Information Retrieval

Giuseppe Magazzù

2021 - 2022

Contents

Chapter 1

Definitions

1.1 Document

Un **documento** è solitamente formato da un testo, una struttura, altri media (immagini, suoni, ...) e da dei metadata.

Per **testo** si intende una sequenza di stringhe di caratteri di un alfabeto.
E.g. le sequenze del genoma, formule chimiche e parole del linguaggio naturale.

Un documento può essere composto da

- structured data (tabelle, database, ...)
- semi-structured data (html, xml, ...)

I **metadati** sono dati esterni riguardo al documento. Possono essere classificati in due categorie:

- **metadati descrittivi**: riguardano la creazione del documento (e.g. titolo, autore, data, ...)
- **metadati semantici**: descrivono informazioni contestualmente rilevanti o specifiche del dominio (e.g. ontologie)

1.2 Terms

I termini sono dei descrittori che vengono associati al testo.

1.3 Stop Words

I termini che non sono significativi per la rappresentazione del testo (particelle, articoli, ...).

Chapter 2

Text Processing

Il text processing è una fase necessaria per preparare e pulire il testo.

2.1 Tokenization

La tokenization consiste nell'identificare e separare all'interno di un testo delle unità chiamate token. I token possono essere parole, frasi, simboli o n-grammi. Ogni token è un candidato a essere un termine significativo (index).

> e.g. "Text mining is to identify useful information"
>
> **Tokens**: "Text", "mining", "is", "to", "identify", "useful", "information"

Problemi:

- parole composte ("Hewlett-Packard" $\rightarrow$ "Hewlett", "Packard")
- numeri, date ("Mar. 12, 1991", "12/3/1991", "(800) 234-2333")
- problemi linguistici (parole composte, assenza di spazi, ...)

I token possono essere raggruppati in sequenze contigue di N elementi chiamate N-grammi.

> e.g. "Corpus is the collection of text documents."
>
> **Bigrammi**: "Corpus is", "is the", "the collection", "collection of", "of text", "text documents", "documents ."

La tokenization si può effettuare tramite espressioni regolari o metodi statistici.

2.2 Normalization

Ad un parola possono essere associati diversi token. La normalizzazione consiste nell'ottenere le classi di equivalenza dei token rimuovendo punti, trattini, accenti.

U.S.A. ⇔ USA
anti-aliasing ⇔ antialiasing
résumé ⇔ resume
15/10/2021 ⇔ 15 Ott 2021

Lemmatization

Le parole vengono ridotte alla loro forma base (lemma) tenendo in considerazione l'intero vocabolario della lingua e analizzando la parte del discorso.

e.g. "ladies" ⇒ "lady", "forgotten" ⇒ "forgot"

Stemming

Le parole vengono ridotte a una radice (stem) rimuovendo le flessioni tramite l'eliminazione dei caratteri non necessari.

e.g. "automate(s)", "automation", "automatic" ⇒ "automat"

Case folding

Tutte le parole vengono convertite in lowercase a parte alcune eccezioni.

Thesaurus and Soundex

Un thesaurus (tesauro) è una risorsa linguistica generata manualmente da essere umani in cui è possibile esprimere relazioni tra parole (e.g. gerarchie, sinonimi, ...).

Soundex è un algoritmo fonetico che permette di rappresentare correttamente diverse parole omofone nonostante differenze di ortografia usando delle euristiche fonetiche.

2.3 Stop Words Removal

Le **stop words** sono le parole più frequenti all'interno di un testo che possono essere rimosse senza perdere il significato. Queste parole essendo presenti in più documenti non portano informazioni utili per distinguerli.

Esistono delle liste di **stop words** definite in base alla lingua che possono essere usate per la rimozione.

I web search engine non effettuano la rimozione delle **stop words** perché sono necessarie per alcune ricerche.

Chapter 3

Text Representation

In un sistema di information retrieval i documenti devono essere rappresentati in un formato interno e ordinati per essere indicizzati.

3.1 Bag Of Words

Un modo semplice per rappresentare un testo è una matrice in cui sulle righe ci sono termini estratti dal corpus (vocabolario) e sulle colonne i documenti.

