

INFORMATION RETRIEVAL – TEXT MINING AND SEARCH

Sommario

1- INTRODUZIONE AL CORSO	2
2- INTRODUZIONE AL TEXT MINING	3
3- TEXT PROCESSING	5
4- INDEXING E TEXT REPRESENTATION	8
5- PART OF SPEECH TAGGING E NAMED ENTITY RECOGNITION.....	10
6- STATISTICAL LANGUAGE MODELING.....	13
7- WORD EMBEDDING.....	15
8- INTRODUCTION TO INFORMATION RETRIEVAL	21
9- DATA STRUCTURES FOR INDEXING	23
10- THE BOOLEAN MODEL AND THE VECTOR SPACE MODEL.....	25
11- LANGUAGE MODELS AS IR MODELS.....	27
12- KNOWLEDGE GRAPHS (NO LEZIONE)	28
13- CONTEXTUALIZED EMBEDDINGS - NEURAL LANGUAGE MODEL.....	29
14- NEURAL INFORMATION RETRIEVAL (RAGANATO)	31
15- CONTEXTUAL SEARCH AND PERSONALIZATION (NO LEZIONE).....	32
16- WEB SEARCH	33
17- EVALUATIONS.....	36
19- LABORATORIO 1	37
20- LABORATORIO 3	37
21- LABORATORIO 3	38
22- LABORATORIO 4	Errore. Il segnalibro non è definito.

1- INTRODUZIONE AL CORSO

Il focus principale del corso riguarderà il testo, infatti i problemi che saranno trattati riguarderanno l'analisi, il processing e la rappresentazione dei dati in forma testuale. Nella prima parte del corso che arriverà fino ad ottobre verrà spiegato cosa sia il text mining e come esso si relaziona all'information retrieval. Dopo ottobre **i due corsi saranno diversificati**.

Seguono gli argomenti trattati nel corso suddivisi in parte comune e parti diversificate:

PARTE COMUNE A TEXT MINING E INFORMATION RETRIEVAL:

1. **Definizione** di Text Mining
2. **Differenze** tra Text Mining e Data Mining
3. **Principali operazioni** del Text Mining
 - a. Text classification, text clustering, text summarization
 - b. Information retrieval, information filtering
4. **Text pre-processing**
5. **Indexing e Text Representation**

PARTE DIVERSIFICATA RIGUARDANTE IL **TEXT MINING E TEXT SEARCH**:

1. **Text Mining tasks**
 - a. Topic modeling
 - b. Text classification
 - c. Text clustering
 - d. Text summarization
2. **Introduzione all'information retrieval**
 - a. Motori di ricerca testuali
 - b. Motori di ricerca web
3. **Open Source software per il text mining e text search**
4. **Laboratorio**

PARTE DIVERSIFICATA RIGUARDANTE **INFORMATION RETRIEVAL**:

1. **Modelli di Information Retrieval**
2. **Motori di ricerca web**
3. **Valutazione dei motori di ricerca**
4. **Advanced topics**
5. **Laboratorio**

L'esame sarà costituito da uno scritto con una serie di domande riguardanti gli argomenti trattati nel corso e la presentazione di un progetto di gruppo fatto da massimo 3 persone.

2- INTRODUZIONE AL TEXT MINING

Una prima **definizione di Text Mining** risale al 2002 formalizzata da Sebastiani: il termine text mining è genericamente usato per individuare un qualunque **sistema che analizza una grande quantità di testo in linguaggio naturale e identifica dei pattern lessicali o linguistici andandone ad estrarre delle informazioni utili.**

Il text mining viene **utilizzato per una grande varietà di operazioni** tra cui:

- **Sentiment analysis:** analisi del testo per **identificare opinioni** generalmente riguardanti un prodotto da parte dei consumatori. Da questa analisi è possibile individuare punti di forza e debolezza di un prodotto nonché adeguare il prodotto alle richieste dei consumatori.
- **Document summarization:** metodo utilizzato **per riassumere grandi quantità di testo in un testo più breve** contenente le informazioni principali del testo originale.
- **Raccomandazioni di ristoranti/hotel/news:** processi correlati alla sentiment analysis
- **Analisi del testo per informazioni finanziarie:** processi correlato alla sentiment analysis

DIFFERENZE TRA TEXT MINING E DATA MINING

Il **Data Mining** si caratterizza dalla **capacità di estrarre informazioni implicite, precedentemente sconosciute** e potenzialmente utili da dei dati di vario tipo. Il **Text Mining** d'altro canto si occupa di **estrarre delle informazioni che sono chiaramente e esplicitamente presenti nel testo**. Se quindi per il **data mining** le **informazioni devono essere scoperte** dall'analisi e scoperta di pattern all'interno di un grande set di dati, per il **text mining** l'operazione si limita ad **estrarre dal testo stesso le sue informazioni più importanti**.

PRINCIPALI FUNZIONI DEL TEXT MINING

Il Text Mining può essere **applicato su molte tipologie di testo** come ad esempio il testo: delle email, dei siti web, dei contratti, delle pubblicazioni scientifiche, degli articoli di news, ecc..

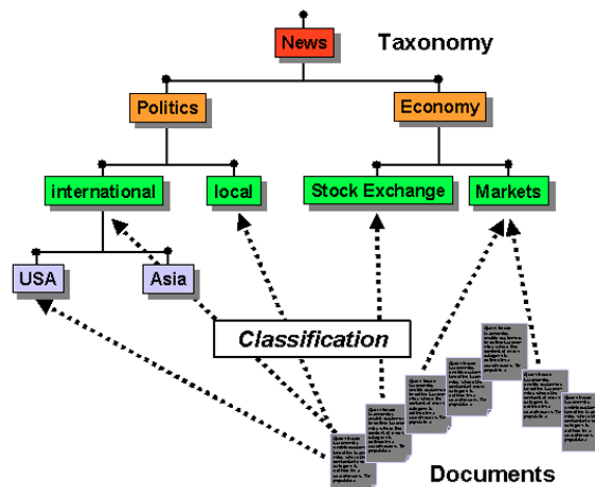
Le **principali sfide** che deve affrontare il Text Mining sono: **la difficoltà dei computer di elaborare il testo** dei documenti in forma non strutturata, **gestire una grande quantità di documenti testuali**, gestire dei dati **semi-strutturati o non strutturati**, affrontare le **ambiguità che il linguaggio naturale contiene** (ambiguità sintattica, lessicale, semantica e pragmatica).

Il Text Mining è strettamente correlato alla ricerca dati, quindi **all'Information Retrieval**. Questa pratica consente di trovare ad esempio più facilmente dei contenuti all'interno del web, in quanto **si fa una richiesta specifica** e i contenuti rilevanti vengono "minati" dal web per **individuare dei risultati utili**. Il processo viene metaforizzato con la tecnica di estrazione dell'oro dato che **l'utente tramite una query va a specificare quali corrispondenze debbano essere presenti all'interno dei documenti analizzati**.

Le principali funzioni influenzate dal Text Mining sono:

- **Text Summarization:** un text summarized si occupa di andare a **produrre una rappresentazione condensata del suo input** per renderlo più fruibile per un lettore.
- **Information Retrieval:** **dati un insieme di documenti testuali** e una richiesta di informazioni da parte di un utente espressa sotto forma di query, l'IR si occupa di **identificare e restituire i documenti più rilevanti tramite la query dell'utente**. Spesso questa funzione viene utilizzata in concomitanza della Text Summarization che, **nelle ricerche web, restituisce una breve sintesi del contenuto dei documenti individuati dalla IR**.
- **Content Based Recommender Systems:** contenuti di tipo testuale prodotti come uno stream di testo che sono proposti all'utente **sulla base delle sue preferenze personali**.

- **Text Classification:** assegnazione di documenti in linguaggio naturale a delle categorie predefinite sulla base del loro contenuto. Ha quindi il compito di analizzare testi in linguaggio naturale per determinarne delle categorie predefinite sulla base del loro contenuto. L'algoritmo determina a quale classe il testo verrà associato.
- **Document Clustering:** il clustering di documenti si basa su apprendimento non supervisionato e quindi si occupa di andare a raggruppare dei documenti che condividono topic simili senza però utilizzare una categoria predefinita.
- **Document classification:** si occupa di classificare dei documenti, ovvero delle unità di informazioni multimediali (web page ad esempio). Il testo contenuto in questi documenti può essere strutturato o non strutturato, con strutturato si intende un testo delimitato da delle sezioni (come ad esempio i capitoli di un libro). Questa pratica rientra nel text classification e non nel clustering in quanto si utilizzano delle classi predefinite come si vede nella immagine con Politics, Economy e le sottoclassi.



- **Text clustering:** sottocategoria di document clustering. Si va ad analizzare un text corpus/insieme di documenti testuali/repository (sono tutti sinonimi). L'algoritmo non supervisionato crea dei cluster di documenti con delle somiglianze. A posteriori, quindi dopo l'esecuzione dell'algoritmo, sta a noi andare ad associare una descrizione significativa ad ogni cluster individuato.
- **Topic modeling:** tecnica supervisionata che si occupa di identificare dei topics nel testo.
- **Mining structured data within texts:** ci sono testi nei quali alcune entità sono menzionate, come ad esempio nome di città, persone, ecc. Questa tecnica consente di identificare questo tipo di entità all'interno di un testo. Se ad esempio in un testo ho un nome di persona e il nome della città in cui è nato, grazie a questi algoritmi è possibile andare a compilare in maniera automatica una tabella che associa la città di nascita al nome di persona individuato nel testo.
- **Mining structured text:** le pagine web contengono del linguaggio strutturato (XML). Per analizzare questo tipo di testo si utilizzano i parser che sono degli analizzatori sintattici comunemente chiamati wrappers per analizzare la struttura della pagina. Il modo in cui viene analizzata la pagina è molto semplice: data la sua struttura, in una pagina web è possibile andare ad analizzare ad esempio solo i titoli delle pagine, oppure solo il primo "capitolo" dato che, essendo una pagina strutturata, è possibile identificare il limite che divide in titolo, capitolo 1, 2, ecc una pagina. Questi analizzatori sintattici vanno ad analizzare il contenuto del testo andando a scremare ad esempio i tag HTML che ti fanno diventare il testo in grassetto oppure i tag che identificano un capitolo o un altro.

3- TEXT PROCESSING

Il text processing è spesso il primo step del text mining e consiste in diversi livelli di semplice processing del testo andando ad eseguire operazioni quali tokenizzazione, lemmatizzazione, normalizzazione o altre operazioni più avanzate.

La finalità dell'analisi del testo può essere sia predittiva o esplorativa:

- **Analisi predittiva del testo:** sviluppare delle funzionalità in grado di riconoscere automaticamente o individuare un concetto particolare all'interno di un segmento di testo.
- **Analisi esplorativa del testo:** sviluppare delle funzionalità in grado di scoprire dei pattern o trends di particolare utilità e interesse all'interno di un segmento di testo.

Andando ad analizzare più nel dettaglio alcune tecniche di analisi predittiva abbiamo:

- **Opinion Mining:** si occupa di individuare un'opinione positiva o negativa riguardo un particolare oggetto all'interno di un testo
- **Sentiment analysis:** si occupa di identificare lo stato emotivo dell'autore facendo uso di classi predefinite che definiscono un set di emozioni sulla base del contenuto del testo.
- **Bias detection:** si occupa di identificare se un testo dato appartiene a una tra due categorie predefinite opposte tra loro come ad esempio il partito politico di destra o di sinistra.
- **Information extraction:** si occupa di identificare in maniera automatica all'interno di una sequenza di parole la presenza di particolari entità.
- **Relation Learning:** si occupa di individuare delle coppie di entità che condividono una particolare relazione. Se la coppia di parole che condividono una relazione sono ad esempio il nome di una persona e il nome di un'azienda, se facessimo una ricerca su internet con un nome di persona e una azienda otterremmo informazioni riguardanti il ruolo ricoperto da tale persona all'interno di tale azienda che corrispondono alla relazione che collega tra di loro le due entità passate in input. L'obiettivo della relation learning quindi quello di avere del testo e di andare a creare delle relazioni tra le entità presenti all'interno dello stesso.
- **Text driven forecasting:** si occupa di analizzare uno stream continuo di testo per eseguire delle previsioni su dei trend correlati ad un particolare argomento. Alcune applicazioni di questo task va a vedere ad esempio l'indice di gradimento di un politico oppure di un prodotto al suo rilascio.
- **Temporal summarization:** si occupa di creare un sommario a partire da uno stream di testo continuo. Quindi simile allo snippet dei motori di ricerca ma con la differenza che non si va a vedere un testo che non cambia, ma uno stream di testo in continuo aggiornamento e continuamente alimentato. Quindi si va a monitorare tutti i testi in arrivo e si va a prevedere se una data frase deve essere inclusa o meno in un sommario per descrivere un particolare evento.

TEXTUAL DOCUMENTS

In genere i documenti e il testo sono tra loro differenti in quanto un documento contiene delle informazioni di diverso formato come ad esempio immagini, audio, video e testo. I documenti testuali contengono al loro interno solo dei dati di tipo testuale e quello che si vuole ottenere con il text processing è di andare a lavorare con il contenuto di questi testi presenti nei documenti.

I documenti testuali possono avere diversi formati e quello più semplice da analizzare è il free-format text identificabile ad esempio con i file .TXT. Questi tipi di file sono ottimi in quanto è possibile identificare immediatamente e in maniera automatica una stringa di caratteri appartenenti ad un alfabeto che vanno a comporre le parole del testo, ovvero individuare e riconoscere le parole che compongono il testo.

I documenti possono avere diverse caratteristiche che ne definiscono il contenuto informativo. I documenti in genere possono essere composti da:

- Solo **testo**
- Testo e **struttura**: strutturato ad esempio con un abstract, titolo, capitoli ecc.
- Testo, struttura e **altri media**: altri media quali immagini, audio, video
- Testo, struttura, altri media e **metadati**: i metadati sono attributi esterni che vanno a definire delle caratteristiche del file in sé. I metadati possono essere di due tipi, **descrittivi** che **contengono dati riguardanti la creazione del documento** (titolo, data di creazione, numero di pagine) oppure di tipo **semantico che contengono dati riguardanti il contenuto del documento**.

I documenti possono anche avere **diversi formati** che possiamo dividere in tre categorie:

1. Documenti scritti con dei comuni **word-processor** come Word, HTML, XML
2. Formati per la **visualizzazione e la stampa** come i PDF, MIME
3. Formati **compressi** come i ZIP, RAR

TEXT PROCESSING

Il primo problema che ci si pone quando si parla di text processing è quello di **come andare a rappresentare** un documento testuale.

