35879	-0.767882	0.938583	0.09598
ruscono	1.41668	-0.767882	0.949663	0.0903133
ruttori	1.34442	-0.767882	0.935735	0.0899179
scatoli	1.20384	-0.767882	0.902231	0.081874
scentia	1.39816	-0.767882	0.946353	0.0816459
scoprie	1.23486	-0.767882	0.910082	0.0793346
spicali	1.36501	-0.767882	0.939926	0.0588065
staggie	1.35859	-0.767882	0.938527	0.0571389
streggi	1.36362	-0.767882	0.939648	0.0540065
svivola	1.41521	-0.767882	0.949349	0.0492166
terelli	1.23264	-0.767882	0.909433	0.0410764
tersive	1.42835	-0.767882	0.951716	0.0404986
tondale	1.23902	-0.767882	0.911192	0.0337775
unicata	1.5828	-0.767882	0.97416	0.0219219
unitore	1.49597	-0.767882	0.962482	0.0216026
ventava	1.56294	-0.767882	0.97193	0.0140655
versita	1.73699	-0.767882	0.988134	0.0127831
vienche	1.44874	-0.767882	0.955066	0.0111054
vigiano	1.29166	-0.767882	0.924213	0.0109179

Figura 34, parole di assonanza italiana ma escluse dalla lingua. Anche in questo caso possono essere presenti errori di battitura (*rivegli* invece di *risvegli*) o parole molto rare (*ruttori*, *scatoli*)

adhémar	-2.6416	1.61456	0.0111105	0.919581
berlioz	-1.5234	2.68952	0.0772058	0.972695
berlocq	-1.5037	2.05305	0.0794714	0.947484
bhishma	-1.70954	1.42588	0.0576712	0.903741
bitcoin	-1.47965	2.10883	0.082543	0.950464
cubeddu	-2.18585	1.49935	0.0263367	0.910057
enfield	-1.62068	2.56024	0.0664044	0.968661
gouldin	-1.44977	2.03062	0.086304	0.946272
hatzair	-2.71223	1.46903	0.00973182	0.907416
hérault	-2.12212	2.78066	0.0295148	0.975209
hodgins	-1.43287	1.73648	0.088372	0.928324
jujitsu	-2.01258	1.82015	0.0353843	0.933879
melchor	-1.51034	1.987	0.0786959	0.944007
najdorf	-2.36243	1.46903	0.0189821	0.907203
peugeot	-1.47761	3.71585	0.0828015	0.992199
qingdao	-2.14471	2.09539	0.0283743	0.949714
sévigné	-2.01276	1.82015	0.0353691	0.933753
shatner	-1.63932	1.97449	0.0644023	0.943302
sunbird	-1.93448	1.70328	0.0403059	0.925663
zutphen	-2.3516	1.41454	0.0194687	0.901562

Figura 35, parole improbabili ma frequenti; tendono ad essere parole straniere

6. Confronto tra lingue diverse

In questo capitolo si accenna a come può essere impostato il confronto tra lingue diverse. La trattazione è solo preliminare in quanto le assunzioni sono state stabilite senza una approfondita fase di valutazione che andrebbe effettuata con l'ausilio di esperti umani, possibilmente madrelingua, e che potrebbe essere oggetto di una tesi futura.
Ciò che segue è solo un esperimento che mostra le potenzialità dell'approccio seguito.

È stata dapprima definita empiricamente una funzione che dovrebbe stimare il grado di appartenenza di una singola parola ad una determinata lingua. La funzione combina i valori di probabilità nel modello e nel campione.