La **Bag Of Words (BOW)** è una rappresentazione del testo che descrive le occorrenze di parole in un documento.

Incidence Matrix: specifica la presenza di un termine in un ogni documento.

Ogni documento può essere rappresentato da un insieme di termini o da un vettore binario.

	Doc1	Doc2	Doc3	Doc4
Term1	1	1	1	0
Term2	0	1	1	1
Term3	0	0	1	0

Rappresentazione di Doc1

$R1 = \{\text{Term1}, \text{Term2}, \text{Term3}\}$
$R1 = \langle 1, 0, 0 \rangle$

Count Matrix: specifica la numero di occorrenze di un termine in ogni documento.

Un documento viene rappresentato da un vettore di occorrenze.

	Doc1	Doc2	Doc3	Doc4
Term1	57	57	71	133
Term2	4	34	17	92
Term3	232	2	10	293

Rappresentazione di Doc1

$R1 = \langle 157, 4, 232 \rangle$

Le rappresentazioni vettoriali non considerano l'ordine delle parole nel testo.

Bag Of Words con N-grammi
Pro: cattura le dipendenze locali e l'ordine
Contro: incrementa la frequenza delle parole

3.2 Zipf's Law

Descrive la frequenza di un evento (parola) in un insieme in base al suo rank.

rank: posizione di un termine nell'ordine decrescente di frequenza dei termini in tutta la collezione.

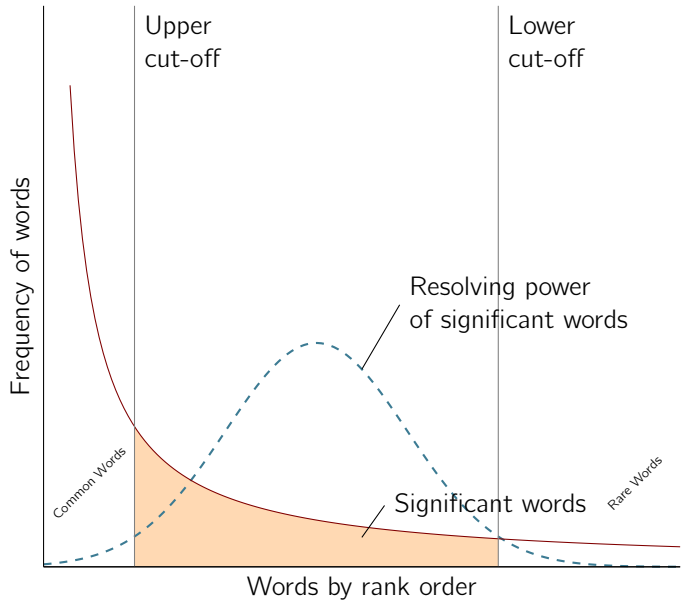
La frequenza di una parola w, f(w) è proporzionale a 1/r(w).

$$f \propto \frac{1}{r} \quad \Rightarrow \quad f \cdot r = k \text{ (costante)}$$

$$P_r = \frac{f}{N} = \frac{A}{r} \qquad \text{probabilità del termine di rank r,} \qquad A = \frac{k}{N} \approx 0.1$$

3.3 Luhn's Analysis

Generalmente termini con frequenza molto alta e molto bassa sono inutili per discriminare i documenti.



L'abilità delle parole di discriminare il contenuto di un documento è massimo nella posizione tra i due livelli di cut-off.

Vogliamo assegnare dei pesi ai termini tenendo conto di questi due fattori:

- corpus-wide: alcuni termini portano più informazione riguardo al documento
- document-wide: non tutti i termini sono ugualmente importanti

Andiamo a definire due frequenze:

- Term Frequency (TF): frequenza di un termine all'interno di un documento
- Inverse Document Frequency (IDF): frequenza di un termine in tutta la collezione

Il peso di un termine deve essere proporzionale a TF e inversamente proporzionale a IDF.