Una prima tecnica di rappresentazione è la **Bag-of-Words**. Questa tecnica va a **rappresentare i termini come gli elementi base del testo**; in particolare **ogni parola** presente nel testo analizzato **viene inserita come colonna di una matrice** le cui **righe rappresentano i documenti analizzati** e **dove ogni cella contiene il numero di ricorrenze** della parola all'interno del documento analizzato.

Rappresentare un testo in tal modo senza compiere alcuna altra azione potrebbe produrre dei risultati poco significativi data la natura eterogenea del linguaggio naturale. Risulta quindi necessario applicare alcuni passaggi di pre-processing dei dati che sono:

- **Tokenizzazione**: **dividere la sequenza di caratteri in dei token** che identificano le parole
- **Normalizzazione**: **ridurre alla stessa forma** parole uguali eliminando così la ridondanza
- **Stemming e Lemming**: **ridurre alla radice** le parole con lo stesso significato (authorization e authorize ricondotte a quest'ultima)
- **Rimozione delle Stop Words**: **omissione di alcune parole molto comuni** come avverbi e congiunzioni

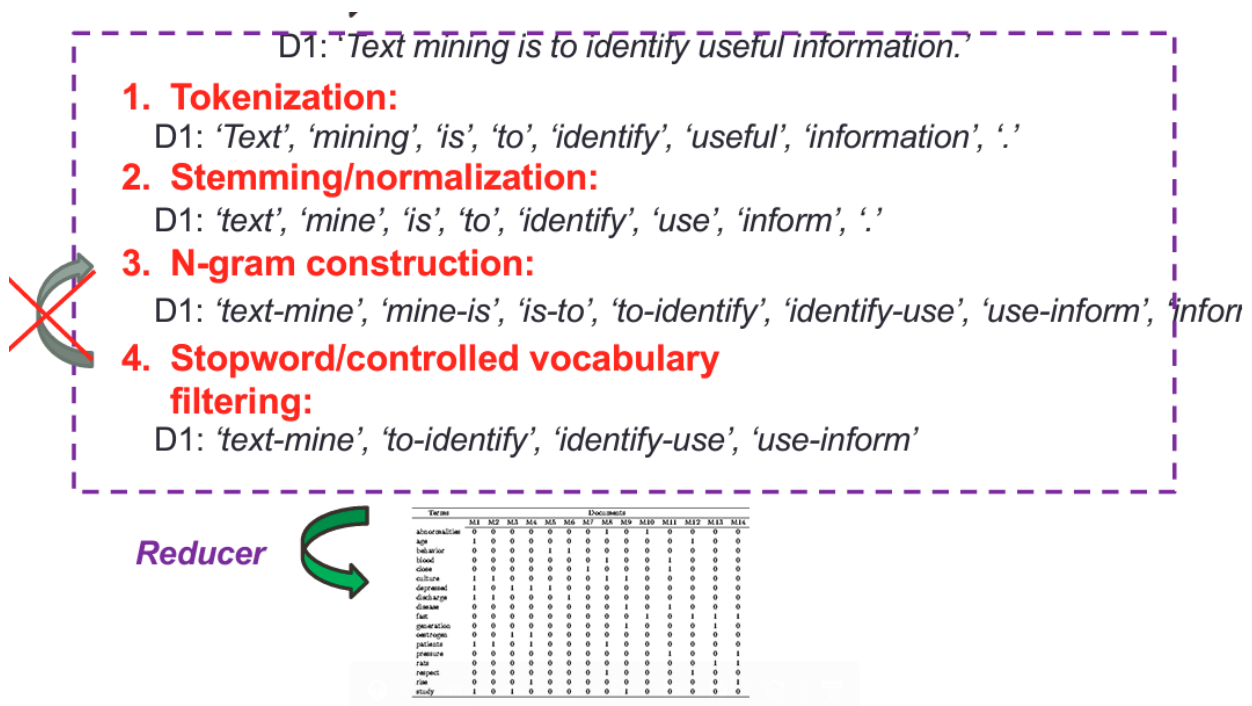
Approfondendo le tecniche appena citate iniziamo con la tokenizzazione. Un token è una **istanza di una sequenza di caratteri** che va a definire un'entità all'interno di una frase. Definire correttamente un token tuttavia risulta complesso in quanto **il linguaggio naturale presenta diverse complicazioni di comprensione**; ad esempio la parola San Francisco che definisce l'omonima città viene considerata come due token se per identificare i token utilizzassimo gli spazi come separatori. Un altro grosso problema che si riscontra durante il processo di tokenizzazione riguarda **la lingua del testo**. Per la lingua italiana ad esempio molto spesso abbiamo l'articolo abbreviato utilizzando l'apostrofo e questo non è facilmente riconoscibile come token dato che la parola viene vista come unica data l'assenza di spazi. Allo stesso modo tutta la lingua **cinese e giapponese non fa utilizzo di spazi** nelle frasi rendendo così impossibile una tokenizzazione che si basa sull'utilizzo degli spazi, oppure la lingua araba che si scrive da destra verso sinistra. Per **risolvere questi problemi** possiamo **definire manualmente delle espressioni regolari per le eccezioni** che vogliamo gestire in modo diverso, **oppure utilizzare i metodi automatici degli NLP**.

Ritornando al concetto di **Stop Words** possiamo dire che il pre-processing di questa tecnica si occupa di **andare ad escludere dal dizionario** dei termini generati dalla tokenizzazione **tutte quelle parole comuni prive di contenuto informativo** come ad esempio gli articoli o avverbi. Tuttavia **i motori di ricerca web non fanno uso della rimozione delle Stop Words** in quanto questo potrebbe comportare una errata interpretazione del senso della frase nella quale si presentano questi tipi di parole.

Parlando di **normalizzazione** possiamo dire che ci sono vari casi in cui questa tecnica è molto utile ad esempio quando **due parole differiscono tra loro per delle lettere accentate**; spesso capita che gli utenti non utilizzino gli accenti anche delle parole che normalmente ne prevedono la presenza quindi ricondurre due parole uguali ma che differiscono per un accento può essere utile. Un altro esempio può essere quello del **formato delle date** dove gli americoglioni mettono il mese per primo e il giorno per secondo. Per risolvere questi problemi nella normalizzazione spesso si fa uso delle **case folding** che sostanzialmente vanno **a ricondurre tutte le lettere che compongono le parole in maiuscolo o minuscolo** per evitare la tokenizzazione di due termini effettivamente uguali nel significato, oppure si fa uso dei **thesauri** che consentono di **individuare casi di sinonimia e omonimia tra parole**.

Concludiamo approfondendo **lo stemming e la lemmatization**. Si dice **lemmatizzazione** quel processo di **riduzione delle parole alla loro forma base**: ad esempio per i verbi questi possono essere ricondotti al loro tempo infinito. Per quanto riguarda lo **stemming** invece si tratta di un **processo che riconduce i termini alla loro radice andando a "tagliare" le parole**: ad esempio automatic e automate e automation vengono tutte ricondotte ad automat. In generale lo stemming viene usato per far fronte al problema dei plurali, avverbi. Una soluzione di stemming è il **Porter Stemmer**; questo algoritmo utilizzato per la **lingua inglese** consiste in **diverse fasi di riduzione delle parole applicate sequenzialmente che si basano sul riconoscimento di alcuni pattern linguistici**. L'utilizzo dello stemming non sempre porta a dei guadagni in termini di risultati in quanto potrebbero ridurre notevolmente la precisione, ma in alcune lingue come il tedesco risulta molto utile.

Ricapitolando il processo di pre-processing del testo segue un **esempio di applicazione delle tecniche appena descritte per ottenere una rappresentazione Bag-of-Words** soddisfacente:



4- INDEXING E TEXT REPRESENTATION

Le modalità di rappresentazione de testo sono varie e distinte tra loro. La più semplice rappresentazione del testo è una **rappresentazione binaria** che **associa i termini/parole ad un documento**; si avrà quindi una matrice di incidenza con i documenti sulle colonne e i termini sulle righe e nelle celle **1 o 0 a seconda se il termine è presente o meno** all'interno del documento. In tal modo **ogni documento può essere anche rappresentato con l'insieme dei termini che lo compongono**. Una seconda semplice rappresentazione è la count matrix, ovvero una matrice che conta il numero di occorrenze di un termine all'interno di un documento. In questo caso ogni documento viene **rappresentato da un vettore che identifica il numero di occorrenze dei termini** all'interno dello stesso.

Una delle rappresentazioni più utilizzata attualmente è la **Bag of Words**. Questa rappresentazione ha la particolare caratteristica di **generare un vettore di rappresentazione che non considera l'ordine delle parole** all'interno di un documento; se avessimo ad esempio le frasi "Marco ama Ylenia" e "Ylenia ama Marco" queste sarebbero rappresentate dallo stesso vettore nonostante il loro ordine sia diverso. In tal modo la rappresentazione più semplice di Bag of Words non tiene conto della dipendenza tra termini consecutivi né tantomeno dell'ordine dei termini all'interno della frase.

Per far fronte a questo problema si utilizza la rappresentazione **Bag of Words con N-grams**. N-grams **definisce una sequenza continua di N token a partire da un testo**, consentendo quindi di prendere in considerazione coppie di parole o anche terne, quaterne, ecc... Questa strategia se da un lato consente di prendere in considerazione le dipendenze tra termini consecutivi e considerarne l'ordine, dall'altro lato incrementa notevolmente la grandezza del vocabolario.

ZIP'S RULE AND TF-IDF

Analizzare il linguaggio naturale può essere un compito difficile, in particolare quando si va ad attribuire un certo valore sulla base della frequenza dei termini. Il problema principale è che **nel linguaggio naturale ci sono pochi termini frequenti e pochissimi termini quasi inutilizzati**; ad esempio le congiunzioni sono usate in ogni frase e quindi saranno molto frequenti e quasi sempre presenti, allo stesso modo le parole complesse, specifiche di un settore come zeugma non vengono quasi mai utilizzate.

La legge di Zip descrive la frequenza di una parola sulla base di un suo grado che viene determinato dalla posizione di tale parola all'interno della lista delle parole ordinate in modo decrescente. In parole povere prende in considerazione la rarità di tale parola oltre alla frequenza della stessa nella frase analizzata.

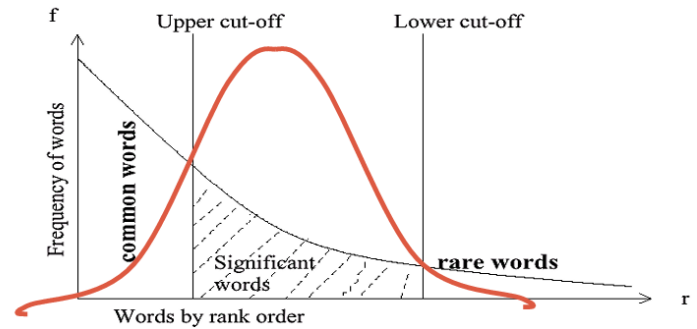
Tale legge si applica nel seguente modo:

Data una collezione di documenti, **si dispongono le words (w) in ordine decrescente in base alla loro frequenza $f(w)$** . La legge di Zip dunque dice che il prodotto della frequenza d'uso delle parole e la loro posizione in classifica è pressoché costante. Quindi **la frequenza di una parola $f(w)$ è proporzionale a $1/r(w)$ dove r è il ranking della parola nella classifica**. La legge di Zip in conclusione mette in risalto come le **parole che hanno un grande numero di occorrenze sono in realtà poco significative**, mentre le parole con meno occorrenza possiedono un alto quantitativo informativo.

Frequent Word	Number of Occurrences	Percentage of Total
the	7,398,934	5.9
of	3,893,790	3.1
to	3,364,653	2.7
and	3,320,687	2.6
in	2,311,785	1.8
is	1,559,147	1.2
for	1,313,561	1.0
The	1,144,860	0.9
that	1,066,503	0.8
said	1,027,713	0.8

Frequencies from 336,310 documents in the 1GB TREC Volume 3 Corpus
125,720,891 total word occurrences; 508,209 unique words

Se da un lato la legge di Zip ci fornisce delle importanti informazioni dall'altro è necessario andare a **determinare delle soglie in grado di distinguere quando una parola può essere ritenuta significativa o meno.** Ovviamente le parole con altissima frequenza, così come anche le parole con bassissima frequenza non sono significative e quindi andrebbero scartate. Per fare ciò **si creano un upper cut-off e un lower cut-off** per delimitare il range in cui le parole sono considerate significative.



Sulla base dell'analisi di Luhn sono stati **proposti dei pesi da assegnare alle parole per determinarne il contenuto informativo:** misurare il contenuto informativo della parola in relazione al contenuto del documento in cui compare e quale sia l'informazione apportata da quel termine all'interno di altri documenti. Queste due **misure euristiche** sono la **Term Frequency (TF)** e la **Inverse Document Frequency (IDF)**.

La TF è la **frequenza di un termine t all'interno di un documento d rispetto alla frequenza dello stesso termine all'interno di tutti gli altri documenti analizzati:** $w(tf) = tf / \max tf$.

Per quanto riguarda la IDF possiamo dire che per un **termine t** , dato il numero di documenti che contiene tale termine df , si ha che: **$idf = \log (N/df)$** dove **N è il numero totale di occorrenze della parola.**

In conclusione possiamo dire che **il peso TF-IDF** di un termine è il prodotto del suo peso TF e IDF:

$$w = (tf / \max tf) * \log(N/df)$$

Dalla formula si può evincere come **il peso di un termine cresca al crescere delle occorrenze** dello stesso nei documenti e che **cresca anche in base alla rarità del termine** all'interno dei documenti.

5- PART OF SPEECH TAGGING E NAMED ENTITY RECOGNITION

Dopo aver visto le fasi preliminari di text processing e di rappresentazione del testo, soffermandoci sul concetto di pesatura dei termini e di posizione degli stessi, vediamo ora come **la rappresentazione del testo può essere arricchita con delle semplici tecniche di NLP**.