Data una parola w, dette p_{typ} la probabilità nel modello e f_r la frequenza relativa si definiscono le quantità

$$t(w) = \begin{cases} 0 & \text{se } p_{typ} = 0 \\ \log_{10} p_{typ} & \text{altrimenti} \end{cases}$$

$$c(w) = \begin{cases} 0 & \text{se } f_r = 0 \\ \log_{10} f_r & \text{altrimenti} \end{cases}$$

e la funzione

$$v(w) = c(w)\sqrt[4]{t(w)}$$

Questo oggetto viene normalizzato tra 0 ed 1 cambiando il segno:

$$val(w) = \frac{-v - min(v)}{max(-v) - min(-v)}$$

Il grado di appartenenza alla lingua si determina in base al valore assunto dalla funzione $val(w)$, sempre compreso tra 0 ed 1: al crescere del valore della funzione cresce anche il grado di appartenenza alla lingua.
Assumendo che la funzione sopra definita possa essere usata nello stesso modo per tutte le lingue, fissata una soglia sotto la quale le parole sono considerate errori, si

può attribuire l'insieme delle parole di tutti i formari alle varie lingue secondo i seguenti criteri:

- se la parola è sotto la soglia di validità in tutte le lingue è da considerare un errore e non viene attribuita a nessuna lingua. Ad esempio ***aarogya*** è una parola indiana presente in Wikipedia ma chiaramente non appartenente a nessuna delle lingue europee prese in considerazione; ***bhuddha*** è una scrittura alternativa a ***buddha*** probabilmente dovuta ad un errore di trascrizione (e comunque presente 76600 volte su Google).

- Se la parola supera la soglia in almeno una lingua viene attribuita alla lingua in cui ha il valore più elevato. Per esempio ***abitare***, ***barboni*** e ***cantare*** sono attribuite all'italiano.

Ovviamente questo processo è suscettibile di errori. La lista seguente tratta dalle parole attribuite allo spagnolo è stata rivista da un madrelingua (lo studente Erick Rivadeneira Lopez che qui ringraziamo) e presenta in verde nomi di luoghi e in giallo parole erroneamente attribuite allo spagnolo.

abanico, abarcas, abaxial, abuelas, acateno, actopan, adjunto, adosada, aeronca, afeitar, agrario, aguanta, aguerre, agujeta, ajedrez, alardos, alarido, alcanar, alcance, alcanzo, alfredo, almacén, almaren, almorta, alocada, alzaron, amerite, amilpas, amparar, andaluz, andinos, angulas, antenal, apalone, apartar, apatita, apatura, apilado, aplazar, apremio, aprismo, apropie, arabela, aramina, ardilla, arenosa, aristas, arlanza, arnesto, arquero, arriate, arrieta, arrolla, artavia, atacado, ataques, atesora, atrasos, bacalar, bailada, ballota, balones, barajar, baratos, barbado, barbeta, bareiro, bargota, barrado, barreal, bertero, biblias, bilardo, bonillo, bordado, boricua, borobia, bosques, briales, brindar, buenfil, buntaro, burrita, busardo, cachete, cachina, cambias, cameros, campino, caninos, canosas, capaces, caranga, caranta, caravia, caricia, cartera, cartuja, castiza, caudiel, cedrela, celosas, cerrado, chabola, chambao, chamula, chapala, chicoma, chiripa, cholila, chupete, cipango, cobijas, codicia, coelemu, colanta, colocan, combate, comerla, comprar, compren, conesup, congela, conocer, copihue, corimbo, cornago, cortijo, coslada, creamos, crezcan, ctesias, culmina, cumbres, curvada, debello, decanta, declive, dejarla, delmira, dentado, denuedo, desafio, desarma, diadema, directo, dividir, donaldo, dudaron, durando, echarse, edmundo, ejercen, emerger, enamora, enredos, enumera, ercilia, erguida, escanda, esconde, escuela, escuque, escutia, esfinge, esmalte, espacio, espalda, espanya, esperen, estacio, estepas, estrato, estruch, estupas, eusebia, excesos, ezcurra, factura, fanales, fanzara, febrero, ficarro, fierita, figallo, flaquer, fonteta, forques, freites, fresera, fumando, gadifer, galador, ganando, ganchos, garnelo, gaumata, génesis, godella, gorospe, goyesca, grafiti, grafito, granizo, greyjoy, griegas, grindor, guardan, guizado, habanos, haberle, hablada, haceros, heredad, hidalgo, himeneo, hogares, holando, honores, huarpes, huberto, huesito, hullera, iguales, imataca, indigna, informe, injusto, insecto, intacto, invocar, irurita, japones, jordana, juanita, juliaca, julissa, laboreo, laciana, laporta, larroca, latorre, lavadas, lectora, ledesma, lestard, levanta, levedad, limonal, limosna, litoria, litosol, lizarza, llanada, llanura, llerena, llevaba, lloreda, lomadas, lombera, lucense, luperca, lusones, macanaz, macarao, macarra, maceros, maderas, maestra, malasia, malevos, malleco, manacus, managua, manlleu, manzana, maraver, maricas, mariela, marista, marsias, martita, masajes, masilla, mataron, mayores, mechero, medidor, melazas, mencias, mensaje, merello, merilia, méritos,

