Text Analysis (Parte 3)

9. Search Engine Implementation

- Un sistema di IR si compone di 4 componenti fondamentali — Ognuna con un ruolo essenziale per assicurare efficienza e qualità dei risultati
- **Tokenizer**
  - Responsabile della segmentazione dei documenti in termini significativi
  - Ogni documento viene trasformato in un vettore
    - Indice — Parola
    - Valore
      - Numero di occorrenze
      - Può essere anche uno smoothed term count — O un TF-IDF
  - Tokenizer basico — **Whitespace tokenizer**
  - Il tokenizer può anche assegnare ID ai documenti ed è la base della generazione delle feature
  - È importante una buona tokenizzazione, altrimenti gli errori si propagano nelle fasi successive
- **Indexer**
  - I moderni motori di ricerca sono in grado di indicizzare dati in quantità maggiore della memoria disponibile
  - L'indexer costruisce strutture dati che permettono una ricerca efficiente anche su colle zioni enormi
  - Utilizza una struttura dati chiamata **inverted index**
    - Permette di ricercare velocemente i documenti che contengono un certo termine
    - Comprende
      - **Lexicon** — Tabella che mappa ogni termine unico alla sua frequenza e alla posizione nel file di posting
      - **Postings file** — Mappa ogni termine ai documenti in cui appare e alle sue frequenze
  - Questo permette di evitare la scansione completa dei documenti durante una query
- **Scorer/Ranker**
  - Una volta creato un inverted index si può creare uno scoring dei documenti che hanno almeno un elemento della query
  - In questo modo invece che avere un forward index con documenti e i loro termini, possiamo fare una ricerca molto più efficiente
  - **Term-at a time Tanking**
    - Per ogni termine
      - Fetch dei corrispondenti conteggi nell'inverted index
      - Si crea un accumulatore per fare scoring dei documenti
      - Si analizza l'inverted index per trovare il termine corrente — Per ogni documento che lo contiene si aggiorna l'accumulatore del documento
    - Basta alla fine ordinare i documenti secondo il loro score e ritornare i migliori k
- **Feedback/Learner**
  - Il modulo di feedback apprende dal comportamento dell'utente per migliorare la classifica dei risultati futuri — Può essere molto sofisticato e adattarsi dinamicamente
  - **Tipi**
    - **Relevance Feedback**
      - Si basa sull'interazione dell'utente con i risultati
      - Gli utenti indicano (esplicitamente o implicitamente) quali documenti sono rilevanti
      - Il sistema può così aggiornare la query per ottenere risultati migliori
      - Non realizzabile siccome bisogna chiedere all'utente uno sforzo ulteriore
    - **Pseudo Relevance Feedback**
      - Si assume che i primi k documenti siano rilevanti, anche senza il giudizio dell'utente
      - Poi vengono usati per migliorare la query
      - Non è sempre affidabile — Bisogna decidere una soglia arbitraria di documenti a cui affidarci e sperare che la funzione di ranking sia in grado di ottenere documenti utili
    - **Implicit Feedback**
      - Non si chiede all'utente un giudizio esplicito, ma si osservano i suoi comportamenti, come i click
      - Se un documento viene cliccato, si presume sia rilevante
      - Questa informazione può essere usata per adattare dinamicamente la classifica futura
  - **Feedback nel vector space model**
    - Nel VSM il feedback viene usato per modificare il vettore della query in modo che sia in una posizione migliore dello spazio — Ovvero vicina ai documenti rilevanti
    - Lo spostamento può avvenire tramite la modifica dei pesi dei termini oppure con l'aggiunta di termini simili (**query expansion**)
    - Il metodo più efficace per il VSM è il **Rocchio Feedback** — La nuova query"qm viene calcolata come
      - $$\vec{q}_m = \alpha\vec{q} + \frac{\beta}{|D_r|}\sum_{\vec{d}_r \in D_r}\vec{d}_r - \frac{\gamma}{|D_n|}\sum_{\vec{d}_n \in D_n}\vec{d}_n$$
        - $\vec{q}$ — Query originale
        - Dr — Insieme dei documenti rilevanti
        - Dn — Insieme dei documenti non rilevanti
        - α — Peso che controlla i termini della query originale
        - β — Peso che avvicina la query al centroide dei termini dei documenti rilevanti
        - γ — Peso che allontana la query dal centroide dei dei termini dei documenti non rilevanti
      - β > γ
        - Si vuole polarizzare le query verso i risultati rilevanti
        - È comune ridurre γ o escludere del tutto i documenti non rilevanti per evitare "rumore" nel nuovo vettore di query