La struttura complessa dei testi rende non sufficiente una semplice rappresentazione tramite BoW per comprenderne il significato appieno. Le tecniche di Natural Language Processing danno la possibilità di scoprire ulteriori informazioni sul significato del testo con funzioni come:

- **Part of Speech tagging (POS)**
- **Named Entity Recognition (NER)**
- Analisi sintattica
- Analisi semantica
- Struttura del discorso

Analizzare le frasi in generale consente di ottenere più informazioni e avere meno ambiguità. Per analizzare le frasi ci sono tre diversi possibili approcci:

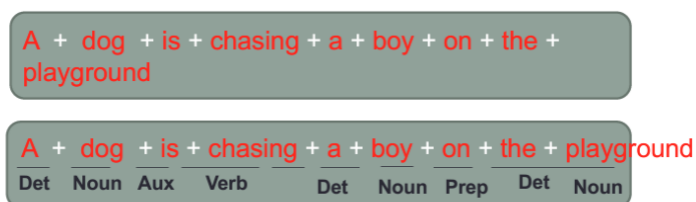
1. Utilizzare Word N-gram come visto prima
2. **Identificare il ruolo sintattico delle parole all'interno delle frasi utilizzando un POS tagger**
3. Memorizzare le posizioni delle parole ed utilizzare query di prossimità (metodo usato dai search engine)

Sia **POS** che **NER** sono dei modelli che, data una sequenza di parole in input, si pongono come obiettivo quello di **definire un modello che produce in output una sequenza di tag (label)**. Per ottenere questo risultato si fa uso o di un **modello rule-based** o di un **modello di apprendimento supervisionato**.

I modelli di **apprendimento supervisionato** consistono in un **training set** composto da coppie di input e label associata; l'obiettivo del modello sarà dunque quindi quello di **individuare una funzione che mappi ogni input x in una label y**. L'approccio utilizzato per definire tale funzione è il **modello generativo**. Questo modello va a stimare la probabilità congiunta $p(x,y)$ per le possibili coppie (x,y) .

PART OF SPEECH TAGGING (POS)

Una volta applicate le fasi preliminari di text processing, il POS tagging va ad affidare ad ogni parola **all'interno di un testo un tag che corrisponde alla parte del discorso identificata da tale parola**. Avremmo quindi tag per i nomi, avverbi, verbi, ecc...



I tag associati alle parole vengono definiti **Word Classes** o **POS** ovvero delle **classi di parole che hanno delle caratteristiche comuni** che sono: nomi, verbi, pronomi, aggettivi, preposizione, avverbio, articolo, congiunzione, interiezione. **Queste classi a loro volta sono suddivise in Closed class words e Open class words**; le **prime** sono solitamente **parole frequenti con una funzione puramente grammaticale** come gli articoli, i pronomi e le preposizioni, mentre **le seconde sono parole con un contenuto informativo** come i nomi, verbi, aggettivi e avverbi. Il POS tagging quindi è quel processo che data una sequenza di x parole in input, associa ad esse un tag y.

Per eseguire un POS tagging bisogna innanzitutto **selezionare un set di tag che può essere ristretto oppure più completo**. È evidente come **sia possibile che molte parole in frasi diverse possano assumere diversi tag** come quello di nome o verbo per la parola *PESCA*. Il problema principale a cui deve far fronte il POS è quindi quello di determinare il tag esatto per una parola all'interno di una frase. **Per disambiguare** quei casi di tag si utilizzano delle tecniche di tagging avanzate come il **Rule-based Tagging** o gli algoritmi di apprendimento supervisionato.

L'approccio **Rule-Based** si articola nel seguente modo:

- Si inizia **selezionando un dizionario**
- **Si assegna tutti i possibili tag** alle parole utilizzando il dizionario
- **Si scrivono manualmente le regole** per rimuovere i tag errati
- **Si eliminano i tag** inappropriati mantenendo il tag corretto per ogni parola

NAMED ENTITY RECOGNITION (NER)

La NER è un sub-task molto importante che ha come scopo quello di **individuare e classificare in un set predefinito di categorie i nomi nei testi** (Marco = persona, eBay = organizzazione, Como = località). Vi sono diversi approcci per applicare la NER che sono:

1. **Rule-based**: **utilizza dei lessici**, ovvero delle liste di parole e frasi, **che categorizzano i nomi**. **Utilizza anche delle regole** che possono essere scritte manualmente **per disambiguare o individuare nuovi tipi di entità**.
2. **Statistical machine learning**: **usa un modello probabilistico delle parole intorno all'entità** e stima le probabilità utilizzando dei dati di training. Un esempio di approccio di questo tipo è l'**Hidden Markov Model (HMM)**.

L'approccio HMM risolve l'ambiguità in una parola utilizzando il contesto all'interno del quale essa è inserita. Il contesto viene modellato **utilizzando un modello generativo della sequenza di parole**.

L'approccio HMM come da nome si basa sul potenziare il concetto di catene di Markov. Le **catene di Markov descrivono un processo come una sequenza di stati con delle transizioni che fanno passare da uno stato all'altro**; **ogni transizione ha una sua probabilità** associata e lo stato successivo dipende dallo stato attuale e dalle probabilità delle transizioni.

Markov assume quindi che: $P(q_i = a | q_1 \dots q_{i-1}) = P(q_i = a | q_{i-1})$

Dove: **q sono gli stati** che possono essere attraversati, **a sono le probabilità** che si hanno di passare da uno stato all'altro.

La catena di Markov quindi è utile per individuare la probabilità di una sequenza di eventi osservabili, ma spesso **gli eventi a cui siamo interessati sono nascosti**, ovvero non direttamente osservabili. Nei testi ad esempio vediamo le parole e non i tag che ne definiscono la part of speech che sono definiti quindi **hidden tags**. La HMM quindi **va ad identificare la sequenza di label che hanno la più alta probabilità all'interno della frase** ad esempio:

- Fred Smith, who lives at 10 Water Street, Springfield, MA, is a long time collector of tropical fish.
- <start><name><not-an-entity><location><not-an-entity><end>

NATURAL LANGUAGE PROCESSING (NLP)

Il Natural Language Processing è una serie di analisi e metodi applicati ad una **stringa affinché ne si comprenda il senso**. I tipi di analisi che vengono eseguiti nell'NLP sono:

- Analisi **morfologiche**: analizza il **significato di ogni singola parola** che compone la stringa
- Analisi **sintattiche**: analizza **come le parole di una stringa si relazionano tra loro**
- Analisi **semantiche**: analizza il **significato di parole combinate**
- Analisi **pragmatiche**: analizza il meta-significato delle parole, ovvero il **significato delle parole in relazione al contesto in cui esse sono inserite**
- Analisi **discorsive**: **analizza una grande porzione di testo**
- Analisi di **inferenza**: **inferisce un senso all'intero testo analizzato**

Le analisi appena presentate non sempre è possibile eseguirle con dei risultati soddisfacenti e soprattutto sono difficili da implementare. **Ciò che rende difficile implementare le analisi di NLP sono le ambiguità** che il linguaggio naturale contiene e che possono stravolgere il senso di una frase se interpretate non correttamente. Vi sono diversi tipi di ambiguità tra cui:

- **Ambiguità di parola**: la parola PESCA ha diversi significati (frutta, azione)
- **Ambiguità sintattica**: la frase prendi il gatto con i guanti è ambigua in quanto non sappiamo se è il gatto ad indossare i guanti o noi
- **Risoluzione di anafora**: la frase Marco convince Massimo a comprare una tastiera per lui è ambigua in quanto non sappiamo se lui si riferisce a Marco o Massimo
- **Presupposizione**: la frase ha smesso di fumare presuppone che qualcuno in passato abbia fumato

Vediamo infine come viene implementata una pipeline di NLP:

1. **Syntactic parsing**: analisi grammaticale di una frase data che individua la struttura grammaticale più probabile per la frase

A + dog + is + chasing + a + boy + on + the + playground
Det Noun Aux Verb Det Noun Prep Det Noun

2. **Relation extraction**: identifica la relazione tra le entità presenti nella frase

Its initial **Board of Visitors** included **U.S.**
Presidents Thomas Jefferson, James
Madison, and James Monroe.

1. Thomas Jefferson Is_Member_Of
Board of Visitors
2. Thomas Jefferson Is_President_Of
U.S.

3. **Logic inference**: converte pezzi di testo in delle rappresentazioni più formali

Its initial **Board of Visitors** included **U.S.**
Presidents Thomas Jefferson, James
Madison, and James Monroe.

$\exists x (Is_Person(x) \ \& \ Is_President_Of(x, 'U.S.') \ \& \ Is_Member_Of(x, 'Board\ of\ Visitors'))$

Tra le varie applicazioni del NLP abbiamo alcune di particolare interesse per il corso che sono: text clustering, text classification, text summarization, topic modeling.

6- STATISTICAL LANGUAGE MODELING

Un testo può essere rappresentato come un language model per rappresentare i suoi topics. Le parole che tendono ad essere presenti quando si parla di un particolare topic avranno un'alta probabilità all'interno del corrispondente language model. Il LM quindi va ad assegnare delle probabilità a delle sequenze di parole tramite un approccio generativo.

Ci sono diverse motivazioni per cui il LM è utile come ad esempio:

- **Machine Translation**: modificare delle parole di una frase utilizzando quelle con probabilità più alta per correggere un errore. ES. $P(\text{forti venti stanotte}) > P(\text{larghi venti stanotte})$
- **Spell Correction**: correggere gli errori di battitura. ES $P(\text{ciao mi chiamo Marco}) > P(\text{caio mi chaiom Marco})$
- **Speech Recognition**: riconoscere le parole che formano una frase e individuarne il senso complessivo. ES. $P(\text{ho visto un cane}) > P(\text{macchina ho del pagano})$
- **Text categorization**: associare una categoria al testo in base ai termini utilizzati al suo interno. Se ad esempio in una frase leggiamo tante volte la parola calcio e pallone allora sicuramente è più probabile che si parli di sport che di politica.
- **Information retrieval**

Il Language Model ha come obiettivo principale quello di individuare la probabilità di una frase o di una sequenza di parole $P(W) = P(w_1, w_2, w_3, w_4, w_5 \dots w_n)$ e di individuare anche la probabilità di una parola successiva a quelle appena considerate $P(w_n | w_1, w_2 \dots w_{n-1})$.

CATENE DI MARKOV

Consideriamo la probabilità $P(w_n | w_1, w_2 \dots w_{n-1})$, ovvero la probabilità di avere una parola w_n data una sequenza di parole che va da w_1 a w_{n-1} . Per calcolare questa probabilità, dato un grande corpus composto da diverse frasi, basta semplicemente contare quante volte all'interno dell'intero corpus si ha la sequenza di parole $w_1 \dots w_{n-1}$ seguita dalla parola w_n .

Se volessimo invece calcolare la probabilità $P(W)$ di una frase W , dovremmo ricorrere al calcolo della probabilità congiunta delle parole che compongono la frase W facendo uso della regola delle catene di Markov. Sulla base della definizione di probabilità condizionate $p(B | A) = P(A, B) / P(A)$ da cui si ricava $P(A, B) = P(A)P(B | A)$ possiamo dire che allora per la regola delle catene si ha che $P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2 | x_1)P(x_3 | x_1, x_2) \dots P(x_n | x_1, \dots, x_{n-1})$. Facendo un esempio con una frase si ha che:

$P(\text{"its water is so transparent"}) = P(\text{its}) \times P(\text{water} | \text{its}) \times P(\text{is} | \text{its water}) \times P(\text{so} | \text{its water is}) \times P(\text{transparent} | \text{its water is so})$

L'assunzione di Markov va a semplificare tale formula affermando che:

$P(\text{the} | \text{its water is so transparent that}) \approx P(\text{the} | \text{that})$ questo significa che non serve tenere in considerazione tutte le parole che compongono la frase ma solo l'ultima.

N-GRAMS

Un'altra tecnica di Language Modeling sono gli **N-grams language models**. In genere si applicano dei modelli **unigram**, **bigram** o al massimo **trigram**. Se per gli unigram language model la probabilità delle parole non dipende dalle parole precedenti e quindi si avrà una probabilità $P(W) = P(w_1, w_2, w_3, w_4, w_5 \dots w_n)$, per N-grams successive si va a calcolare la probabilità di una parola in una sequenza di parola sulla base delle parole che precedono la stessa.

Quindi se applicassimo un 3-gram language model sulla frase “Ho fame ma Massimo o Martina sono sempre in ___” andremmo a prendere in considerazione solo le ultime 2 parole, ovvero “sempre in” e applicando la formula precedente $P(B|A) = P(A,B)/P(A)$ si ha che:

$$P(w|\text{sempre in}) = \text{occorrenze totali di (sempre in w)} / \text{occorrenze totali di (sempre in)}$$

Il Language Modeling può essere utilizzato per le operazioni di Information Retrieval in tre differenti modalità:

1. Probabilità di generare la query testuale a partire da un document language model
2. Probabilità di generare il documento testuale a partire da una query language model
3. Confrontare i language model rappresentanti la query e i document topics

L’approccio basico di language modeling per l’information retrieval consiste nell’individuare quale LM abbia maggiore probabilità di generare una query q. (NON SI CAPISCE BENE STO PEZZO).

Stimare le probabilità con le formule appena presentate potrebbe creare dei problemi, in quanto se dovessero mancare delle query word all’interno del documento lo score assegnato sarà pari a 0. Le parole mancanti, tuttavia, non dovrebbero avere 0 di probabilità di occorrenza e qui entra in gioco la tecnica di smoothing. Questa tecnica consente di stimare le probabilità di parole mancanti andando a ridurre la probabilità stimata per le parole individuate nel testo del documento e/o andando ad assegnare una probabilità di “left-over” alle stime delle parole che non sono state individuate nel testo.

VALUTAZIONE DEL MODELLO

Per valutare un language model basta semplicemente valutare se il modello preferisce frasi reali, frequentemente usate e grammaticalmente corrette rispetto a delle frasi sgrammaticate e poco usate. Applicato al modello sappiamo che i parametri del modello vengono allenati con un training set e che le performance del modello vengono valutate andando ad applicare il modello ad un test set e valutandone i risultati ottenuti.

Una valutazione per i modelli N-gram è la valutazione estrinseca. Questa valutazione consiste nel comparare due modelli A e B; i modelli vengono utilizzati per dei task come correggere gli errori di battitura ad esempio, successivamente si misurano le performance di entrambi i modelli sul task individuando ad esempio quante parole con errori di battitura sono state corrette e, infine, si comparano le accuracy dei modelli A e B. La valutazione estrinseca può essere molto dispendiosa a livello di tempo, perciò spesso si fa uso di una valutazione intrinseca che fa uso della perplessità; essa è la probabilità inversa del test set normalizzata dal numero di parole. Massimizzare la perplessità equivale a massimizzare la probabilità; conseguentemente tanto più basso è il valore della perplessità tanto più efficace sarà il modello.

7- WORD EMBEDDING

Il **Word Embedding** indica un set di tecniche di Natural Language Processing (NLP) in cui le frasi o le parole del vocabolario sono mappate in dei vettori di numeri reali. Concettualmente si ha un embedding matematico che passa da uno spazio con molte dimensioni per parola ad un vettore spaziale di dimensioni molto inferiori. I modelli in grado di generare questo tipo di mapping sono: **Count-based models** e i **Predictive models**.