migajas, militan, mineria, minniza, minueto, mixteco, mojarra, molusco, momplet, monfero, montaje, moraron, mordido, morisca, mostras, muciano, mudaron, mudarra, muguiro, multado, munidal, muniesa, muriano, nadando, navajas, negarle, negrita, nemesio, notaria, novarro, numitor, obdulio, oblicua, obligan, obstina, ocupada, ofertas, oncenio, orégano, oriundo, orrente, oscense, ostrero, palleja, paquete, parches, paridad, paridos, parques, parvula, pasiego, pastuso, patones, patotas, patraix, paulita, payasos, pentodo, pepinos, perfora, permach, permito, pesetas, pestana, picardo, pinchos, pinillo, pintada, pintora, piquete, piropos, pixeles, plastos, plomero, podando, podrido, pomares, ponerla, ponsoda, poquito, porcino, portgas, postula, postura, pozones, premiso, prendes, presura, previos, procese, promero, proveen, pubenza, puberal, quidora, quietos, quijote, quincha, ranquin, ranulfo, reactos, rechace, reclama, rectora, recueva, reducto, regateo, regatos, relucir, renuevo, repique, réplica, resumir, revuelo, roldana, rosemon, rosinos, rosique, rotalia, rurales, sabinos, sacanta, salares, saliera, salinas, salvada, samario, santito, saravia, sastres, scirtes, seguido, seguras, selecto, seremos, siderca, sigamos, silicio, sobrino, solarte, sonaran, soporto, sorondo, soteras, standar, suanzes, subirla, sucinto, sullana, superas, tabales, talledo, tamarit, tapices, tapirus, tarajal, tarento, tejados, también, teorias, tiermas, tipitos, tiramos, tirarse, tiscapa, tobarra, tomista, tomoxia, tondero, toneles, torrego, trancho, tranque, trazara, tricias, troncos, troquel, trovero, turbias, turgida, turquia, tuviere, ultraje, unirnos, uralita, valacos, validad, validez, vandera, varitas, vejigas, ventana, veremos, verraco, vestida, vidente, vienesa, viguesa, visiedo, vivenda, zaballa, zafiros, zancada, zulueta

Appendice

Di seguito si riporta il codice Python relativo a tutti i programmi.

