

Document Semantic Similarity

TIS Project

Alberto Pirovano Francesco Picciotti

Politecnico di Milano

4th May 2017



1 State of art

- NLP tradizionale
- Vector Space Model
- Deep Learning

2 Data Preparation

- Preprocessing
- Cleaning del testo

3 Word2Vec

4 Doc2Vec



Le tecniche adottate attualmente per trovare la **similitudine semantica tra testi** si basano su tre approcci:

- 1 NLP Tradizionale
- 2 Vector Space Model
- 3 Deep Learning based



- 1 State of art
 - NLP tradizionale
 - Vector Space Model
 - Deep Learning
- 2 Data Preparation
 - Preprocessing
 - Cleaning del testo
- 3 Word2Vec
- 4 Doc2Vec



Questo approccio si basa sull'utilizzo delle tradizionali tecniche di **Natural Language Processing** e si costituisce dei seguenti step:

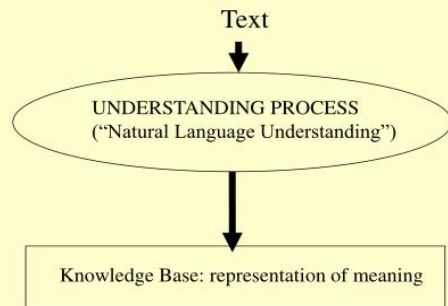
- Cleaning dei dati
- Pos-Tagging
- Stemming o Lemmatisation
- Parsing
- Ontologia

Tuttavia, dato che il nostro lavoro è molto **sensibile** e **dipendente** dalla qualità dei tool utilizzati, abbiamo trovato alcune consistenti **criticità** riguardanti:

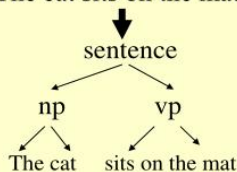
- **L'affidabilità** del Pos-Tagger italiano di TreeTagger
- **Reperire** una Ontologia e un parsing toll per la lingua italiana



NLP: the process



"The cat sits on the mat"



↓

Fact(type: statement,
agent: cat-002,
action: sits_on,
object: mat-001)

Fact(type: statement,
agent: Fido,
action: is_a,
object: cat)

Fact(type: statement,
agent: Freda,
action: loves,
object: Fido)

Natural Language Processing

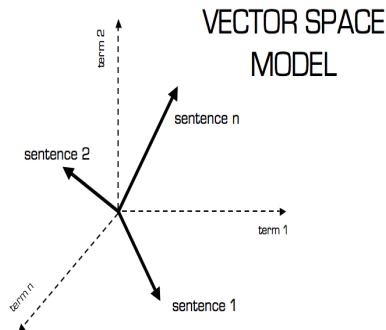


- 1 State of art
 - NLP tradizionale
 - **Vector Space Model**
 - Deep Learning
- 2 Data Preparation
 - Preprocessing
 - Cleaning del testo
- 3 Word2Vec
- 4 Doc2Vec



Vector Space Model - Explanation

Differentemente dal precedente, questo approccio ha le sue basi nello sviluppo di una **rappresentazione geometrica e vettoriale** per i documenti testuali. **Documenti e query** sono rappresentati da vettori con un numero di elementi pari al numero di termini presenti nel vocabolario. Tipicamente i termini sono le **parole distinte** presenti nell'insieme di documenti. A valle di questa rappresentazione vengono spesso utilizzate le **operazioni vettoriali** per confrontare due **documenti**.



Nel vector space model proposto da **Salton, Wong and Yang** i vettori sono composti da **weights**, ognuna associata ad un termine del dizionario e calcolata tramite **tf-idf**.

Considerando un **documento** d_j , questo viene rappresentato tramite un **vettore** d_j :

- $d_j = (w_{1,j}, w_{2,j}, \dots, w_{t,j})$ **dove:** $w_{i,j} = tf_{t,d} * \log \frac{|D|}{|\{d' \in D | t \in d'\}|}$



Latent Semantic Analysis è una tecnica di **Topic Modelling** che si colloca a valle del **document encoding** con **tfidf**.

È una tecnica di **feature extraction** (PCA) che permette di migliorare significativamente la qualità di un lavoro di **clustering**, dato che la metrica non considera la differenza tra **features non importanti**.