TEXT REPRESENTATION

Ogni documento può essere rappresentato da un vettore di parole in diverse variazioni, infatti un documento può essere rappresentato come:

- **Rappresentazione binaria**: il vettore che identifica il documento sarà composto da 0 e 1. Vediamo quindi la rappresentazione del documento come una tabella che ha come colonna il documento e come righe tutte le parole presenti in tutti i documenti analizzati. Il vettore quindi avrà 0 se tale parola non è presente nel documento, viceversa avrà 1 se essa è presente.

$ D = N$			
	d_1	d_2	d_3
bear	1	0	0
cat	0	1	0
frog	0	0	1

$d_1 = [1, 0, 0]$ $d_2 = [0, 1, 0]$
 $d_3 = [0, 0, 1]$

- **Rappresentazione di frequenza**: in maniera analoga alla rappresentazione precedente avremo per ogni documento il numero di occorrenze di una determinata parola all'interno del loro corpus.
- **Rappresentazione pesata**: rappresentazione analoga alle precedenti che va a calcolare per ogni documento la frequenza pesata dei termini applicando TF-IDF.

Analizzando queste semplici rappresentazioni si possono confrontare vari documenti e individuare delle similarità come ad esempio l'occorrenza di determinate parole più frequente all'interno di documenti che condividono lo stesso topic. Proprio per questo il vettore che rappresenta i documenti è alla base dell'Information Retrieval.

Anche le parole possono essere rappresentate con dei vettori. Pensiamo ad una rappresentazione di frequenza presentata poco fa; se anziché utilizzare i valori delle colonne (che identificano i documenti) utilizzassimo i valori delle righe, avremmo una rappresentazione delle frequenze di una determinata parola all'interno di vari documenti. Di solito però la somiglianza di parole non è computata utilizzando una rappresentazione termine-documento.

Due parole sono simili se i loro vettori di contesto sono simili. La matrice utilizzata per questa rappresentazione ha parole sia sulle righe che sulle colonne. Tra le varie rappresentazioni delle parole possibili abbiamo:

- **Rappresentazione Locale**: ogni parola è rappresentata da un vettore chiamato vettore 1-hot. I vettori di questo tipo ci danno molte poche informazioni e la matrice generata può portare a dei problemi. Il numero di dimensioni (colonne) aumenta in maniera lineare all'aumentare delle parole che inseriamo nel vocabolario dato che per ogni parola andiamo ad identificare un vettore. La matrice ottenuta quindi sarà molto sparsa e composta principalmente da 0, inoltre non ci sono delle informazioni condivise tra le parole così rappresentate e quindi non è possibile individuare somiglianze tra le parole.

- **Rappresentazione distribuita**: per ogni parola del vocabolario sono associate k dimensioni di contesto che rappresentano le proprietà associate alle parole del vocabolario. Detto in altre parole si avrà come righe le parole del vocabolario e come colonne delle parole che definiscono delle caratteristiche che queste parole possono avere; il valore della cella quindi conterrà il grado di affinità tra la parola del vocabolario e la caratteristica.

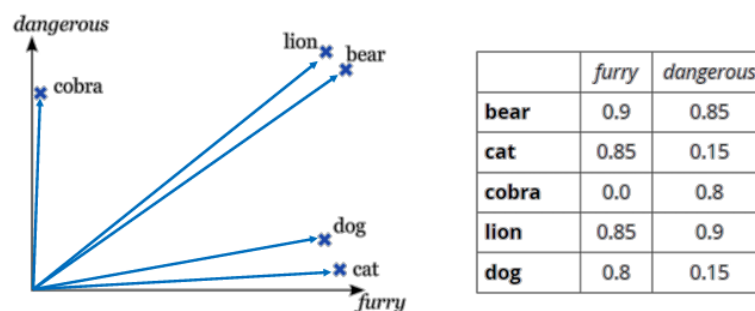
$C \quad (|C| = k \ll |V|)$

	<i>furry</i>	<i>dangerous</i>	<i>mammal</i>
<i>bear</i>	0.9	0.85	1
<i>cat</i>	0.85	0.15	1
<i>frog</i>	0	0.05	0

V

bear = [0.9, 0.85, 1.0] cat = [0.85, 0.15, 1.0]

Questo tipo di rappresentazione distribuita consente ai **vettori simili di essere raggruppati in base al contesto**.



Questa rappresentazione mette in risalto le caratteristiche comuni alle parole e quindi le loro somiglianze; **tanto più i vettori sono vicini tanto più simili sono le parole tra loro**. A differenza della rappresentazione locale, **la matrice risultante da questa rappresentazione è molto meno sparsa** risultando quindi decisamente più utile e facile da consultare.

Riassumendo quindi si ha che nella **rappresentazione locale** (one-hot) ogni termine del vocabolario è rappresentato da un vettore binario della lunghezza del vocabolario stesso dove una sola posizione possiede il valore 1 e tutte le altre invece 0. Nella **rappresentazione distribuita** invece ogni termine del vocabolario è rappresentato da un vettore di valori reali (quindi non solo 0 e 1).

Comporre dei vettori k-dimensionali che catturano il significato di un documento, dove le parole simili hanno simili vettori, è un compito difficile. **Assegnare manualmente questi vettori sarebbe impossibile**, perciò sono stati **implementati vari algoritmi** in grado di ricevere in input grandi corpus di testi e **creare dei modelli significativi**. Questi algoritmi si basano sulla **ipotesi distribuzionale** che dice che **parole simili tra loro occorrono in contesti simili**, quindi è **necessario conoscere il contesto** (ovvero le co-occorrenze) che sta attorno ad una parola quando vogliamo verificare se due parole sono simili.

CONTARE LE CO-OCCORRENZE DI PAROLE

Tra i vari metodi per contare le co-occorrenze di parole all'interno di documenti abbiamo il **Window-based Co-occurrence Matrix**. In questo metodo, **dato un corpus** di testo, si va a **conteggiare il numero di volte in cui la parola co-occorre all'interno di una finestra** di dimensioni particolari **oppure il numero di volte in cui la parola co-occorre con la parola di interesse**. La matrice risultante da questo processo viene definita window-based e possono essere di diverso tipo:

- Word-Word co-occurrence Matrix
- Term-context Matrix
- Count Matrix

Se volessimo creare un vettore per una parola con una Word-Word co-occurrence Matrix, dovremmo contare quante volte una parola co-occorre assieme ad altre specifiche parole:

Prima di tutto prendiamo i nostri corpus e identifichiamo le parole target, ovvero quelle da rappresentare.



Successivamente selezioniamo una finestra di dimensione 2 intorno alle parole target per definirne il contesto



Infine creiamo una Window-based Co-occurrence Matrix

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

Utilizzare questo approccio che si basa sulla frequenza è sicuramente utile ma ha notevoli difficoltà con termini molto spesso usati in tutti i possibili contesti come gli articoli ad esempio. Per far fronte a questo problema bisogna strutturare un modello di pesatura delle parole del contesto come ad esempio utilizzando il TF-IDF o il Pointwise Mutual Information (PMI).

La PMI va a vedere se due eventi (o parole nel nostro caso) x e y co-occorrono di più insieme rispetto alle occorrenze che avrebbero considerati indipendentemente l'uno dall'altro:

$$PMI(x, y) = \log_2 \left(\frac{P(x, y)}{P(x)P(y)} \right)$$

Il valore del PMI va da $-\infty$ a $+\infty$ tuttavia i risultati negativi sono problematici in quanto identificano la presenza di parole che co-occorrono meno rispetto a quanto occorrerebbero considerate indipendentemente. Per far fronte a questo problema si rimpiazzano i valori di PMI negativi con 0 ottenendo quindi la seguente formula:

$$PPMI(w_1, w_2) = \max \left(\log_2 \left(\frac{P(w_1, w_2)}{P(w_1)P(w_2)} \right), 0 \right)$$

Facendo un esempio pratico di calcolo della PMI si ha che data una matrice avente come righe le parole e come colonne le parole di contesto:

	Count(w,context)				
	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

Si avrà che la PMI di information con la parola di contesto data sarà calcolata con:

$$P(w=\text{information}, c=\text{data}) = 6 / 19 = 0.32$$

Inoltre si ha anche che:

$$P(w=\text{information}) = 11 / 19 = 0.58 \quad P(c=\text{data}) = 7 / 19 = 0.37$$

Una volta trovati tutti questi valori possiamo finalmente **calcolare la PPMI**:

$$\begin{aligned}
 \bullet \text{ PPMI}(\text{information}, \text{data}) &= \max \left(\log_2 \left(\frac{P(\text{information}, \text{data})}{P(\text{information})P(\text{data})} \right), 0 \right) \\
 &= \max \left(\log_2 \left(\frac{0.32}{0.58 \cdot 0.37} \right), 0 \right) = \boxed{0.57}
 \end{aligned}$$

Il PPMI è notevolmente influenzato dalla presenza di eventi poco frequenti, infatti le parole rare hanno un alto valore di PMI. Per far fronte a questo problema o si attribuisce alle parole rare una maggiore probabilità oppure si utilizzano degli strumenti di add-k smoothing che vanno ad aggiungere un valore di k ad ogni frequenza nella matrice term-context (ad esempio quindi prendiamo la matrice sopra presentata e aggiungiamo 2 a tutti i valori contenuti in essa).

Una matrice di co-occorrenza nella realtà è costituita da un grande numero di parole da cui ne conseguono dei vettori prodotti da PPMI che sono lunghi e sparsi (molti elementi sono uguali a 0). Esistono tuttavia delle tecniche per estrarre dei vettori di dimensioni inferiori per le parole che sono corti e densi (molti elementi non sono 0) e questo tipo di vettori vengono definiti embeddings.

WORD EMBEDDINGS

Questi embeddings si utilizzano due tipi di modelli:

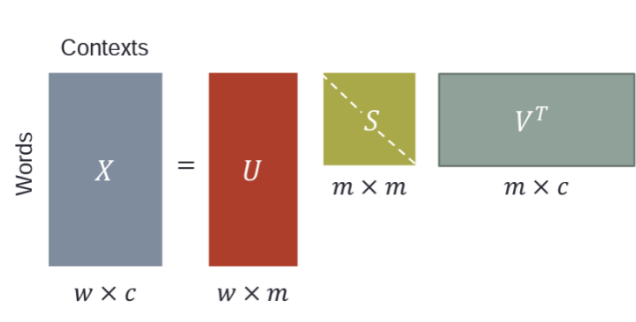
1. **Count-based models**: elabora la statistica di quanto spesso una parola co-occorre con le sue parole vicine all'interno di un grande corpus di testo, successivamente mappa queste statistiche in un vettore piccolo e denso per ogni parola. Questo vettore viene ottenuto tramite una riduzione della dimensionalità della term-context matrix; questo processo avviene tramite la fattorizzazione della matrice che porta ad ottenere una matrice di dimensione inferiore composta da parole e features. Una delle tecniche più famose utilizzata per questo compito è la Singular Value Decomposition (SVD) che fa uso della Latent Semantic Analysis (LSA).
2. **Predictive models**: questi modelli cercano di prevedere direttamente una parola basandosi sui suoi vicini. Questi modelli si ispirano alle reti neurali ed uno tra i vari che vengono utilizzati per questo scopo è il word2vec.

COUNT-BASED MODELS – SINGULAR VALUE DECOMPOSITION

Il modello afferma che ogni matrice X rettangolare $w \times c$ può essere espressa come un prodotto di 3 matrici:

1. U : una matrice $w \times m$ dove le righe w corrispondono alle righe della matrice originale X e le colonne rappresentano una dimensione (feature) in un nuovo spazio latente.

2. S : una matrice diagonale $m \times m$ di valori singoli che esprimono l'importanza di ogni dimensione (feature).
3. V^T : matrice trasposta $m \times c$ dove le colonne c corrispondono alle colonne della matrice originale X mentre le righe m corrispondono ai singoli valori.



Se anziché mantenere tutte le dimensioni m , decidessimo di mantenere solo i top- k valori singoli, allora potremmo ottenere una approssimazione di grado basso della matrice originale X ; questo processo si chiama **Latent Semantic Analysis (LSA)**. In sostanza quindi andiamo a sostituire k che è un parametro che scegliamo noi alle m per ridurre la dimensione di tutte le matrici.

Questo approccio ha diversi difetti tra cui:

- La dimensione della matrice cambia molto spesso dato che spesso si aggiungono nuove parole e il corpus conseguentemente cambia di dimensioni
- La matrice è estremamente sparsa dato che la maggior parte delle parole non co-occorrono
- Il costo per performare la SVD è quadratico

Per far fronte ad alcuni di questi problemi possiamo andare ad ignorare alcune parole come "the", "he", ecc... ovvero le **stopwords** e possiamo anche andare a pesare la co-occorrenza basandoci sulla distanza tra le parole all'interno del documento.

COUNT-BASED MODELS – GLOVE

Un altro modello count-based molto utilizzato è il **modello GloVe** (Global Vectors). Questo modello prende dalle informazioni statistiche solo gli elementi non pari a 0 e li inserisce in una **word-word co-occurrence matrix**. Lo sviluppo generale di tale modello si configura nella seguente formula:

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

Dove w sono i vettori di parole e w con l'ondina sono vettori di parole di contesto. La frazione rappresenta il rapporto tra le probabilità di co-occorrenza di due parole (i e j) con le context words (k).

Questa funzione viene utilizzata dal modello per apprendere la rappresentazione di parole in vettori.

Successivamente **GloVe** costruisce una funzione obiettivo J che associa i vettori di parole alle statistiche del testo. Il risultato di questa funzione ci fa ottenere U e V che contengono tutti i vettori di parole e tutti i vettori di parole di contesto. Questi due valori se sommati ci aiutano a rappresentare la co-occorrenza in un unico vettore $X = U+V$.

PREDICTIVE MODELS – WORD2VEC

La tecnica word2vec mette a disposizione un tool per creare delle collezioni di concetti simili in maniera automatica, basandosi su testi grezzi e senza la necessità di particolari abilità linguistiche da parte dell'utente. I testi grezzi sono utilizzati come **training set** per un apprendimento supervisionato; al crescere del training set e al crescere della varietà di parole presenti nei testi cresceranno anche le performance del

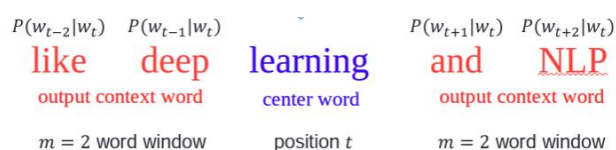
modello. Questo approccio è molto simile al language modeling ma va a prevedere il contesto attorno ad una parola anziché la parola successiva.