`elab`:

```
import re
import codecs
from time import time
import glob
import json

#dirin =  "/Users/giuliaguidi/Dropbox/WIKIg/30_11/wiki/"
#dirout = "/Users/giuliaguidi/Dropbox/WIKIg/30_11/wiki/"

dirin = "/Users/FR/Desktop/WIKI/DATA/"
dirout = "/Users/FR/Dropbox/ LAVORI/WIKIW/FREwiki/"

def lista(dir):
    return glob.glob(dir + "*")

def write(file, D):
    out=codecs.open(file,"w", "utf-8")
    words=sorted(list(D.keys()))
    for word in words:
        fre = str(D[word])
        out.write(str(word)+"\t"+fre+"\n")
    out.close()

def dump(file, D):
    with codecs.open(file, "w", "utf-8") as out:
        json.dump(D, out)

def elab(file, D):
    start = time()
    # apro il file in lettura
    inp = codecs.open(file, "r", "utf8")
    # efettuo un ciclo su ciascuna riga del file
    for line in inp:
      #chiamo il metodo strip per rimuovere i caratteri di inizio e fine
      #riga
        line = line.strip()
        #controllo che line sia maggiore di tre
        if len(line) > 3:
            #uso una RegExp per eliminare le < > e il relativo contenuto
            line=re.sub(r"<.*>", " ", line)
            #trasformo i caratteri in un array di caratteri
```

```
            ba = list(line)
            #controllo che ciascun carattere dell'array sia un carattere
             #alfabetico, altrimenti lo elimino
            for i in range(len(ba)):
                c = ba[i]
                if not c.isalpha():
                    ba[i] = ' '
            #restituisco una line pulita
            line = "".join(ba)
            #chiamo il metodo split per ottenere le parole
            words = line.split()
            # per ciascun elemento in words
            for word in words:
                # controllo che la lunghezza sia maggiore di 3
                if len(word) > 3:
                    # trasformo tutte le lettere in minuscole con casefold
                    word = word.casefold()
                    # aggiungo le parole alla lista D con la relativa
                   #frequenza
                    D[word] = D.setdefault(word, 0) + 1
    #calcolo il tempo
    t = int(time() - start + 0.5)
    #stampo i risultati
    print(file + "  " + str(t) + " sec.")


def scan(lang):
    start = time()

    D = {}
    for file in lista(dirin + lang + "/"):
        elab(file, D)

    dump(dirout +lang+"/raw.json", D)
    write(dirout+lang+"/raw.txt", D)

    t = int(time() - start + 0.5)
    print(lang + "  " + str(t) + " sec.")

scan("IT")
scan("ES")
scan("FR")
scan("DE")
scan("EN")
```