10. Text Classification

- Consiste nell'assegnare una categoria, tema o genere a un documento — Casi d'uso
  - Assegnare categorie, argomento, genere
  - Rilevamento dello spam
  - Riconoscimento dell'autore
  - Analisi del sentiment
- **Formalizzazione**
  - Dati
    - Documento d
    - Insieme di classi Y = {y1 , . . . , yJ }
  - Output — Previsione della classe y ∈ Y
- **Metodi**
  - **Regole a mano**
    - Regole basate su combinazioni di parole o altre features — Ex. Se contiene "dollari" e "sei stato selezionato" è spam
    - L'accuratezza può essere alta — Se le regole sono ben fatte
    - **Problema** — Costruire e mantenere una struttura del genere è molto complesso
  - **Machine learning supervisionato**
    - Input
      - Documento
      - Classi
      - Un training set di m documenti associati alle classi
    - Output — Modello trainato che associ un documento a una classe
    - Tipi
      - Naive bayes
      - Logistic Regression
      - SVM
      - K-Nearest Neighbors
- **Sentiment Analysis: gestione della negazione**
  - Classificazione del testo per i sentimenti
  - Dobbiamo gestire la negazione — Si consideri la differenza tra le frasi
    - I really like this movie — Positiva
    - I didn't really like this movie — Negativa
  - Una baseline usata durante la normalizzazione del testo è quella di mettere NOT dopo ogni parola dopo un token di negazione logica — I didn't like this movie diventa — I didn't NOT like NOT this NOT movie
- **Computational Lexical Semantics**
  - La semantica lessicale computazionale studia il significato delle parole in modo automatico
  - Per rappresentare il significato di un termine si parte dal **lemma**
    - La forma della parola che viene usata per presentarla nel vocabolario
    - Poi si passa alle forme che un lemma assume, ovvero le **wordforms**
  - Si occupa di
    - **Calcolare la similarità delle parole** — Se esprimono lo stesso concetto (persona e individuo)
    - **Calcolare le relazioni fra le parole**
      - Antonimia — (bello e brutto, sono contrari)
      - Supertipo-sottotipo
    - **Disambiguazioni** — Distinzioni dei significati sulla base del contesto
    - **Ruolo semantico** — Capire di cosa parla una frase e individuare soggetti, predicati e complementi
    - **Sentiment analysis**
- **Word Sense Disambiguation (WSD)**
  - Disambiaguazione del significato
  - Fondamentale per capire quale senso ha una parola in un certo contesto
  - Tecniche
    - **Supervised Machine Learning**
      - Richiede un corpus annotato manualmente con i sensi corretti delle parold
      - Si basa su un classificatore supervisionato
    - **Metodi lessicali (thesaurus/dictionary)**
      - Il labeling è costoso
      - Potremmo utilizzare una supervisione indiretta da dizionari e theseaurus
        - I testi in input non sono labeled
        - Sono anche detti weakly supervised
      - **Approccio classico** — **Lesk Algorithm**
        - Sceglie il significato la cui definizione condivide più parole con quelle vicine alla parola target
        - Il senso con il maggiore numero di parole sovrapposte viene scelto
        - Esempio
          - The bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities
          - La parola bank ha due significati

bank[1]	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
bank[2]	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"

Banca o Riva

The **bank** can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

ank[1]	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
ank[2]	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"

Scegliamo banca siccome le parole deposit e mortgage sono presenti nel dizionario

        - **SImplified Lesk**

```
best-sense ← most frequent sense for word
max-overlap ← 0
context ← set of words in sentence
for each sense in senses of word do
  signature ← set of words in the gloss and examples of sense
  overlap ← COMPUTEOVERLAP(signature, context)
  if overlap > max-overlap then
      max-overlap ← overlap
      best-sense ← sense
end
return(best-sense)
```

Versione semplificata di lesk

          - ComputeOverlap ritorna le parole comune i due insiemi, ignorando quelle blacklisted