Ci sono due architetture utilizzate da word2vec:

- **Skip-gram model:** questa architettura va a prevedere le parole di contesto (quelle attorno) basandosi sulla parola attuale (la parola al centro)
- **Continuous Bag-of-Words (CBOW):** questa architettura va a prevedere la parola attuale basandosi sulle parole di contesto.

Anche le tecniche di training del modello sono generalmente due ovvero la Hierarchical Softmax e la Negative Sampling.

Approfondendo lo Skip-gram model possiamo dire che tale modello definisce una finestra che si muove assieme allo scorrimento delle parole della frase. La parola al centro è il target, mentre le parole attorno al target e all'interno della finestra sono le parole di contesto.



Il modello skip-gram viene allenato a prevedere le probabilità che una parola appartenga alle context word per un dato target, quindi prevedere la probabilità che data una parola target attorno ad essa ci siano parole di contesto. Il modello si basa su una rete neurale a strato nascosto che riceve in input un vettore x e produce in output un vettore y . Entrambi i vettori sono delle rappresentazioni di parole sotto forma di vettore one-hot, mentre lo strato nascosto è l'embedding delle parole.

In maniera simile si ha anche il funzionamento del modello CBOW che si basa anch'esso su una rete neurale. Tra i due modelli in generale skip-gram riesce a funzionare bene anche con dei training data ridotti riuscendo a rappresentare bene anche le parole rare; il modello CBOW invece in generale è molto più veloce da allenare rispetto allo skip-gram ed ha una accuracy migliore sulle parole più frequenti.

Più recentemente le tecniche di embedding si sono evolute in delle tecniche che tengono traccia del cambiamento del significato delle parole sia in base al contesto nel quale vengono usate sia in base al tempo; ad esempio la parola "gay" ha cambiato significato nel tempo passando dall'essere un termine che definisce allegria ad un termine che definisce l'omosessualità.

8- INTRODUCTION TO INFORMATION RETRIEVAL

La necessità dell'information retrieval è nata come conseguenza alla crescita sproporzionata di informazioni disponibili con l'avvento dell'internet e la conseguente necessità di avere dei sistemi in grado di reperire informazioni da questa grande mole di dati in maniera automatica. Si può dire che la IR sia la struttura che sta alla base di tutti i search engine del web, tuttavia il problema principale da affrontare da parte di questi sistemi è quello di **aiutare gli utenti a identificare informazioni rilevanti sulla base delle loro preferenze**. Per raggiungere questo scopo è necessario **interpretare correttamente il contenuto dell'informazione e interpretare le necessità dell'utente**.

I diversi **tipi di sistemi** utilizzati per accedere all'informazione sono:

- **Information Retrieval Systems (Search Engines)**: sistemi che richiedono l'utilizzo di query
- **DBMS**: sistemi che richiedono l'utilizzo di query
- **Information Filtering Systems (Recommender Systems)**: sistemi che non richiedono l'uso di query, ma che **richiedono dei profili utente**, ovvero delle informazioni riguardanti le necessità dell'utente che vengono **aggiornate dinamicamente sulla base del comportamento** dell'utente stesso

Le due principali tecnologie per implementare tali soluzioni sono:

- **Pull technology**: **l'utente richiede esplicitamente l'accesso** a dell'informazione come ad esempio utilizzando i motori di ricerca web
- **Push technology**: **l'utente viene automaticamente aggiornato** ricevendo dell'informazione di possibile interesse, come ad esempio l'utilizzo di sistemi di raccomandazione

INFORMATION RETRIEVAL SYSTEMS VS DBMS

Shannon da diverse possibili definizioni di informazione che sono:

- 1- Acquisizione di contenuto trasferito da un soggetto ad un altro
- 2- Informare, dare forma a qualcosa, eliminare incertezza
- 3- Un set di dati e la loro interpretazione

In sostanza si può dire che i dati sono fatti elementari che devono essere interpretati affinché possano diventare informazione.

I sistemi DBMS vengono utilizzati per gestire grandi quantitativi di dati riguardanti applicativi di business, mentre i sistemi di Information retrieval vennero creati per gestire un grande quantitativo di testo. Le **principali differenze** tra loro sono

DBMS:

- **Composti da dati e schemi concettuali** che ne consentono l'interpretazione
- Il reperimento dei dati viene compilato con una **serie di rigide selezioni con condizioni chiamate query**
- Possiede una **semantica ben definita** sia per i dati che per le condizioni

Information retrieval systems:

- Individua informazioni riguardanti un **particolare topic**
- Elabora documenti **il cui contenuto deve essere interpretato**
- La **presenza di errori nei risultati è tollerata**

INFORMATION RETRIEVAL SYSTEMS MODELS AND STRUCTURE

Un sistema IR interpreta il contenuto dei documenti andando a **definire una rappresentazione formale**. Data una query, un IR system **genera un ranking dei documenti basata sulla rilevanza** dei documenti che viene stimata a partire dalle necessità dell'utente definite all'interno della query.

La struttura base di un IRS è composta da una parte off-line e una on-line. La prima contiene i **documenti**, la loro **indicizzazione** e la **rappresentazione formale** degli stessi. La seconda parte è quella a cui si interfaccia l'utente inserendo la **query di ricerca** che verrà utilizzata per **reperire i documenti ritenuti rilevanti**.

Affinché sia possibile descrivere in maniera formale il documento, la query di ricerca e il metodo per confrontare query e documento gli IRS **utilizzano un modello matematico**.

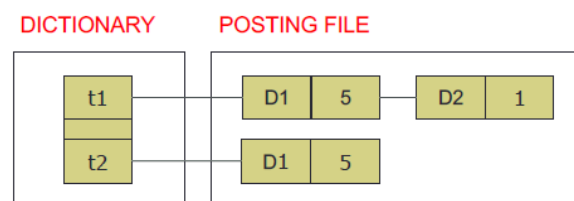
Il processo di **indicizzazione** si basa **sull'estrazione di features** che formano la base della rappresentazione del documento stesso. Nel caso di documenti testuali questi elementi chiamati indici generalmente sono delle parole.

9- DATA STRUCTURES FOR INDEXING

L'indicizzazione automatica di un documento testuale è il processo che ha come obiettivo quello di associare degli indici (parole) con un testo; gli indici associati con tutti i documenti di una collezione costituiscono il dizionario della collezione. L'utilizzo di indici consente di reperire in maniera veloce ed efficiente i documenti tramite l'utilizzo di parole chiave inserite all'interno di una query.

Per ottenere una rappresentazione della tabella degli indici (occorrenze delle parole) all'interno di una collezione di documenti conviene utilizzare una matrice term/document organizzata in una struttura dati dinamica. In questo modo gli indici vengono organizzati all'interno di un dizionario dove ogni termine punta ad una lista contenente i riferimenti ai documenti dentro i quali quel termine è un indice. Questa struttura si chiama **Inverted File** data structure.

Segue una rappresentazione dove abbiamo il dizionario coi vari termini t e i posting file contenenti la lista di documenti e il relativo numero di occorrenze del termine all'interno degli stessi.



Con questa modalità di rappresentazione degli indici si ha che:

- Il dizionario contiene per ogni indice: termine, frequenza globale all'interno del dizionario, pointer che indica alla posting list
- La posting list contiene per ogni elemento: identificatore univoco del documento, frequenza del termine all'interno del documento, posizione nel documento di ogni occorrenza del termine

DICTIONARY ORGANIZATION & POSTING FILE OPTIMIZATION

Dato che la memoria richiesta per creare una rappresentazione di questo tipo dipende dalla dimensione del dizionario e dalla occorrenza dei termini all'interno della collezione di documenti, risulta necessaria una ottimizzazione della struttura di rappresentazione. Il dizionario può essere organizzato come:

- Struttura lineare
- Struttura ad albero binaria
- Struttura B-tree

Nei posting file è possibile eseguire una compressione su:

- I termini nel dizionario
- Gli identificatori univoci dei documenti:
- I valori numerici che identificano l'occorrenza dei termini
- La posizione dei termini nel corpus del documento

Per quanto riguarda l'organizzazione del dizionario abbiamo come prima tipologia di rappresentazione la struttura lineare. Gli index term sono salvati all'interno del dizionario e vengono ordinati in ordine alfabetico. Questa struttura ha il vantaggio di avere accesso e ricerche veloci, utilizzare in maniera efficiente lo spazio in memoria, ma ogni qualvolta si dovesse inserire un nuovo termine all'interno del dizionario questo comporterebbe l'intera ricostruzione del dizionario.

La seconda rappresentazione possibile è la **binary tree**. Questa rappresentazione consiste nel generare un **albero che possiede due figli**. Le **ricerche partono dal nodo radice** e scendono fino alla foglia desiderata; ovviamente questa **soluzione rende veloci le ricerche**, tuttavia **bilanciare la struttura dati quando si aggiungono o rimuovono termini** risulta essere parecchio oneroso.

La B-tree structure è una **struttura ad albero bilanciata** dove ogni nodo ha un numero variabile di figli e un numero variabile di puntatori a dei sotto alberi. Il vantaggio di questa struttura è che **l'accesso avviene velocemente**, è possibile **inserire velocemente nuovi termini** e **utilizza in maniera efficiente la memoria disponibile**, tuttavia per le **ricerche sequenziali** questa struttura fatica.

Per ottimizzare il **posting file** ci sono diverse tecniche possibili tra cui la **Block Division**. Questa tecnica consiste nel **dividere il testo contenuto in ogni documento all'interno di blocchi** e le **occorrenze puntano ai blocchi all'interno dei quali è presente una determinata parola**. Il vantaggio che si ottiene da questa tecnica è ovviamente quello di **ridurre il numero di indirizzi a cui puntare** in quanto non si avrà più un indirizzo per singola parola ma uno per blocco di parole.

Block 1	Block 2	Block 3	Block 4
That house has a	garden. The garden has	many flowers. The flowers	are beautiful.

• Inverted index

Dictionary	DocID	Block number	Position in the text (Word number)
beautiful	D1	4	6
flowers	D1	3	4, 5
garden	D1	2	2, 3
house	D1	1	1

ALTERNATIVE ALLA INVERTED FILE DATA STRUCTURE

Vi sono principalmente due alternative alla inverted file structure appena descritta: la **suffix-tree/array structure** e il **signature structure (bit mask)**.

La suffix-tree/array structure viene usata per **indicizzare delle sequenze alfanumeriche**; di fatto queste strutture vedono il testo come se fosse una lunghissima stringa. **Ogni posizione (carattere) nel testo viene considerata come un suffisso** e ogni suffisso è **identificato dalla sua posizione all'interno della stringa**. Gli index points sono selezionati all'interno del testo come segue:

This is a text. A text has many words. Words are made from letters.

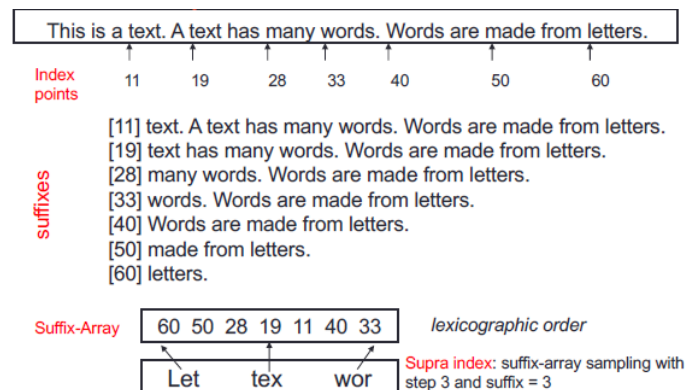
↑ ↑ ↑ ↑ ↑ ↑ ↑
 11 19 28 33 40 50 60

• Index points (points of identification of suffixes)

- [11] text. A text has many words. Words are made from letters.
- [19] text has many words. Words are made from letters.

I **puntatori dei suffissi sono salvati all'interno delle foglie dell'albero suffix-tree**. Questa struttura ha il vantaggio di essere **molto utile per ricerche complicate** come le ricerche sequenziali o di frasi intere; tuttavia il processo di costruzione di tale struttura risulta essere parecchio oneroso e con necessità di parecchio spazio in memoria.

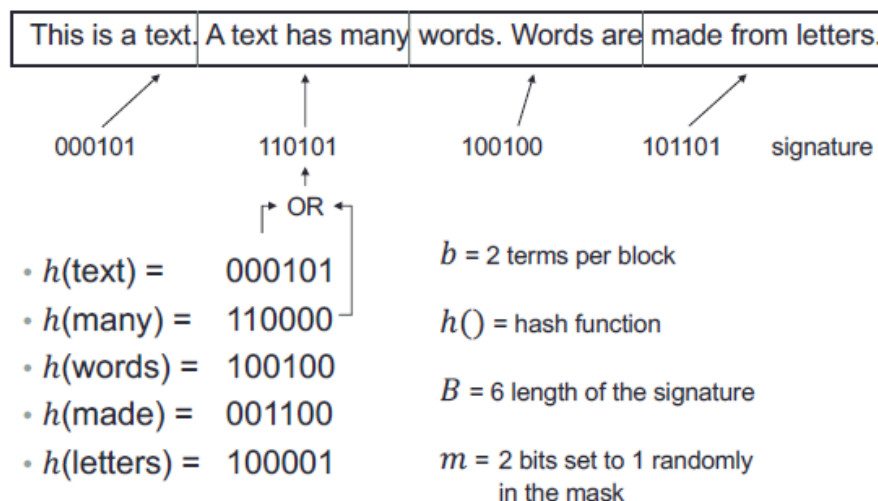
I **Suffix arrays** mettono a disposizione le stesse funzionalità delle suffix trees ma richiedendo molto meno spazio. Questa struttura è costituita da un semplice **array contenente tutti i puntatori ai suffissi del testo in ordine lessicografico**.



I vantaggi di questa struttura sono **l'ottimizzazione dello spazio su disco**, la facilità **nell'eseguire ricerche binarie** a causa dell'ordine lessicografico e, infine, **l'ottimizzazione del numero di accessi**.

La **signature structure** ha come idea di base quella di **dividere il documento in blocchi di uguale dimensione**, **assegnare ad ognuno dei blocchi una firma tramite funzione di hash** che verrà utilizzata per identificare il blocco durante la fase di ricerca con query. Sostanzialmente **si prendono dei blocchi di parole fissate**, e, a partire dal contenuto dei blocchi, **si rimuovono le stopwords** e successivamente si **codificano in hash binario le parole rimanenti**; la composizione di queste parole dà luce all'hash che descrive il blocco.