filter:

```
import codecs
from time import time
import json

#dir = "/Users/FR/Dropbox/ LAVORI/WIKIW/FREwiki/"
dir =  "/Users/giuliaguidi/Dropbox/WIKIg/30_11/Wiki”

maxlenT  = 50    # massima lunghezza per il training
maxlenW  =100    # massima lunghezza per work
minfreq = 100   # minima frequenza per il training
F={} #liste per frequenze e lunghezze
L={}

def read(file):
    #apro il file in lettura
    inp=codecs.open(file,"r","utf8")
    return json.load(inp)

def write(file, D):
    out=codecs.open(file,"w", "utf-8")
    words=sorted(list(D.keys()))
    for word in words:
        fre = str(D[word])
        out.write(str(word)+"\t"+fre+"\n")
    out.close()

def dump(file, D):
    with codecs.open(file,"w", "utf-8") as out:
       json.dump(D, out)

def filter(lang):
    start = time()
    longT=[]
    longW=[]
    rare=[]
    strangeW=[]
    strangeT=[]

    setc = read(dir+"setAll.json") # dipende dall'insieme delle lingue
      #analizzate
    D = read(dir+lang+"/raw.json") #alfabeto locale

    for word in sorted(list(D.keys())):
        lw = len(word)
        fre = D[word]
```

```
        if lw>maxlenW:
            longW.append("{0:9d}\t{1:10d}".format(fre,lw)+"\t"+word)
            D.pop(word,None)
        elif not letter(word, setc): #alfabeto locale
            strangeW.append("{0:9d}\t{1:4d}".format(fre,lw)+"\t"+word)
            D.pop(word,None)

    write(dir+lang+"/work.txt", D)
    dump(dir+lang+"/work.json", D)
    with codecs.open(dir+lang+"/longW.txt","w", "utf-8") as out:
        for line in sorted(longW, reverse=True):
            out.write(line+"\n")
    with codecs.open(dir+lang+"/strangeW.txt","w", "utf-8") as out:
        for line in sorted(strangeW, reverse=True):
            out.write(line+"\n")

    setc = read(dir+lang+"/setc.json") # alfabeto di lang
    for word in sorted(list(D.keys())):
        lw = len(word)
        fre = D[word]
        if lw>maxlenT:
            longT.append("{0:9d}\t{1:3d}".format(fre,lw)+"\t"+word)
            D.pop(word,None)
        elif not letter(word, setc):
            strangeT.append("{0:9d}\t{1:3d}".format(fre,lw)+"\t"+word)
            D.pop(word,None)
        elif fre<minfreq :
            rare.append("{0:9d}\t{1:3d}".format(fre,lw)+"\t"+word)
            D.pop(word,None)
        else:
            L[lw]=L.setdefault(lw, 0)+1
            F[fre]=F.setdefault(fre, 0)+1

    write(dir+lang+"/training.txt", D)
    dump(dir+lang+"/training.json", D)
    write(dir+lang+"/fre.txt", F)
    write(dir+lang+"/len.txt", L)

    with codecs.open(dir+lang+"/longT.txt","w", "utf-8") as out:
        for line in sorted(longT, reverse=True):
            out.write(line+"\n")
    with codecs.open(dir+lang+"/rare.txt","w", "utf-8") as out:
        for line in sorted(rare, reverse=True):
            out.write(line+"\n")
    with codecs.open(dir+lang+"/strangeT.txt","w", "utf-8") as out:
        for line in sorted(strangeT, reverse=True):
            out.write(line+"\n")
    t = int(time()-start+0.5)
    print(lang+"  "+str(t)+" sec.")
```

```
def letter(word, setc):
    ba = list(word)
    for c in ba:
        if c not in setc :
            return False
    return True

filter("IT")
filter("ES")
filter("FR")
filter("DE")
filter("EN")
```

chars:

```
import codecs
import json

dir = "/Users/FR/Dropbox/ LAVORI/WIKIW/FREwiki/"
#dir =  "/Users/giuliaguidi/Dropbox/WIKIg/30_11/Wiki”

setAll =
['a','à','ä','â','á','æ','b','c','ç','d','e','ë','è','é','ê','f','g','h','k
','i','ì','í','ï','î','j','l','m','n','ñ','o','ò','ó','ö','ô','œ','p','q','
r','s','t','u','ù','ú','ü','û','v','w','x','y','ÿ','z']
setIT =
['a','à','b','c','d','e','è','é','f','g','h','i','ì','î','j','l','m','n','o
','ò','p','q','r','s','t','u','ù','v','z']
setEN =
['a','b','c','d','e','f','g','h','k','i','j','l','m','n','o','p','q','r','s
','t','u','v','w','x','y','z']
setFR =
['a','à','â','æ','b','c','ç','d','e','ë','è','é','ê','f','g','h','k','i','ï
','î','j','l','m','n','o','ô','œ','p','q','r','s','t','u','ù','ü','û','v','
w','x','y','ÿ','z']
setES =
['a','à','b','c','d','e','è','é','f','g','h','i','ì','j','l','m','n','ñ','o
','ò','p','q','r','s','t','u','ù','ü','v','w','x','y','z']
setDE =
['a','ä','b','c','d','e','f','g','h','i','j','k','l','m','n','o','ö','p','q
','r','s','t','u','ü','v','w','x','y','z']


def dump(file, D):
    with codecs.open(file,"w", "utf-8") as out:
       json.dump(D, out)

dump(dir+"/setAll.json", setAll)
dump(dir+"IT/setc.json", setIT)
dump(dir+"EN/setc.json", setEN)
dump(dir+"DE/setc.json", setDE)
dump(dir+"FR/setc.json", setFR)
dump(dir+"ES/setc.json", setES)
```