Questa procedura viene usata per generare una categorizzazione di un **set di documenti** in un **set di topic** o anche per osservare le parole che descrivono un certo topic.

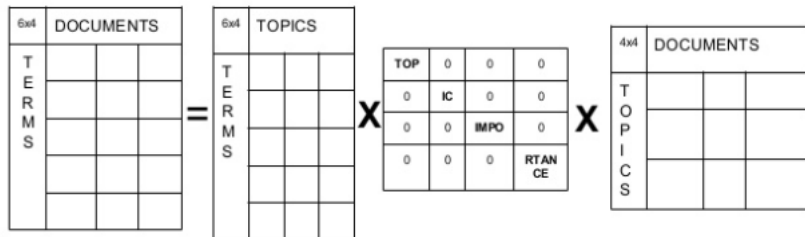
Si basa sulla creazione di una **Document-Term Matrix** nella quale le **righe** rappresentano le parole del **Bag Of Words** e ha una **colonna** per **documento** nel corpora.

Il cuore di questa procedura sta nella **riduzione della dimensionalità** di questa matrice tramite **SVD**.



Vector Space Model - LSA - SVD

Find three matrices U , Σ and V so that: $X = U\Sigma V^T$



Questa procedura ci permette di:

- **Estrarre** quanti topic desideriamo da un set di documenti.
- **Conoscere** la rilevanza di un certo topic dopo averlo estratto, in questo modo siamo in grado di fermare il processo di estrazione quando i topic cominciano a diventare poco significativi.
- **Categorizzare** documenti in topic
- **Descrivere** topics con le parole del **Bag Of Words**.



Vector Space Model - LSA - Example

In questo esempio possiamo osservare la riduzione di dimensionalità:

LSA is essentially low-rank *approximation* of document term-matrix

Word assignment to topics

IT cars

3	4	1	0
4	3	0	1
3	4	4	3
0	1	4	3
2	0	3	3
0	1	3	4

=

	IT	cars
linux	-0.33	-0.53
modem	-0.32	-0.54
the	-0.62	-0.10
clutch	-0.38	0.42
steering	-0.36	0.25
petrol	-0.37	0.42

X

Topic importance

11.4	
	6.27

X

Topic distribution across documents

	D1	D2	D3	D4
IT	-0.42	-0.48	-0.57	-0.51
cars	-0.56	-0.52	0.45	0.46

Il processo di LSA permette di costruire le 3 matrici che vediamo sopra, ognuna con una sua utilità:

- 1 Word assignment to topics
- 2 Topic importance
- 3 Topic distribution across documents

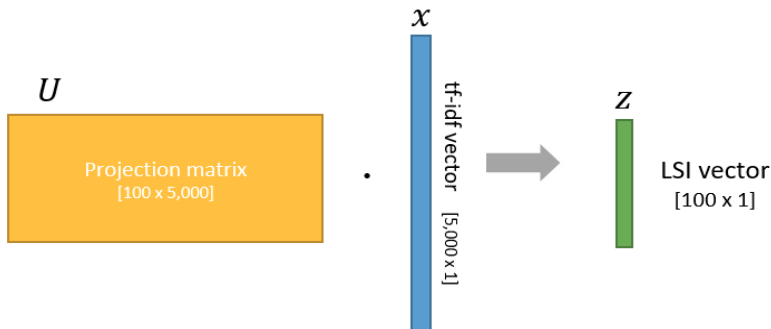


Vector Space Model - LSA - Example

Ipotizzando un numero di topics pari a 100 e la cardinalità del **Bag of Words** pari a 5000, per ottenere la rappresentazione dei documenti in termini di topics, cioè un vettore \mathbf{z} , dobbiamo:

- 1 **Vettorizzare** un documento in \mathbf{x} .
- 2 **Proiettare** il **tfidf** vector \mathbf{x} sul topic space.