Signature structure (Bit Mask)



Allo stesso modo, facendo il procedimento inverso, è **possibile partire dall'hash di un blocco e ottenere l'hash delle parole che lo compongono**. Questo processo inverso tuttavia potrebbe ottenere un risultato di tipo **false drop**, ovvero che **il blocco soddisfi i criteri di ricerca di una query nonostante il termine ricercato non sia presente** all'interno del blocco stesso.

10- THE BOOLEAN MODEL AND THE VECTOR SPACE MODEL

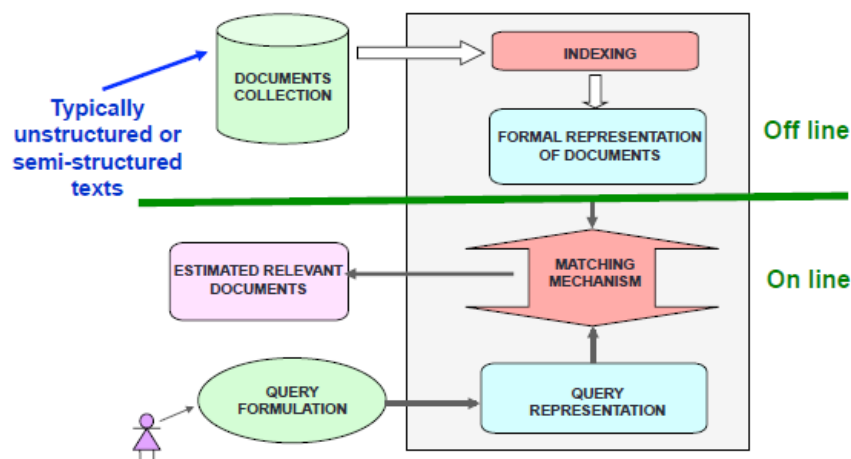
Un Information Retrieval System è **basato su un modello matematico** che può essere:

- **Modello classico**: booleano, VSM, probabilistico

- **Modello set based:** fuzzy model
- **Modello algebrico:** VSM generalizzato, latent semantic indexing
- **Modello probabilistico avanzato:** belief networks, language models

MODELLO BOOLEANO

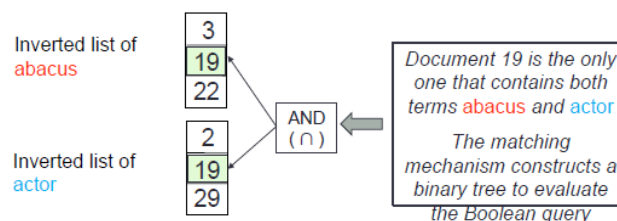
Il modello booleano è basato sulla **Set Theory**; un documento è formalmente rappresentato come un **set di termini** indicizzati a cui vengono associati dei **pesi binari**. Una **query** è formalmente rappresentata da una **espressione booleana** applicata ai termini. Il **matching mechanism** applica un **set di operazioni**. La **rilevanza** è modellata come **una proprietà binaria di documenti**. La descrizione appena fatta delle componenti di questo modello si basa sulla struttura tipica di un IRS che segue:



Le query booleane **utilizzano due o più termini** di ricerca **collegati tra di loro tramite connettori booleani** come AND, OR, ecc... Ovviamente i termini possono essere negati con l'operatore NOT. Nel pratico se nella query cerco *t1 AND t2* otterrò come **risultato i documenti all'interno dei quali entrambi i termini sono presenti**.

L'indicizzazione avviene tramite **inverted file**, quindi per ogni termine presente nel dizionario avremo associati **l'elenco (posting list) dei documenti all'interno dei quali sono presenti i termini**.

Quando il sistema riceve in ingresso una query, si esegue l'accesso al dizionario e si prende la posting list corrispondente al termine ricercato:



La **priorità** di valutazione di query complesse prevede che vengano **prima valutate tutte le AND e NOT** e successivamente le **OR**.

La **valutazione** delle query booleane avviene in **due modalità: full evaluation e lazy evaluation**.

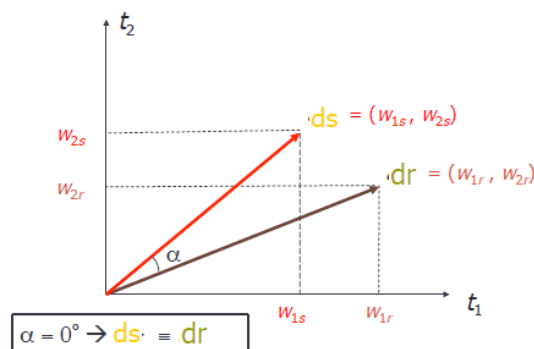
Per ottimizzare la valutazione delle query, all'interno del dizionario si potrebbe **memorizzare il totale dei documenti all'interno dei quali tale documento è presente**; in tal modo è possibile ottimizzare in maniera automatica le query andando ad **eseguire per prime le operazioni booleane tra i termini meno ricorrenti nei documenti**.

VECTOR SPACE MODEL (VSM)

Il modello VSM è basato sull'algebra lineare; sia i documenti che le query sono rappresentati in un vettore spaziale di N-dimensioni, dove N è il numero di termini indicizzati nella collezione considerata. I vettori che rappresentano i documenti contengono al loro interno dei pesi associati ad ogni termine presente all'interno del documento.

La rilevanza è modellata come una nozione graduale ed è proporzionale alla prossimità del vettore che identifica il documento con il vettore che identifica la query. I termini che sono presenti contemporaneamente in un documento non sono considerati correlati tra loro.

Si avrà la seguente rappresentazione:



La similarità tra i due vettori equivale a: $\cos \alpha = (x \cdot y) / |x| |y|$

Il VSM grazie all'utilizzo dei pesi migliora la qualità dei risultati e, grazie all'utilizzo della similarità, consente di avere un modello in grado di ottenere un documento come risultato anche se tale documento non soddisfa in toto la query e conseguentemente consente di creare un ranking dei documenti ottenuti come risultato in tal modo.

11- LANGUAGE MODELS AS IR MODELS

I language models analizzano il corpus dei documenti testuali per utilizzarli come dati per creare un modello predittivo in grado di determinare la probabilità di una data sequenza di parole che occorrono in una frase. Tra i vari language model abbiamo:

- Unigram language model: identifica la probabilità del verificarsi di un termine andando ed estrarre parole da un bucket sulla base della probabilità di distribuzione.
- N-gram language model: servono ad identificare la probabilità del verificarsi di una parola sulla base di parole precedenti (bigram usa la parola precedente, trigram usa le due parole precedenti)

Un topic all'interno di un documento o query può essere rappresentato come un language model; ad esempio le parole che tendono ad occorrere spesso quando si discute su un topic avranno alte probabilità nel corrispondente language model.

Ci sono 3 possibili applicazioni dei language models per l'IR:

- Probabilità di generare il testo della query da un document language model
- Probabilità di generare il testo del documento da una query language model
- Confrontare i language model rappresentanti le query e i topics dei documenti

Ci sono anche diversi modelli per calcolare la rilevanza dei topic, il primo è il query likelihood model. Questo modello classifica i documenti in base alla probabilità che la query ha di essere generata da un document model. La formula di tale probabilità è: $P(D|Q)=P(Q|D)P(D)$ quindi in sostanza si calcola la probabilità di generare una query Q a partire da un documento D.

Un'altra tecnica è quella dello smoothing. Questa tecnica viene utilizzata per stimare le probabilità delle parole mancanti; per fare ciò riduce le probabilità per le parole viste all'interno del documento e assegna questo scarto di probabilità alle parole che non sono presenti nel testo del documento. Il calcolo della probabilità delle parole non viste è:

$$\alpha_D P(q_i|C)$$

Dove $P(q_i|C)$ è la probabilità di una query word i all'interno del collection language model per la collezione C, e α_D è un parametro. Il valore di questo parametro in diversi tipi di tecnica di smoothing varia; nel caso della Jelinek-Marczer Smoothing α è una costante, mentre nel caso della Dirichlet Smoothing α dipende dalla lunghezza del documento.

RELEVANCE MODEL

Oltre ai language model esistono anche i relevance models; questi modelli, che ricevono in input i documenti rilevanti e le query, servono per determinare la rilevanza di un documento rispetto ad una query. La probabilità $P(D|R)$ è la probabilità di avere un documento rilevante rispetto ad una determinata query.

Un esempio di relevance model è il Pseudo-Relevance Feedback. Questo modello va a stimare la rilevanza utilizzando la query in ingresso e un insieme di top-ranked documents; questa classifica di documenti viene stillata sulla base della loro somiglianza al document model di partenza. La misura che viene utilizzata per confrontare i language models è il KL-divergence che va a misurare la differenza tra le due distribuzioni di probabilità dei language model. Nei language model la KL-divergence viene utilizzata per stimare un ranking dei documenti in base alla somiglianza tra la loro probabilità distribuita e la vera distribuzione dei dati.

12- KNOWLEDGE GRAPHS (NO LEZIONE)

Un Knowledge Graph descrive degli oggetti di interesse e le loro interconnessioni. I dati sono organizzati in nodi e archi. La differenza tra un DB e un KG è che se il primo immagazzina dati per uno specifico scopo applicativo, basandosi su un sistema di tabelle, colonne e attributi che utilizzano una semantica complessa compresa solo da chi ha strutturato lo stesso DB, il secondo, invece, immagazzina dei dati riguardanti un argomento senza uno scopo applicativo specifico, utilizza una semantica facilmente comprensibile da tutti e una struttura dati indipendente da applicativi.

I KG possono essere utilizzati per vari scopi nell'information retrieval tra cui:

- Document retrieval: inserire i documenti in una classifica partendo da una query
- Entity retrieval: inserire entità in una classifica a seconda della loro rilevanza con una query

- **Raccomandazioni:** inserire entità in una classifica a seconda della loro correlazione con una query

13- CONTEXTUALIZED EMBEDDINGS - NEURAL LANGUAGE MODEL

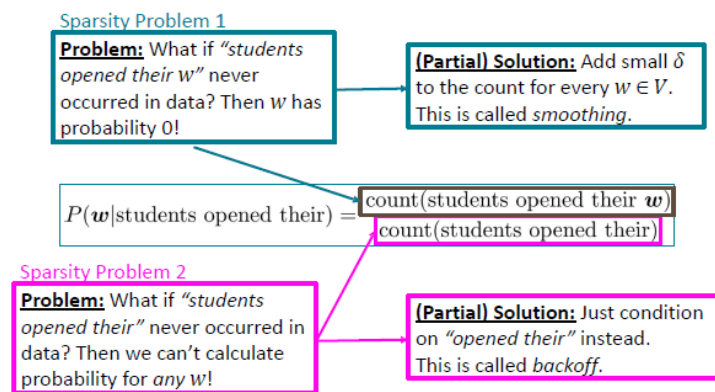
Il **word embedding** indica un set di tecniche di **NLP** dove **parole o frasi di un vocabolario sono mappate in vettori di numeri reali**. I modelli utilizzati per ottenere questo mapping sono: **Count based models e Predictive models**. I vettori che rappresentano le parole presenti all'interno di più documenti vengono uniti per creare una **matrice di co-occorrenza** che generalmente è molto **grande e sparsa**. Esistono diverse tecniche in grado di trasformare questi vettori in vettori appartenenti ad uno **spazio dimensionale inferiore**; queste tecniche sono gli **embeddings**.

Un **testo può essere rappresentato come un language model per rappresentare i suoi topics**; in questo modo le **parole che tendono ad occorrere spesso quando si parla di un particolare topic** avranno alte probabilità di essere presenti in un language model (es. il suggeritore di parole quando scrivi i messaggi).

Le parole sono **semanticamente dinamiche**, ovvero **cambiano il loro significato in base al contesto** nel quale compaiono. **Una parola in base al contesto può assumere diversi significati**, ad esempio la parola pesca può riferirsi al frutto o al pescare. La **rappresentazione vettoriale delle parole non coglie la polisemia** delle parole, ovvero parole uguali ma con significato diverso.

Per poter eseguire un **embedding senza perdere il contesto** nel quale le parole vengono inserite bisogna utilizzare i **type embedding**. Più nello specifico, come visto nei **Language Models**, è possibile utilizzare la tecnica degli **n-gram** che va a **prendere in considerazione una o più parole consecutive** per determinarne il significato. In questo modo andando a confrontare **diverse applicazioni di n-gram** (unigram, bigram, trigram) allo stesso testo, possiamo **identificare come il significato di una parola vari** a seconda delle parole successive prese in considerazione. Il **principale problema** di questa tecnica si può individuare nella **sparsità delle parole nel testo** e dalla **difficoltà nel memorizzare tutti gli n-gram** osservati per produrre la previsione.

Vediamo ora come un modello N-gram viene applicato per calcolare la probabilità di una parola w .

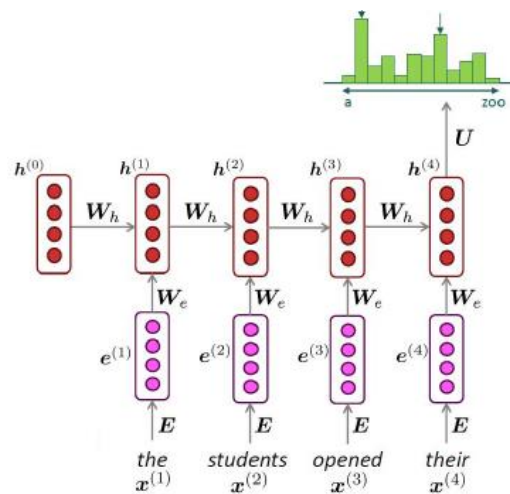


RETI NEURALI

Una soluzione più avanzata consiste **nell'utilizzo delle reti neurali** che consentono di produrre **rappresentazioni di parole che considerano il contesto senza avere un vettore di parole di grandezza prefissata**. Eventualmente è **possibile creare una finestra di dimensione fissata** da analizzare con la rete neurale. Rispetto alla soluzione dei LM, l'utilizzo della rete neurale consente di risolvere i problemi di sparsità e di memorizzazione, tuttavia **utilizzare una finestra di dimensione prefissata limita notevolmente la predizione del modello** che dovrebbe prendere in considerazione un input di qualunque dimensione.

Per ovviare a questo problema la soluzione più adoperata nei language model basati su reti neurali sono le **RNN**, ovvero le **Recurrente Neural Network**: queste reti neurali sono composte da **diversi strati nascosti** che

consentono quindi di **processare un input di qualunque dimensione**. In particolare le **RNN utilizzano le informazioni ricavate dalle computazioni eseguite da uno step precedente per computare lo step successivo**, riuscendo così a rimuovere il limite della finestra di dimensione prefissata di termini imposta dai LM.