trainer:

```
import codecs
from time import time
import json

dir = "/Users/FR/Dropbox/ LAVORI/WIKIW/FREwiki/"
#dir = "/Users/giuliaguidi/Dropbox/WIKIg/30_11/Wiki”

minfreq = 10

def read(file):
    #apro il file in lettura
    inp=codecs.open(file,"r","utf8")
    return json.load(inp)

def writeMap(file, D):
    with codecs.open(file,"w", "utf-8") as out:
        for word in sorted(list(D.keys())):
            fre = str(D[word])
            out.write(str(word)+"\t"+fre+"\n")

def dump(file, D):
    with codecs.open(file,"w", "utf-8") as out:
       json.dump(D, out)

#controllo che una parola sia fatta solo da lettere
def letter(word, setc):
    ba = list(word)
    for c in ba:
        if c not in setc :
            return False
    return True

def trainer(lang):
    start = time()

      #inizializza dizionari: coppie iniziali, coppie, coppie    #finali,
      #triple
    CI = {}
    C  = {}
    CF = {}
    T  = {}

    #variabili somma coppie iniziali, somma coppie finali, somma coppie,
      #somma triple
    sCI = 0
    sCF = 0
    sC  = 0
    sT  = 0

      #legge il formario con tutte le parole
    D = read(dir+lang+"/work.json")

      #per ogni parola del dizionario
    for word in D.keys():
        fre = D[word]
      #se la frequenza è più grande di una frequenza minima ne calcolo la
      #lunghezza
        if fre >= minfreq:
            lw = len(word)
            sub = word[0:2]
            CI[sub]=CI.setdefault(sub, 0)+fre #calcola la frequenza delle
             #coppie iniziali
            sCI+=fre #somma le frequenze delle coppie iniziali
            for l in range(lw-2): #lw-2 --> triple
                sub = word[l:l+2] #prende la coppia
                C[sub]=C.setdefault(sub, 0)+fre
                sC+=fre
                sub = word[l:l+3] #prende le triple
                T[sub]=T.setdefault(sub, 0)+fre
```

```
                sT+=fre
            sub = word[-2:]
            CF[sub]=CF.setdefault(sub, 0)+fre
            sCF+=fre
            C[sub]=C.setdefault(sub, 0)+fre
            sC+=fre

    sC=float(sC)
    sCI=float(sCI)
    sCF=float(sCF)
    sT=float(sT)

    #divide le frequenze per le lunghezze, quindi ottiene le probabilità
    for s in CI.keys():
        CI[s]=CI[s]/sCI
    for s in CF.keys():
        CF[s]=CF[s]/sCF
    for s in C.keys():
        C[s]=C[s]/sC
    for s in T.keys():
        T[s]=T[s]/sT

      #dump dei risultati
    dump(dir+lang+"/CI.json", CI)
    dump(dir+lang+"/CF.json", CF)
    dump(dir+lang+"/T.json", T)
    dump(dir+lang+"/C.json", C)

      #scrittura dei risultati
    writeMap(dir+lang+"/CI.txt", CI)
    writeMap(dir+lang+"/CF.txt", CF)
    writeMap(dir+lang+"/T.txt", T)
    writeMap(dir+lang+"/C.txt", C)

    t = int(time()-start+0.5)
    print(lang+"  "+str(t)+" sec.")

trainer("IT")
trainer("ES")
trainer("FR")
trainer("DE")
trainer("EN")
```