La **term frequency** $tf_{t,d}$ è il numero di occorrenze del termine t nel documento d diviso il numero totale di termini nel documento.

$$tf_{t,d} = \frac{f_{t,d}}{\sum_{t_i} f_{t_i,d}}$$

Questa misura può essere normalizzata dividendo per la frequenza massima nel documento d per essere confrontabile tra documenti diversi.

$$ntf_{t,d} = \frac{f_{t,d}}{max_{t_i} f_{t_i,d}}$$

La **inverse document frequency** idf_t è la frazione inversa della frequenza di un termine in un documento in scala logaritmica.

$$idf_t = log(N/df_t), \quad df_t \leq N$$

dove df_t è la **document frequency**, ovvero il numero di documenti che contengono il termine t, e N il numero totale di documenti.

Infine possiamo calcolare la funzione TF-IDF come prodotto di TF e IDF.

$$tf\text{-}idf_{t,d} = (tf_{t,d}/max_{ti} tf_{ti,d}) * log(N/df_t)$$

Questa funzione rappresenta il peso del termine t all'interno di un documento d.

- termine comune in un documento $\rightarrow$ high tf $\rightarrow$ high weight
- termine raro nella collezione $\rightarrow$ high idf $\rightarrow$ high weight

Chapter 4

Text Enrichment

riconoscere una frase: - n-grams - pos tagger - store words position in a index

POS, NER approaches: - rules based - supervised learning

4.1 Part-of-Speech (POS) tagging

Il POS tagging è il processo che marca ogni termine nel documento con un tag che corrisponde a una part-of-speech.

EXAMPLE...

Word Classes

words that somehow behave alike:

- similar transformations
- similar functions in the phrase
- similar contexts

9 traditional word classes of POS (noun, verb, adjective, adverb, preposition, article, interjection, conjunction)

Applicazioni: - Machine Translation - Parsing - Speech Recognition

Tag Ambiguity

Spesso una parola può essere associata a più di un POS, quindi è necessario considerare il contesto.

- The **back** door (adjective)
- Promised to **back** the bill (verb)
-

Rule Based Tagging

- assegno ogni possibile tag a una parole usando un dizionario - scrivo delle regole a mano per rimuove dei tag - lascio un solo tag per parola

consideriamo le frasi con n-grammi

4.2 Named Entity Recognition (NER)

Trovare e classificare nomi in un testo (persone, date, luoghi, organizzazioni).

Chapter 5

Statistical Language Models

Un statistical LM ci permette di rappresentare un testo mantenendo degli aspetti semantici, tramite una distribuzione di probabilità su sequenze di parole.

Applicazioni:

- Machine Translation, P(**high** winds tonite) > P(**large** winds tonite)
- Spell Correction, P(about 15 **minutes**) > P(about 15 **minuets**)
- Speech Recognition, P(I saw a van) > P(eyes awe of an)

5.1 Language Model

L'obiettivo di un Language Model è quello di calcolare la probabilità di una sequenza di parole $P(W) = P(w_1, w_2, ..., w_n)$.

Con una Language Model è possibile calcolare anche la probabilità di una parola data una sequenza $P(w_5|w_1, w_2, w_3, w_4)$.

Possiamo calcolare la probabilità della sequenza W con la chain rule:

$$P(w_1, w_2, ..., w_n) \approx \prod_{i=1}^{n} P(w_i|w_1, ..., w_{i-1})$$

Con la Markov Assumption possiamo ridurre il numero di parole da condizionare

$$P(w_1, w_2, ..., w_n) \approx \prod_{i=1}^{n} P(w_i|w_{i-k}, ..., w_{i-1})$$

Un Language Model è ben formato su un alfabeto Ω se $\sum_{s \in \Omega} P(s) = 1$.

N-grams Language Model

La probabilità di una parola di una sequenza dipende dalle N parole precedenti. Nel caso di uni-grammi la probabilità non dipende da nessuna altra parola.

Sparsity Problems

1. La parola di cui vogliamo calcolare la probabilità non è presente nel corpus. Soluzione: aggiungere una δ alla frequenza di ogni parola (smoothing).
2. La sequenza di cui vogliamo calcolare la probabilità non è presente nel corpus. Soluzione: ridurre la sequenza da condizionare.