ELMO MODEL E BERT MODEL

Tra i vari modelli sviluppati per l'embedding abbiamo il modello **ELMO (Embedding from Language Models)**. I due aspetti chiave di questo modello sono:

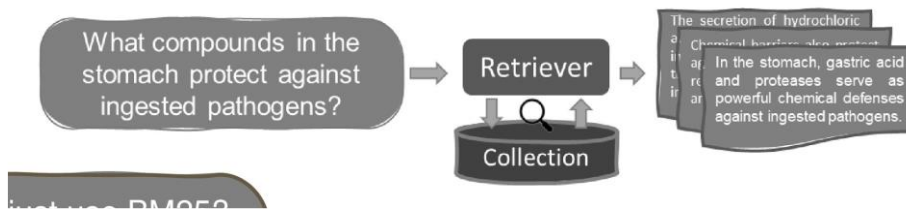
- 1- L'embedding di un word type deve dipendere dal suo contesto, ma la dimensione di questo contesto non deve essere prefissata. Utilizzeremo quindi una RNN
- 2- I language models codificano in contemporanea il significato delle parole basato sul loro contesto

Il modello esegue un **embed** delle word type **all'interno di un vettore** facendo uso di **step precedenti di embeddings computati** per poi successivamente usare un **language model bidirezionale applicato agli embeddings**. Con bidirezionale si intende un language model che valuta sia le parole precedenti alla parola presa in considerazione, sia le parole successive alla stessa. In questo modo ELMO consente di generare una rappresentazione delle parole presenti sia nei documenti che nelle query **più coerente con il contesto nel quale sono inserite**. Per valutare le prestazioni del modello lo si deve applicare: dopo aver scelto un task di NLP che prevede l'utilizzo di un modello a rete neurale, si applica il modello ELMO per rimpiazzare gli embeddings delle parole, successivamente si esegue un training del modello con i nuovi embeddings e, infine, **si valutano e confrontano le prestazioni dei due modelli**.

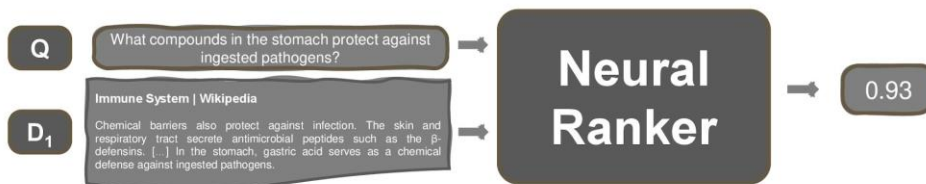
Un altro modello utilizzato è **BERT (Bidirectional Encoder Representation from Transformers)**. Questo modello esegue un **pre-training di modelli bidirezionali di encoding basati su transformers** per comprendere al meglio il linguaggio utilizzato. Con **bidirezionale** si intende che il modello **utilizza il contesto presente sia alla sinistra della parola presa in considerazione sia alla sua destra**. Questo modello fa uso di una architettura basata su **transformers**, ovvero una **rete neurale in grado di pesare l'importanza di parole diverse all'interno di una frase**.

14- NEURAL INFORMATION RETRIEVAL (RAGANATO)

Nell'information retrieval le reti neurali sono fondamentali perché risolvono il problema del **ranked retrieval** che riceve in input un corpus molto grande e una query testuale, andando a **restituire i top K documenti per rilevanza**.



Questo modello porta con sé diversi problemi che vengono risolti dalle **reti neurali che assegnano uno score ad ogni coppia di documento-query**; questo tipo di rete neurale viene definito **neural ranking**. La valutazione della coppia query-documento in molti casi risulta computazionalmente troppo onerosa in quanto spesso le collezioni contengono al loro interno milioni di documenti.



Per far fronte a questo problema è stato introdotto il **neural re-ranking che consiste nel creare un tetto massimo di documenti da prendere in considerazione per la valutazione**.

Un modello successivamente creato che riscontra risultati migliori è il **Query-Documents Interaction Model**; questo modello esegue una **tokenizzazione di query e documento** per poi eseguire un embed di ogni token, successivamente si costruisce una **matrice di interazione document-query** che consente di **ridurre la stessa in uno score**.

L'ultimo modello presentato è il BERT che però presenta una importante **problematica: le computazioni sono ridondanti e onerose** dato che la **rappresentazione delle query viene ripetuta per ogni documento** nella collezione e viceversa la rappresentazione del documento viene fatta una volta per ogni query. Una possibile soluzione sarebbe quella di andare a pre-computare i documenti e le loro rappresentazioni per poterle successivamente utilizzare abbinate alle query.

REPRESENTATION SIMILARITY E LATE INTERACTION

Per ottenere queste pre-computazioni sono stati introdotti alcuni paradigmi di reti neurali: la **Representation Similarity** e la **Late Interaction**.

La representation similarity va a **tokenizzare query e documenti**, per poi **codificarli in maniera indipendente all'interno di un vettore singolo** in modo tale da poter **stimarne la rilevanza**. In questo modo il paradigma consente di **pre-computare i documenti**, **ammortizzare il costo computazionale delle query** e **abbassare notevolmente il costo computazionale del calcolo della similarità**.

Per quanto riguarda le **late interaction**, questa tecnica consente di **raffinare la query di ricerca basandosi sui risultati iniziali della prima ricerca**. In questo modo si può andare a iterare la procedura di raffinamento di una stessa query piuttosto di andare ad utilizzarne una nuova.

WORD SEGMENTATION

I modelli neurali hanno lo svantaggio di **non essere in grado di elaborare bene le parole Out Of Vocabulary (OOV)** o in generale poco frequenti e di rappresentare in maniera poco precisa le stesse. Il problema di Open-Vocabulary è molto comune dato che la maggior parte dei corpus utilizzati contengono milioni di parole talvolta non presenti all'interno del vocabolario. **Una tecnica di risoluzione** di questo problema sono i **Back-off Models** che vanno a **rimpiazzare le parole rare con un valore N/A** che consentono al modello di **non considerare tali parole nella computazione**. Un'altra soluzione sono le **wishlit** che **consistono nel codificare tutte le parole tramite un vocabolario di piccole dimensioni andando a generalizzare le parole non conosciute**.

15- CONTEXTUAL SEARCH AND PERSONALIZATION (NO LEZIONE)

Gli utenti che necessitano di informazioni spesso hanno difficoltà nel **definire effettivamente quali siano le informazioni che necessitano**. Il meccanismo **query-centered** dei motori di ricerca, seppur semplificata accettando il linguaggio naturale, comporta la necessità da parte dell'utente di avere la capacità di esprimere in modo corretto di quali informazioni necessita. Un altro grande problema riguardante le query di ricerca degli utenti è quello di **identificare il contesto di riferimento**: se ad esempio un utente ricerca la parola pesca non si sa se si riferisce al frutto o al pescare.

La **relevance feedback** è stata introdotta per far fronte a questi problemi in quanto il meccanismo di ricerca viene migliorato andando a **monitorare il comportamento dell'utente nella fase di ricerca dei documenti**. Questa particolare attenzione rivolta al contesto ha portato la ricerca nei search engine da essere query-centered ad essere **context-aware**. Tra i vari fattori che, oltre alla query, influiscono sul contesto ci sono: **la posizione dell'utente, l'orario in cui si effettua la ricerca, il device con cui si esegue la ricerca** ecc...

RICERCA PERSONALIZZATA

La **ricerca personalizzata produce un risultato di ricerca sulla base del contesto dell'utente** che la esegue; il contesto dell'utente viene rappresentato con uno **user model**. Questo tipo di ricerca quindi non si limita ad utilizzare il contesto come prima menzionato, **ma modella lo user model sulla base delle preferenze di un utente**, la sua **cronologia di ricerca**, i siti web visitati, ecc... I due principali problemi a cui deve far fronte la ricerca personalizzata sono:

- 1- **Definire lo user model nella maniera più semplice e meno invasiva possibile** andando a mantenere inalterate le interazioni con il sistema da parte dell'utente e collezionando nel frattempo tutte le informazioni necessarie a definire lo user model
- 2- **Utilizzare lo user model per migliorare la qualità di ricerca** sulla base delle informazioni tratte dallo user model

Per sviluppare uno user model bisogna definire un contesto che può contenere **diverse informazioni** come:

- Dati **demografici**
- **Informazioni sul contenuto definite esplicitamente** dall'utente come parole chiave
- Informazioni generate dalle **attività social**
- **Cronologia di ricerche, pagine visitate, navigazione web**

Una volta trovate le informazioni per generare uno user profile è necessario **rappresentarle**, alcuni dei metodi di rappresentazione sono:

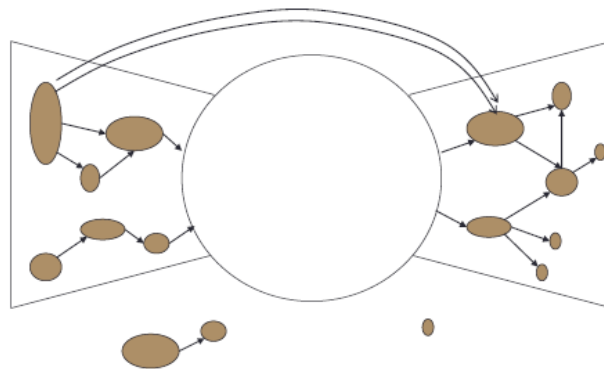
- **Bag of words**
- **Vettori**
- **Grafi**
- **Ontologie**
- **Language models, topic models, word embeddings**

Per **adoperare** in maniera utile i dati ottenuti dalla generazione dello user model è necessario effettuare alcuni accorgimenti che seguono i seguenti step:

- **Pre-processing approaches**: la query è modificata andando ad integrare o riscrivere termini utilizzando le informazioni dello user profile
- **Post processing approaches**: **re-ranking dei risultati**, quindi attribuire uno score per ogni documento risultato sulla base delle informazioni ottenuto dallo user profile e combinare questo nuovo score con il precedente
- **In-process**: personalizzazione integrata nel processo di ranking dei nuovi risultati ottenuti
- Presentazione personalizzata dei risultati

16- WEB SEARCH

Possiamo pesare al web come un grafo in cui i nodi sono gli URL e gli archi che collegano tra di loro i nodi sono i link ai siti web. Questo grafo prende il nome di **web graph** e ovviamente è un **grafo dinamico** in continua evoluzione e aggiornamento. Il web graph quindi è un **grafo diretto** dove sono anche presenti **relazioni di equivalenza** (archi che collegano due nodi in entrambi i sensi) che consentono di individuare delle **componenti fortemente connesse**. Il **grafo quindi può essere ridotto andando a rappresentare solo le sue componenti fortemente connesse tra loro** anziché i singoli nodi fortemente connessi.

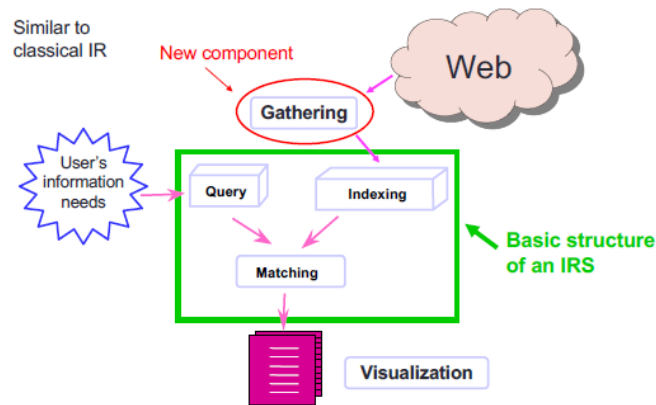


La rappresentazione appena mostrata individua **al centro una gigante componente fortemente connessa**, **sulla sinistra** abbiamo invece tutte quelle **pagine che puntano direttamente alla componente centrale ma che non sono raggiungibili da essa**, infine **sulla destra** abbiamo tutte le **pagine che sono raggiungibili dalla componente gigante ma che non possono raggiungere la stessa**. In realtà oltre a queste tre grandi sezioni abbiamo delle **componenti isolate** che non sono collegate a nessuna delle tre grandi sezioni che definiscono la struttura principale del web.

SEARCH ENGINE

I servizi che un utente richiede ad un search engine quando naviga il web sono in genere: **reperire informazioni**, **raggiungere una determinata pagina** (pagina di Ryanair), per **ottenere dei dati** (download foto, documenti, ecc..).

La **struttura dei search engine è molto simile a quella della classica IR**, ovvero: si parte dall'utente che necessita di una informazione ed interroga il search engine con una **query**, sulla base della query e dei documenti raccolti e **indicizzati** dal web avviene un **matching** che identifica i risultati più adeguati e, infine, tali dati vengono **visualizzati**.



Una pagina web in questo modo può essere vista come un documento in IR. Allo stesso modo l'insieme delle pagine web (ma solo quelle pubblicamente indicizzabili) costituiscono la collezione di documenti reperibili dal web. Le pagine web che non sono indicizzate dai search engine costituiscono il deep web; in generale si tratta di pagine ad accesso limitato, private o non linkate da altre pagine web.

I problemi che portano con sé le web search sono:

- **Scalabilità**: bisogna attingere ad un enorme quantitativo di informazioni
- **Volatilità**: le pagine web e i dati contenuti in esse cambiano velocemente
- **Ridondanza**: una grande quantità di dati è ripetuta in diverse pagine web
- **Qualità**: molti dati possono essere obsoleti o non attendibili
- **Eterogeneità**: diversi formati di dati come audio, video, testo
- **Query**: le query spesso sono imprecise, contenenti poche parole e vaghe

Andremo ora a vedere più nel dettaglio le varie fasi che strutturano il processo di web search:

- **Gathering**: reperimento delle pagine web
- **Indexing**: rappresentazione dei contenuti dei documenti selezionati
- **Ranking**: classificare i documenti ottenuti

DOCUMENT GATHERING

Ci sono due possibili alternative per questa prima fase:

- 1- Le pagine web sono inviate direttamente al search engine dagli stessi proprietari della pagina
- 2- Il search engine è dotato di un software agent che ricerca e invia pagine nuove o aggiornate ad un server che le indicizza

Un tipo di software agent disponibile è il crawler che naviga il web utilizzando degli URL conosciuti come punti di partenza per poi utilizzare i link in queste pagine per passare ad altre pagine. Il crawler quindi si occupa di collezionare le pagine visitando il web graph, ma questa navigazione porta con sé un grande problema, ovvero quello di scegliere da quale pagina web iniziare la ricerca.