work:

```
import codecs
from time import time
import json
import math

dir = "/Users/FR/Dropbox/ LAVORI/WIKIW/FREwiki/"
#dir = "/Users/giuliaguidi/Dropbox/WIKIg/30_11/Wiki”

def read(file):
    #apro il file in lettura
    inp=codecs.open(file,"r","utf8")
    return json.load(inp)

def dump(file, D):
    with codecs.open(file,"w", "utf-8") as out:
       json.dump(D, out)

#stima della frequenza come numeratore su denominatore
def step(num, den):
    if num is 0:
        return 0 #se numeratore=0 non posso fare il rapporto allora
                 #restituisce 0
    if den is 0 :
        return 0 #se denominatore=0 restituisce 0
    return num/den #altrimenti restituisce il rapporto


def prob(word, C, CI, CF, T): #alla fne passo come parametri la parola e le
                              #probabilità delle coppie ecc..
    lw = len(word) #calcola la lunghezza della parola
    #calcolo della prob con la formula della freq
    p = CI.setdefault(word[0:2],0) #SETDEFAULT per forzare a zero le coppie
                                   #iniziali che non sono nel dictionary
    for l in range(lw-2):
        p*=step(T.setdefault(word[l:l+3],0), C.setdefault(word[l:l+2],0))
    sub = word[-2:]
    return p * step(CF.setdefault(sub,0),C.setdefault(sub,0))

def work(lang):
    start = time()
      #legge i dizionari
    CI = read(dir+lang+"/CI.json")
    C  = read(dir+lang+"/C.json")
    CF = read(dir+lang+"/CF.json")
    T  = read(dir+lang+"/T.json")

      #legge il formario
    D = read(dir+lang+"/work.json")
    sD = 0
    for word in D.keys():
      #conta quante sono le parole
        sD += D[word]
    sD = float(sD)
    for word in D.keys():
      #calcola la freq rel e ne fa il log in base 10
        D[word] = -math.log10(D[word]/sD) # --> -log base 10 della prob nel
                                          #campione

    P = {}
    for word in D.keys():
        P[word] = prob(word, C, CI, CF, T)

    rej=[]
    with codecs.open(dir+lang+"/P.txt", "w", "utf-8") as out:
        for word in sorted(list(P.keys())):
            if P[word] > 0: #stampa in chiaro delle prob > 0
                P[word] = -math.log10(P[word])

out.write("{0:s}\t{1:10.5f}\t{2:10.5f}\n".format(word,P[word],D[word]))
```

```
            else:
                rej.append("{0:10.5f}\t{1:s}".format(D[word],word))
                # nelle rej metto le rifiutate
                P.pop(word,None)

    with codecs.open(dir+lang+"/R.txt","w", "utf-8") as out:
        for line in sorted(rej):
            out.write(line+"\n")

    for l in range(4,31): #per lavorare su lunghezze diverse fa un file
                          #per ciascuna lunghezza in un fold SPLIT
        with codecs.open(dir+lang+"/SPLIT/"+lang+str(l)+".txt", "w", "utf-
8") as out:
            for word in sorted(list(P.keys())):
                if len(word) is l:

out.write("{0:s}\t{1:10.5f}\t{2:10.5f}\n".format(word,P[word],D[word]))

    t = int(time()-start+0.5)
    print(lang+"  "+str(t)+" sec.") #tempi


work("IT")
work("ES")
work("FR")
work("DE")
work("EN")
```

Bibliografia

- Alessandro Lenci, Simonetta Montemagni, Vito Pirelli; *Testo e computer, elementi di linguistica computazionale*, Carocci editore 2005.

- Norman Abramson; *Information theory and coding,* Mc Graw-Hill book company 1963.

- Robert Ash; *Information and theory*, John Wiley & sons 1967.

Sitografia

- Wikipedia, l'enciclopedia libera.
  https://www.wikipedia.org

- Wikimedia Downloads.
  https://dumps.wikimedia.org/backup-index.html

- Wikipedia Extractor, Medialab.
  http://medialab.di.unipi.it/wiki/Wikipedia_Extractor

- Wikizionario, il dizionario multilingue libero.
  https://it.wiktionary.org/wiki/Pagina_principale