I modelli con uni-grammi sono i più usati

- Spesso sono sufficienti per valutare l'argomento
- Con N più grandi ci sono più problemi di sparsity
- Implementazione semplice ed efficiente

Perplexity

Chapter 6

Word Embedding

Il termine **word embedding** indica una famiglia di tecniche che rappresentano le parole come vettori di numeri reali.

La rappresentazione può essere ottenuta in due modi:

1. Modelli count-based
2. Modelli predittivi

6.1 How to Represent Word Vectors?

Local Representation

Ogni termine del vocabolario T è rappresentato da un vettore binario (one-hot) di dimensione $|T|$, in cui un sola componente ha valore 1 e tutte le altre 0. Ogni elemento del vettore corrisponde una parola diversa.

	King	**Woman**	**Princess**
King	1	0	0
Woman	0	1	0
Princess	0	0	1

$$\text{King} = [1, 0, 0]$$
$$\text{Woman} = [0, 1, 0]$$
$$\text{Princess} = [0, 0, 1]$$

Problemi:

- Il numero di dimensioni incrementa linearmente con il numero di parole del vocabolario.
- La matrice è molto sparsa.
- Non c'è informazione condivisa tra le parole

Distributed Representation

Ogni termine del vocabolario T è rappresentato da un vettore di k valori reali. Le dimensioni del vettore possono essere osservate (e.g. hand-crafted features) o latenti (e.g. embedding dimensions).

Il numero di dimensioni k vogliamo sia molto piccolo rispetto alla dimensione del vocabolario $|C| = k \ll |T|$.

Questa rappresentazione distribuita ci permette di raggruppare parole simili in base al contesto considerato.

	Feminity	Youth	Royalty
Man	0	0	0
Woman	1	0	0
King	0	0	1
Princess	1	1	1
Child	0.5	1	0

$$\text{King} = [0.0, 0.0, 1.0]$$
$$\text{Woman} = [1.0, 0.0, 0.0]$$
$$\text{Princess} = [1.0, 1.0, 1.0]$$
$$\text{Child} = [0.5, 1.0, 0.0]$$

Parole simili hanno vettori simili quindi possiamo eseguire delle combinazioni lineari dei vettori per scoprire le relazioni tra le parole.

$$\text{King - Man + Woman} = \text{Queen}$$
$$[0, 0, 1] - [0, 0, 0] + [1, 0, 0] = [1, 0, 1]$$

Vantaggi:

- La matrice è molto meno sparsa.
- Vengono catturate e mantenute le relazioni tra parole.

Co-occurrence Matrix

Un'altra possibilità è quella di considerare un certo contesto limitando le occorrenze a un certo numero di termini vicini. Il contesto può essere espresso a diverse granularità: documenti, frasi, n-grammi.

Definiamo una matrice chiamata **window-based co-occurrence matrix** (oppure term-context matrix) che contiene il numero di volte che ogni *context word* co-occorre in una finestra di una specifica dimensione con una *target word*.

La finestra di dimensione k contiene le k parole a sinistra e le k a destra della parola in considerazione.

Data la seguente collezione di documenti. Prendiamo in considerazione come target words *magazine* e *newspaper*. Individuiamo le parole nella finestra di dimensione due per ogni target word.

- He is reading a magazine
- This magazine published my story
- She buys a magazine every month
- I was reading a newspaper
- The newspaper published an article
- He buys this newspaper every day

Costruiamo la matrice di co-occorrenza window-based

	reading	a	this	published	my	buys	the	an	every	month	day
magazine	1	2	1	1	1	1	0	0	1	1	0
newspaper	1	1	1	1	0	1	1	1	1	0	1

Per poter rappresentare ogni termine del vocabolario V la matrice di co-occorrenza deve essere calcolata su tutti i termini. Dal momento che ogni termine comparirà almeno una volta in qualche finestra anche le context word saranno tutti i termini. Quindi la matrice sarà composta da $|V| \times |V|$ elementi.