La scelta del seed influenza il set di pagine che verranno visitate in quanto se iniziassimo la ricerca da una pagina presente all'interno della giant component del web avremmo accesso a molte pagine, viceversa se iniziassimo la ricerca da una pagina isolata non avremmo tale risultato. Allo stesso modo un processo difficile da definire è quello della scelta delle pagine da scaricare e di quale contenuto presente nelle stesse scaricare; in generale la selezione delle pagine viene fatta sul tipo del contenuto delle pagine visitate o su determinati criteri come la lingua utilizzata.

I web server, per evitare di essere sovraccaricati di richiesta utilizzano dei file **robots.txt** che indicano se i crawler possono indicizzare il contenuto del sito o eventualmente bloccare gli stessi nel caso di ripetute richieste di accesso ai dati.

Il processo di azione di un crawler è il seguente:

- 1- Inizializza una coda di pagine con degli URLs noti
- 2- Seleziona un indirizzo URL dalla coda
- 3- Seleziona la pagina
- 4- Cerca altri URLs presenti nella pagina selezionata
- 5- Scarta gli URLs che sono già stati visitati o che non possono essere analizzati (PDF, JPG, EXE)
- 6- Aggiunge gli URLs idonei alla coda tramite una strategia di breadth-first o depth-first
- 7- Se il tempo a disposizione non si è esaurito, itera il processo dal punto 2

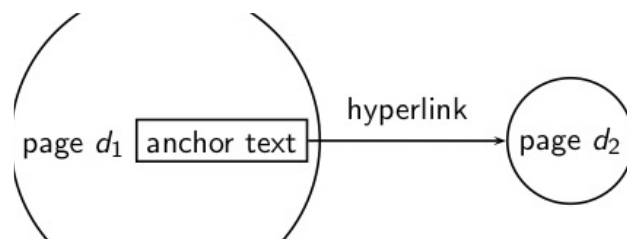
Le tecniche utilizzate per scegliere l'ordine col quale visitare gli URL sono generalmente la **breadth-first** o **depth-first**. La prima va ad analizzare per prime tutte le pagine di cui è presente un link nella pagina che si sta attualmente analizzando, mentre la seconda segue il primo link che trova all'interno della pagina attualmente analizzata e continua finché non arriva ad una pagina senza link. Le pagine analizzate tuttavia possono cambiare nel tempo e perciò il crawler visita più volte pagine che sono già state indicizzate per verificare che esistano ancora e mantiene in memoria l'ultima volta che ha indicizzato una pagina.

In genere i crawler vengono usati in maniera distribuita per aumentare la velocità di analisi. Ogni agente lavora su una macchina diversa e esegue i task del crawler prima menzionati; questi agenti sono tutti collegati tra loro e gestiti da un coordinatore, oppure, in assenza di un coordinatore, ogni agente procede indipendentemente. Questa seconda soluzione se da un lato può evitare un fenomeno di bottleneck che può verificarsi in presenza di un coordinatore, dall'altro rischia di generare fenomeni di overlap oppure, nel caso di malfunzionamento di un crawler, di perdere porzioni intere di analisi del web.

Un'ultima tecnica di crawling possibile è il **focused crawling**: in questa tecnica il crawler va a visitare solo le pagine che sono considerate rilevanti rispetto ad un topic specifico.

LINK ANALYSIS

La presenza di un link all'interno di una pagina identifica un segnale di qualità; la pagina che referencia un'altra pagina con un hyperlink generalmente va a descrivere il contenuto della pagina referenziata tramite un anchor text.



I link che referenziano una pagina identificano una misura di popolarità della stessa e, in genere, le pagine popolari contengono informazioni più rilevanti rispetto a pagine non popolari. Il concetto di popolarità di una pagina viene analizzato in due modi differenti che determinano due approcci differenti:

- 1- **Popolarità indipendente dalla query**: PageRank simula una navigazione casuale del web e calcola il grado di probabilità di raggiungere una pagina p
- 2- **Popolarità dipendente dalla query**: HITS identifica le pagine più adeguate partendo da quelle reperite dall'utilizzo di una query o da quelle ad esse collegate.

L'algoritmo di PageRank di Google **simula una navigazione casuale di un utente nel web**. Una pagina avrà un alto coefficiente di PageRank se la somma dei link che la referenziano è alta. In **generale il coefficiente di PageRank è influenzato dal numero di link entranti** che identificano una referenza **e dal coefficiente di PageRank delle pagine che stanno referenziando**. Il processo di questo algoritmo è modellato sulla base di catene di Markov dove **gli stati sono le pagine web** e i **passaggi da uno stato e l'altro** sono identificati dai **link di referenza**.

L'algoritmo **HITS** (Hypertext Induced Topic Search) **parte dall'utilizzo di una query per reperire delle pagine**, successivamente identifica due tipi di pagine:

- Pagine **autorevoli**: pagine che **contengono informazioni rilevanti**. L'autorità di una pagina è **influenzata dal numero di link entranti**.
- Pagine **Hub**: pagine che **puntano a pagine utili**. Una pagina per essere un Hub deve avere **molte link uscenti che referenziano pagine autorevoli**.

Ogni pagina quindi possiede uno score di Hub e autorità; grazie a questi score è possibile determinare se una pagina è **rilevante**. Inoltre, l'algoritmo HITS utilizza **un'iterazione per calcolare i punteggi Hub e authority**, in modo che l'autorevolezza e la rilevanza si influenzano a vicenda.

17- EVALUATIONS

L'obiettivo primario di un search engine è quello di **reperire quanti più documenti rilevanti possibile**, andando al contempo a cercare di **minimizzare il numero di documenti non rilevanti reperiti**. Ci sono moltissimi criteri di valutazione, due di questi sono **l'efficacia e l'efficienza**. **L'efficacia** si riferisce alla **capacità di un sistema di recuperare i documenti rilevanti per una determinata query**. È misurata utilizzando metriche come la Precision, il Recall e la Mean Average Precision (MAP). Un sistema che ha **un'elevata efficacia** è in grado di recuperare un **elevato numero di documenti rilevanti** per una data query. **L'efficienza** si riferisce alla **velocità con cui un sistema recupera i documenti rilevanti**. È misurata utilizzando metriche come il tempo di risposta o il numero di documenti analizzati per ottenere i risultati. Un sistema che ha **un'elevata efficienza** è in grado di recuperare i documenti rilevanti in modo rapido e con un basso utilizzo di risorse.

PRECISION E RECALL

Per eseguire dei benchmark bisogna utilizzare delle misure di valutazione delle prestazioni; due di queste sono la **precision e la recall**. La **precision** individua il **rapporto tra documenti rilevanti reperiti e totale documenti reperiti**, la **recall** invece individua il **rapporto tra documenti rilevanti reperiti e il totale di documenti rilevanti reperibili**.

	Relevant	Nonrelevant
Retrieved	tp	fp
Not Retrieved	fn	tn

$$\bullet \text{ Precision } P = tp / (tp + fp)$$

$$\bullet \text{ Recall } R = tp / (tp + fn)$$

Un **alto** valore di **precision** e un **basso** valore di **recall** individua un **insieme di documenti tutti rilevanti che però identificano solo una piccola porzione di tutti i documenti rilevanti reperibili**. Un valore **basso** di **precision** e un valore **alto** di **recall** individua un **insieme di documenti composto da molti documenti non rilevanti e da altri documenti rilevanti che però ricoprono quasi l'intera totalità di documenti rilevanti reperibili**. Nel caso in cui sia **precision** che **recall** siano alti avremo un insieme di documenti quasi solo rilevanti che ricoprono quasi tutto l'insieme di documenti rilevanti reperibili.

RANKED-BASED MEASURES – MAP, P@K E NDCG


Vi sono altri tipi di misure di valutazione delle prestazioni che si basano sul concetto di ranking che sono:

- **Binary** relevance: per rilevanza binaria si intende **valutare i documenti solamente con due possibili stati** ovvero rilevante e non rilevante. Alcune misure per valutare questo tipo di rilevanza sono la **Precision@K**, la **Mean Average Precision** (MAP) e la Mean Reciprocal Rank (MRR)
- Multiple levels of relevance: **Normalized Discounted Cumulative Gain** (NDCG)

La **Precision@K** è una misura che valuta la quantità di documenti rilevanti presenti **tra i primi K risultati**, dove **K è un valore settato per la valutazione**. Allo stesso modo è possibile eseguire la misura della **Recall@K**.

La **Mean Average Precision** calcola la **precision media e diversi livelli di recall**; considera la posizione di ranking di **ogni singolo documento rilevante** e ne calcola la Precision@K; una volta calcolata la Precision@K per ogni documento rilevante si fa **la media tra le precision** così calcolate.

• Average precision = average of P@K

• Ex:  has AvgPrec of $\frac{1}{3} \cdot \left(\frac{1}{1} + \frac{2}{3} + \frac{3}{5} \right) \approx 0.76$

In questo modo si va a **calcolare per ogni singola query la precision media per i top K risultati ottenuti per poi farne, in conclusione, una media**.

Per quanto riguarda le **misure non binarie** di valutazione abbiamo la **Discounted Cumulative Gain**. Questa misura **calcola la rilevanza dei documenti reperiti tenendo però anche in considerazione l'ordine dei documenti risultati**; in sostanza una volta calcolata la rilevanza per ogni documento reperito si va a **moltiplicare la stessa per un fattore che ne diminuisce l'importanza tanto più tale documento è in basso nell'elenco dei documenti reperiti**. Per andare a sopperire il problema del **numero variabile di documenti rilevanti risultati in base alla query di ricerca**, di solito si usa la versione **normalizzata** della DCG, ovvero la NDCG. Questa misura normalizzata è definita dal **rapporto tra la DCG e la IDCG**, ovvero la ideale DCG che corrisponde allo score massimo ottenibile dalla DCG dato un determinato set di query.

19- LABORATORIO 1 – TEXT PRE PROCESSING E NORMALIZATION

20- LABORATORIO 2 - WORD EMBEDDINGS

L'obiettivo del laboratorio è creare una rappresentazione vettoriale di un testo. In questo laboratorio si utilizzeranno gli **embeddings che creano un vettore di rappresentazione delle parole di uno spazio dimensionale inferiore** rispetto ai vettori one hot, **mantenendo al contempo le proprietà delle parole**.

Un primo embedding è il **Word2Vec che raggruppa parole simili basandosi sulla co-occorrenza delle stesse**. Le architetture possibili da utilizzare per questo approccio sono la **CBOW e lo Skip-Gram**.

In codice python il modello si implementa così:

```
from gensim . models import Word2Vec
# Train the model
model = Word2Vec ( sentences =corpus , vector_size =100 , window =5,
min_count =1, workers =4)
# Save the model
model . save ( path )
```

Dove **vector_size** è la dimensione dello word space e **window** è la dimensione del contesto da prendere in considerazione.

Un altro modello di embedding è il **Glove** che, a differenza di W2V che utilizza solo un contesto locale, **utilizza la matrice di co-occorrenza per generare l'embedding**. Per entrambi questi metodi esistono dei **modelli pre-trained** dalla libreria **gensim**:

```
from gensim import downloader
# See all the pretrained models available :
print ( list ( gensim . downloader . info ()['models']. keys ()))
# load a pre - trained model
model = downloader . load ('word2vec -google -news -300 ')
# train a model even further for 5 epochs
model . train ( new_corpus , total_examples =50 , epochs =5)
# get a word vector ( embedding ) in numpy
w_emb = model .wv['word ']
# get top 10 similar words
sim_words = model .wv. most_similar ('word2 ', topn =10)
```

Se volessimo creare un **bigram** con gensim dobbiamo fare:

```
from gensim . models import Phrases
bigram_generator = Phrases ( tokenized_data , ...)
bigram_tokens = bigram_generator [ tokenized_data ]
```

Allo stesso modo se volessimo creare un **trigram** avremo:

```
trigram_generator = Phrases ( bigram_tokens , ...)
trigram_tokens = trigram_generator ( bigram_tokens )
```

L'embedding delle parole è utile, tuttavia alle volte è necessario eseguire un **embedding dei documenti**. Esiste il **Doc2Vec** che funziona in maniera simile al Word2Vec con l'aggiunta di un id paragrafo come token.

21- LABORATORIO 3 – INFORMATION RETRIEVAL

Per la fase di information retrieval utilizzeremo la libreria **pandas** di python. **Questa libreria è utile per la data manipulation**; in particolare per eseguire il processo di **data loading** si può fare:

```
import pandas as pd
import numpy as np
# Read csv as a pandas Dataframe
data = pd. read_csv ('path /to/ dataframe .csv ', sep=',')
# view the dataframe
data . head (5) # shows the first five rows
data . tail (3) # shows the last three rows
data . columns # show the columns names
data . index # show the index columns
data . describe () # basic data statistics
```

Per il **data accessing** possiamo fare:

```
data . to_numpy () # converts the dataframe to a numpy matrix
data . to_dict () # converts the dataframe to a python dictionary
# Access data
data [['column_a ', 'column_b ']] # gets the data in specific column
data [5:10] # get row 5 to 10
# Selection by explicit label
# get all rows and mentioned columns
data . loc[:, ['column_a ', 'column_b ']]
# Selection by implicit position
# get rows 5-10 and all columns
data . iloc [5:10 , :]
```

Per il **data filtering** si può fare:

```
# Create boolean conditions for each row
cond_1 = data ['column_a'] > 5
cond_2 = data ['column_b'].str.contains('this string')
data_cond_1 = data [ cond_1 ] # data that satisfies cond_1
data_cond_1_and_2 = data [ cond_1 & cond_2 ] # satisfies both cond_1
and cond_2
data_cond_1_or_2 = data [ cond_1 | cond_2 ] # satisfies at least one
# You can also use functions , for example for cond_2
cond_2 = data ['column_b'].apply ( lambda x: 'this string' in x)
# we use lambda function but any function is good
```

Per il **data aggregation** si può fare:

```
# operation aggregating whole dataset
data . mean () # average of all columns
data . mean ( axis =1) # average of all rows
# apply a function to the whole dataset
data . apply ( lambda x: x.min () - x.max ())
# change axis to change where the function is applied
# Groupby and some operation on grouped data
data . groupby (" column_a ")[[" column_b ", " column_c "]].sum ()
data . groupby (" column_a ").agg({
" column_b ": "sum",
" column_c ": " count " # any function
})
```

Per il **data join** si può fare:

```
# to do a join similar to SQL
data . merge ( right_data , on='column ')
# if column names are different
pd. merge (data , right_data , left_on ='data_column ',
right_on =' right_column ')
# to concatenate two dataframes
pd. concat ([data , right_data ])
```