Se definiamo la matrice $U = (\text{Wordassignment to topics})^T$ possiamo visualizzare la **proiezione** in questo modo:



Vector Space Model - Algorithm

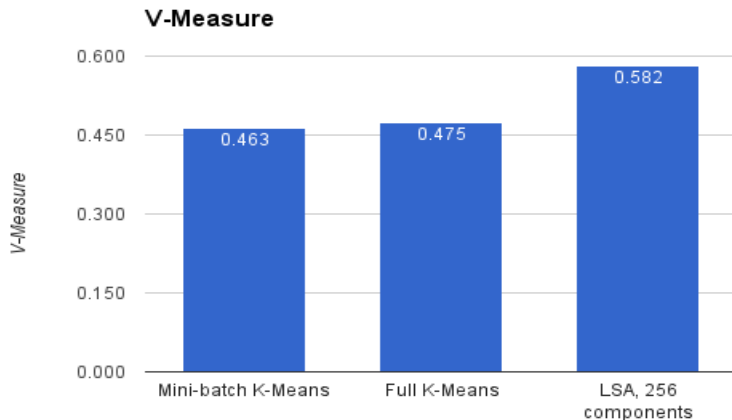
Riassumendo, possiamo individuare i seguenti **passaggi**:

- 1 Cleaning dei dati (Stemming o Lemmatisation)
- 2 Vectorization using TF/IDF (\mathbf{x} nella figura nella slide precedente)
- 3 LSA (optional) (\mathbf{z} nella figura precedente)
- 4 Clustering using similarity measure (Cosine, Pearson, ...) tra \mathbf{x} vectors o \mathbf{z} vectors



Vector Space Model - LSA - Performance

Come possiamo vedere nella figura, utilizzare le classiche tecniche di **clustering** senza **LSA** può portare una riduzione rilevante delle **performances**:



Vector Space Model (con LSA) - Pros and Cons

Pros:

- Modello semplice basato sull'**algebra**.
- Le **weights** non sono binarie.
- Permette di calcolare un grado di **similarità continuo**.

Cons:

- Non adatto a trattare **lunghi documenti**, infatti a causa della alta **dimensionalità** il valore **tfidf** dei componenti si riduce, riducendo di conseguenza il **dot-product**. (**LSA fixes**)
- **Fortemente sensibile a Falsi Negativi**, infatti documenti nello stesso **contesto** ma con diversa terminologia non saranno considerati simili. (**LSA fixes**)
- **Perde l'ordine delle parole**.
- **Pre-processing** dipendente.



- 1 State of art
 - NLP tradizionale
 - Vector Space Model
 - Deep Learning
- 2 Data Preparation
 - Preprocessing
 - Cleaning del testo
- 3 Word2Vec
- 4 Doc2Vec



Il terzo approccio che proponiamo è molto diverso dai precedenti, infatti si basa sull'utilizzo di **neural networks**. La fondamentale differenza è che, mentre il secondo **definiva un algoritmo** per determinare le rappresentazioni vettoriali delle parole (tfidf), questo **definisce una neural network** con la task di imparare quell'algoritmo dai dati. Questa **NN** può essere **costruita** in due diversi modi, ognuno dei quali descrive **come** imparare la **word-representation** per ogni parola:

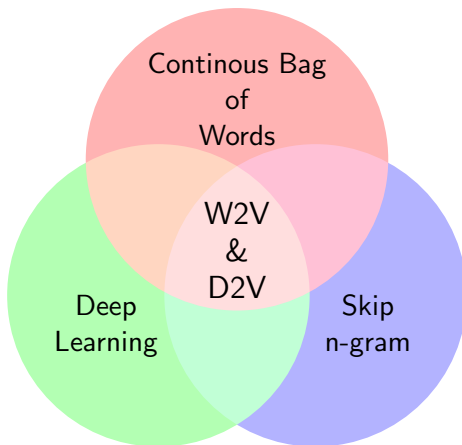
- **Continuous Bag Of Words model**
- **Skip Grammar model**

Dato che il processo di apprendimento è **unsupervised**, questi modelli permettono di **determinare la task** tramite la definizione di un target per ogni input.



Deep Learning based

Questa tecnica quindi si basa su 3 ambiti:



Deep Learning based: Word2Vec & Doc2Vec

Questo metodo è stato implementato da **Google** sotto il nome di **Word2Vec** e **Doc2Vec**. Il primo permette di determinare le **relazioni semantiche** che un particolare **corpus di testi** assegna ad un **Bag Of Words** di parole. **Doc2Vec** invece è una tecnica che si configura come una **estensione di Word2Vec** la quale, preso in ingresso un set di documenti (corpora), genera un **grado di similarità** reciproco.