Questa matrice contiene vettori che sono semanticamente legati, ma non sono densi. E' necessario ridurre il numero di parole utili a rappresentare le varie parole.

PointWise Mutual Information (PMI)

Come già osservato con l'analisi di Luhn le frequenze assolute non sono una buona rappresentazione. Utilizziamo quindi una tecnica che assegna dei pesi a ogni coppia di parole term e context $\text{PMI}(w_i, c_j)$.

$$\text{PMI}(w_i, c_j) = \log_2 \left(\frac{P(w_i, c_j)}{P(w_i)P(c_j)} \right)$$

$$\text{PPMI}(w_i, c_j) = \max \left(\log_2 \left(\frac{P(w_i, c_j)}{P(w_i)P(c_j)} \right), 0 \right)$$

Sia f_{ij} il numero di volte che le parole w_i e c_j co-occorrono.

$$P(w_i, c_j) = \frac{f_{ij}}{\sum_i^W \sum_j^C f_{ij}} \qquad P(w_i) = \frac{\sum_i^W f_{ij}}{\sum_i^W \sum_j^C f_{ij}} \qquad P(c_j) = \frac{\sum_j^C f_{ij}}{\sum_i^W \sum_j^C f_{ij}}$$

6.2 Count-based models

Si calcolano delle statistiche su quanto spesso una parola co-occorre con le parole vicine in una grande collezione. Queste statistiche vengono mappate in un vettore denso di piccole dimensioni.

I vettori vengono imparati eseguendo una riduzione della dimensionalità della matrice term-context. La matrice viene fattorizzata a una matrice term-feature di dimensione ridotta in cui ogni parola è rappresentata da un vettore denso.

Alcuni modelli:

- Latent Dirichlet Allocation (LDA)
- Singular Value Decomposition (SVD)
- Global Vectors (GloVe)

6.2.1 SVD

Una qualsiasi matrice rettangolare W di dimensione $w \times c$ può essere espressa come prodotto di tre matrici $W = U \times S \times V^T$.

- U matrice $w \times m$ dove le righe w corrispondono alle righe di W e le colonne m rappresentano una dimensione (feature) in un nuovo **spazio latente**.
- S matrice $m \times m$ di singular values che esprimono l'**importanza** di ogni dimensione (feature).
- V^T matrice $m \times c$ dove le colonne c corrispondono alle colonne della matrice W e le righe m corrispondono ai singular values.

Possiamo ottenere un'approssimazione di dimensione minore tenendo soltanto k dei m singular values. La matrice U troncata contiene le k feature più importanti associate a tutte le parole presenti in W.

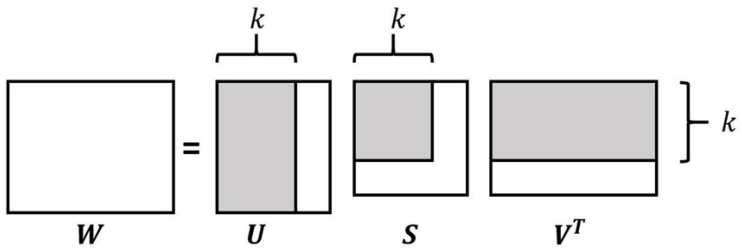


Figure 6.1: Troncamento della SVD [1].

Problemi:

- Se vogliamo aggiungere nuove parole dobbiamo ricalcolare la fattorizzazione perché cambia la dimensione della matrice.
- La matrice è comunque molto sparsa perché la maggior parte della parole non co-occorre.
- Costo quadratico per il calcolo della fattorizzata.

Una soluzione allo sbilanciamento delle frequenze può essere quella di ignorare le stop words, oppure pesare la co-occorrenza in base alla distanza delle parole nel documento.

6.2.2 GloVe

Sia X la matrice di term-context. Indichiamo con X_{ij} la frequenza di una parola j che occorre nel contesto della parola i, e X_i la frequenza di una qualsiasi parola di apparire nel contesto della parola i.