11. Vector Semantics and Embeddings

- Quando non si ha a disposizione un dizionario per ricavare il significato della parola, si fa uso del **contesto**
  - Termini in contesti simili hanno significati simili
  - **Il significato di una parola è relativo alla distribuzione delle parole nel suo intorno**
- I metodi che utilizzano la distribuzione delle parole nell'intorno del target sono detti **distributivi**
- Ogni parola viene genralente rappresentata come un vettore in uno spazio n-dimensionale (**vector semantics**)
  - Le parole simili semanticamente si trovano vicine nello spazio
  - Fondamentalmente vogliamo che il vettore di una parola w sia vicino alle parole del suo stesso contesto
- Il vettore assegnato alla parola w è detto **embedding** — Questi vettori indicano il significato di una parola come quanto questa appaia nell'intorno di altre parole
- **Quattro tipi di modelli vettoriali**
  - **Rappresentazioni sparse** — Matrici di co-occorrenza
  - **Rappresentazioni dense**
    - Singular Value Decomposition
    - Neural-Network-Inspired Models
      - Skip-grams
      - CBOW — Continuous Bag Of Words
    - Contextualized Embeddings
- I metodi vettoriali e distributivi si basano su una matrice di co-occorrenza
  - Rappresenta quanrto spesso le parole co-occorrono
  - Due tipi
    - **Termine-Documento**
      - Riga = parola del dizionario
      - Colonne = documenti di una collezione
      - Ogni cella indica il numero di volta hce la parola compare nel documento
      - Il documento è un vettore contenente i conteggi
      - Erano pensate per trovare documenti simili — Ma documenti simili avranno al contempo termini simili, e quindi vettori simili
    - **Termine-Termine**
      - Righe e colonne contengono parole
      - Rappresenta megli i vettori delle parole
      - Ogni cella contiene il numero di volte che la riga e la colonna co-occorrono in qualche corpus di training
      - Il contesto può essere il documento
        - Però è più comune usare contesti ristretti (ex. 4 parole da sx e dx)
        - Più la finestra e corta più le rappresenzaioni sono sintattiche — Dato che l'informazione proviene dalle parole vicine
        - Più la finestra è ampia, maggiori sono le relazioni di tipo semantico
      - Molti valori saranno 0, elementi sparsi
  - Invece che usare il conteggio si può fare uso di
    - Tf-idf — Per penalizzare l'eccessiva frequenza
    - Usare **PMI**
      - **Pointwise Mutual Information**
      - Misura di quanto spesso due eventi x e y occorrono insieme, rispetto alla loro indipendenza — $I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$
        - $$\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$ Nel nostro caso, probabilità che la parola w sia nel contesto di un'altra parola fratto la probabilità che le due parole compaiano nel documento, senza alcuna relazione tra le due
      - Essendo che il numeratore e spesso piccolo, la quantità risulta < 1, e di conseguenza il logaritmo negativo
        - $$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$ Si fa uso della Positive PPI
- **Uso del coseno per misurare la similarità**
  - Non uso il dot product classico — Siccome va a premiare troppo i vettori lunghi
  - Dot product normalizzato
    - 1 se i vettori puntano nella stessa direzione
    - 0 vettori ortogonali
    - -1 vettori opposti
  - Dal momento che le frequenze grezze sono non negative, il coseno per questi vettori varia da 0 a 1
- **Avere gli embedding densi**
  - Gli embedding sono vettoroni, molto informativi, ma pieni di zeri
    - Diventa problematico quando devo muovere diverse parole
    - Oltretutto rimuovendo informazioni molto specifiche, ho feature molto più adatte alla generalizzazione
  - **Singular Value Decomposition**
    - Metodo che cerca di individuare le dimensioni più importanti di un dataset
    - Quello che ci interessa è catturare la maggior parte di varianza del dataset originale, in modo da conservare quanto più contenuto informativo possibile — Simile a quello che fa PCA
    - Questo metodo di riduzione è indicato come **Latent Semantic Analysis**
    - SVD scompone la matrice in input in tre matrici — $A_{[m \times n]} = U_{[m \times r]}\ \Sigma_{[r \times r]}\ (V_{[n \times r]})^T$
      - La matrice sigma ha valori sulla diagonale principale — I valori sono dimensionati dal più grande al più piccolo
      - Se uso solo i primi elementi di sigma (ex.10)
        - Ottengo un valore che è diverso dalla matrice iniziale
        - Ma con i valori più rilevanti
      - La matrice che otterremo sarà densa
    - Otteneniamo una matrice che, oltretutto, contiene embedding che escludono informazioni più specifiche — Mi permette di generalizzare meglio

12. Word2Vec/Skip-Gram

- Non vuole predire la parola che segue, vuole predire se la parola possa seguire o meno il contesto di un altra parola — Rende il problema booleano
- Non ci interessa la predizione — Utilizziamo i pesi per creare gli embedding della parola
- Funziona con una Logistic Regression che opera su corpus testuali
- Esistono due varianti principali
  - **Skip-Gram** — Predice il contesto dato il centro
  - **CBOW** — Predice la parola centrale dato il contesto
- Nel modello Skip-Gram con Negative Sampling, si allena un classificatore binario per distinguere coppie reali di parole da quelle campionate casualmente
  - I pesi del classificatore diventano le embeddings delle parole
  - Fondamentalmente, non ci interessa la previsione, ma ci bastano i pesi del modello
  - Questi saranno i nostri embeddings
  - Il dataset per l'allenamento è fondamentalmente "gratis" — Intuizione — **Self-supervision**
    - Non abbiamo bisogno di etichette esterne
    - Il testo stesso ci fornisce i dati di addestramento
    - Prendiamo del testo grezzo, come frasi normali
    - Il modello crea da solo i propri esempi supervisionati
    - Ogni parola nel contesto agisce da etichetta o obiettivo da predire
    - Ex. — Se stai osservando la parola centrale apple, e vedi che la parola fruit è vicina — Puoi usarla come etichetta corretta in un esempio di training
- **Intuizione**
  - Per ogni parola target w, si prendono le parole nel suo contesto ±c
  - Si trattano le coppie (w, ci) come esempi positivi
  - Si campionano parole casuali come esempi negativi — Per ogni esempio positivo, ne ho k negativi — Parametro empirico
  - Si allena un classificatore che distingua i due casi
  - Si usano i pesi di regression come embeddings
- **Scopo**
  - Avvicinare la probabilità di w alle parole vicine
  - Allontanare le parole distanti