Deep Learning based - Pros and Cons

Dopo una ampia **discussione**, seguita da una approfondita **analisi critica** di questi due approcci, siamo giunti alle seguenti **conclusioni**, che in termini di pro e contro si possono riassumere nel seguente modo:

Pros:

- Molto meno dipendente da un preprocessing
- **Context-aware**
- Combina il metodo *Geometrico* con quello *NLP Tradizionale*
- **Non sfrutta una ontologia, ma la crea**
- **Language independent**
- **Tiene in considerazione anche l'ordine delle parole**

Cons:

- Tecnica **unsupervised**
- Necessità di un esperto per **validare** la similarità
- Può risultare in **GIGO** system (Garbage In Garbage Out)



- 1 State of art
 - NLP tradizionale
 - Vector Space Model
 - Deep Learning
- 2 Data Preparation
 - Preprocessing
 - Cleaning del testo
- 3 Word2Vec
- 4 Doc2Vec



"Preprocessing is 80% of NLP work"

Lev Konstantinovskiy

Il **dataset** si suddivide in due corpora:

- il corpus del **Sole 24 Ore** con 3265 articoli, di cui 31 non hanno body
- il corpus di **Radiocor** con 6916 articoli

Il corpus prima del **preprocessing** contiene quindi 10150 articoli.

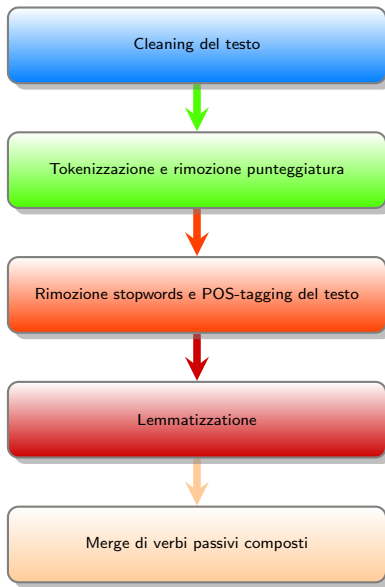
Togliendo i **duplicati** otteniamo 9283 articoli, cioè ci sono 867 articoli duplicati.



- 1 State of art
 - NLP tradizionale
 - Vector Space Model
 - Deep Learning
- 2 Data Preparation
 - Preprocessing
 - Cleaning del testo
- 3 Word2Vec
- 4 Doc2Vec



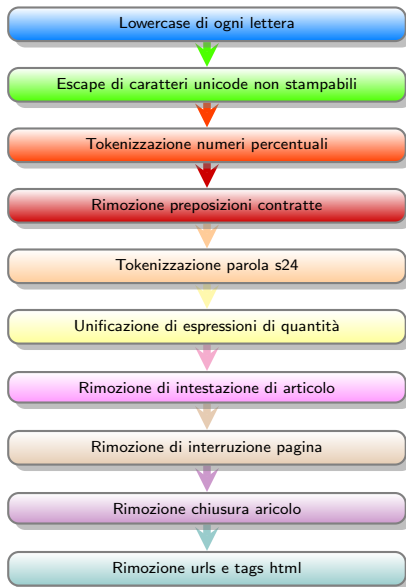
Pipeline completa



- 1 State of art
 - NLP tradizionale
 - Vector Space Model
 - Deep Learning
- 2 Data Preparation
 - Preprocessing
 - Cleaning del testo
- 3 Word2Vec
- 4 Doc2Vec



Cleaning pipeline



- 1 State of art
 - NLP tradizionale
 - Vector Space Model
 - Deep Learning
- 2 Data Preparation
 - Preprocessing
 - Cleaning del testo
- 3 Word2Vec
- 4 Doc2Vec



Qui spaghiamo per bene come funziona word2vec



- 1 State of art
 - NLP tradizionale
 - Vector Space Model
 - Deep Learning
- 2 Data Preparation
 - Preprocessing
 - Cleaning del testo
- 3 Word2Vec
- 4 Doc2Vec



Qui spaghiamo per bene come funziona doc2vec