Definiamo la probabilità di co-occorrenza P_{ij}, ovvero la probabilità che la parola j occorra nel contesto della parola i.

$$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}, \qquad X_i = \sum_k X_{ik}$$

...

valutiamo il rapporto $\frac{P_{ik}}{P_{jk}}$

GloVe costruisce una funzione F che permette di imparare la rappresentazione.

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

Minimizziamo una funzione obiettivo J.

$$J = \sum_{i=1,k=1}^{V} f(X_{ik})(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log(X_{ik}))^2$$

6.3 Predictive Models

I modelli predittivi provano direttamente a predire una parola dai termini vicini.

Il modello più usato è il Word2Vec, una rete neurale poco profonda con un hidden layer che permette di ricostruire il contesto di una parola. Il raw text viene usato come training set di un approccio di apprendimento supervisionato.

Gli esempi di training vengono generati da coppie (target, context) scorrendo una finestra di dimensione fissa su tutta la collezione.

Sia gli input che gli output sono codificati usando la rappresentazione one-hot.

Dopo avere eseguito l'addestramento la matrice dei pesi W conterrà i word embedding del vocabolario V.

Il word embedding della parola V_i sarà rappresentato dal vettore di N elementi $W_i = (w_{(i,1)}, w_{(i,2)}, ..., w_{(i,N)})$.

Word2Vec può produrre il word embedding utilizzando due architetture:

- **CBOW**

  Predice la parola corrente (target word) dalle vicine (context word).

  Funziona bene con training set piccoli e rappresenta bene le parole rare.

- **Skip-gram**

  Predice le parole vicine (context word) dalla parola corrente (target word).

  Molto più veloce del modello Skip-gram nella fase di addestramento e ha un accuratezza leggermente migliore nelle parole frequenti.

Metodi di training:

- Hierarchical Softmax
- Negative sampling

Grazie alla rete neurale è possibile catturare dei pattern complessi oltre la similarità tra le parole.

Più il training set è grande ($>$ 10M parole) più le performance saranno migliori.

I testi devono includere più parole diverse possibili.

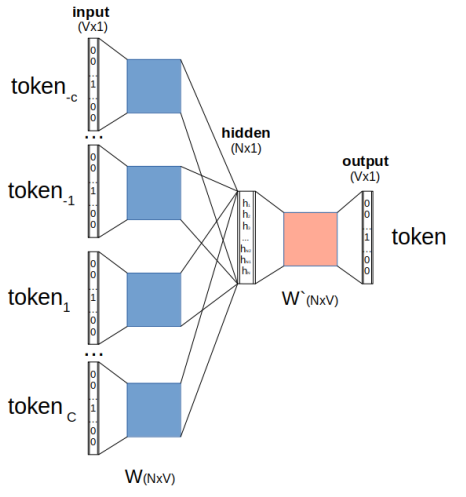


Figure 6.2: Schema della rete con architettura CBOW [2].

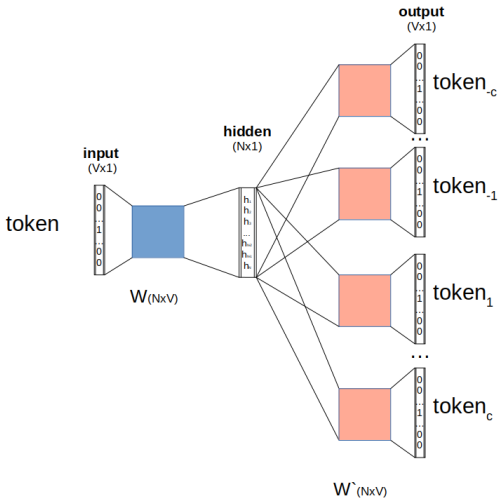


Figure 6.3: Schema della rete con architettura Skip-gram [3].

Bibliography

[1] Davide Chicco. Computational algorithms to predict gene ontology annotations - scientific figure on researchgate, 04 2015. [visited on 11/11/2021].

[2] Wikimedia Commons. File:cbow.png — wikimedia commons, the free media repository, 2020. [Online; accessed 12/11/2021].

[3] Wikimedia Commons. File:skipgram.png — wikimedia commons, the free media repository, 2020. [Online; accessed 12/11/2021].