TLN TerzaParte

Mario Scapellato

Terza Parte dell'esame in Tecnologia del Linguaggio Naturale.
Prof. Luigi Di Caro

1 Prima e Seconda Esercitazione

1.1 Introduzione

La prima e la seconda esercitazione riguardano prima di tutto la creazione di definizione per 4 concetti scelti da noi studenti a lezione, dando, per ciascun concetto, 2 asserzioni concrete (di cui 1 generico e 1 specifico) e 2 astratte (di cui 1 generico e 1 specifico). Ogni studente ha inserito all'interno del documento una definizione per ciascun concetto in maniera indipendenten e sulla base della propria conoscenza sull'argomento stesso.

I 4 concetti scelti sono i seguenti :

- Generico Astratto : *Courage*
- Generico Concreto : *Paper*
- Specifico Astratto : *Apprehension*
- Specifico Concreto : *Sharpener*

Nel file defs.csv sono presenti le 30 definizioni assegnate per i 4 concettti illustrati.

1.2 Sviluppo

L'esercitazione e' stata sviluppata nella seguente maniera :

- Sono partito prima di tutto con una semplice implementazione di una funzione che mi permettesse di leggere riga per riga il mio csv.
- Successivamente ho eseguito le varie operazioni di Text Cleaning, ovvero :
  - *remove_punctuation* : operazione che mi permette di rimuovere tutte le punteggiature; data una frase contenente le punteggiature, verra' restituita una senza punteggiatura.
  - *remove_stopword* : mediante il file stop_words_FULL.txt ho eliminato tutte le stopword per ogni frase data in input.

- *tokenize_sentences*: suddivido una frase in unita' piu' piccole (come singole parole) e sucessivamente, ogni parola viene portata al suo lemma

- Sucessivamente ho definito la funzione *definitions* che mi legge il file .csv, definendo un dizionario formato da una coppia chiave-valore, dove la chiave e' il termine, mentre il valore rappresenta una lista di *Bag Of Words* correlate alle definizioni.

- Ho calcolato poi la similarita' del coseno : essa e' una misura che viene utilizzata per misurare la similarita' tra documenti durante la fase di text analysis. La formula e' la seguente :

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

  La similarita' del coseno prende in input 2 vettori inizialmente vuoti che corrispondono alle due definizioni che poi verranno prese in analisi. I vettori conterranno valore 1 se la parola analizzata appartiene al set, 0 altrimenti. Questo procedimento viene svolto per ogni singola parola che appare in entrambe le definizioni. Una volta fatta questa analisi, posso andare a calcolare della distanza del coseno e restituirne i risultati.

- Nella funzione definita *compute_results*, prendo in input il dizionario di liste contenente tutte le definizioni processate e vado a calcolare la similarita' del coseno tra tutte le coppie di definizion dello stesso concetto. Verra' restituito il valore medio della similarita' per ogni concetto.

- Nella funzione *most_frequent_words*, prendo in considerazione l'insieme delle parole presenti all'interno di ciascuna definizione e considero in input un dizionario che conterra' una lista di parole frequenti per tutte le definizioni. Per ogni concetto, vado a calcolare quelle parole che sono presenti in almeno il 50% delle definizioni. Verra' restituito l'elenco delle parole piu frequenti.

1.3 Risultati Ottenuti

I risultati ottenuti, come annunciato precedentemete a lezioni, non sono particolarmente elevati; in particolare :

```
Similiarita del coseno :  {'Courage': 0.21054727554969985, 'Paper':
0.29258850377799267, 'Apprehension': 0.083033031313557733, 'Sharpener':
0.386387871182442 4}
```

i risultati mostrano di come concetti piu' concreti come *Paper* e *Sharpener* sono descritti mediante delle definizioni piu' semplici e in linea tra loro, a differenza

dei concetti piu' generici come *Apprehension* e *Courage* che trovano maggiori difficolta' ad essere descritti da delle definizioni simili tra di loro. Inoltre, vediamo che le parole piu' frequenti presentano dei risultati piu' "attesi" :

```
Most Frequent words :
[('Courage', ['ability', 'fear']), ('Paper', ['write', 'material']),
('Apprehension', []), ('Sharpener', ['pencil', 'sharpen', 'tool'])]
```

2 Terza Esercitazione

2.1 Introduzione

Tra le tre proposte, e' stato scelto il task che riguarda la caratterizzazione delle risorse attraverso WordNet. L'obiettivo consiste nel ricercare i vari pattern (che fanno riferimento alle definizioni) all'interno delle ontologie e definire uno studio che riguado la forma (come la lunghezza), il contenuto o l'aspetto relazionale (come la presenza di iperonimi, antonimi ecc..).

2.2 Sviluppo

La verifica dei task di analisi e' stata fatta mediante l'utilizzo della risorsa lessicale Wordnet.

Gli studi effettuati sono stati principalmente 2:

- Nella prima parte, e' stata definita la lunghezza di ciascuna definizione. L'idea era la seguente : preso in considerazione un concetto (in questo caso erano i concetti che avevamo analizzato nell'esercitazione precedente, ovvero Courage, Paper, Sharpner e Apprehension), tramite il metodo *definitions_lenght(word)* verificavo la lunghezza di ciascuna definizione presente in WordNet associata a quel concetto.

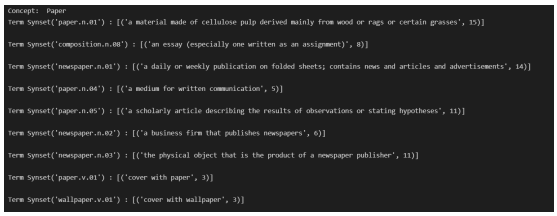
- Nella seconda parte dell'esercitazione ho focalizzato l'attenzione sugli aspetti relazionali: per ogni concetto, ho osservato il numero di definizioni associate, il numero totale di iperonimi e meronimi tramite i metodi *total_hypernym_path(word)* (con rispettiva ricerca della distanza dall'iperonimo e la radice che rappresenta il concetto) e *find_antonyms(word)*.

- Infine ho implementato un metodo aggiuntivo, definito *similarity_search(word)* che indica uno score di quanto sono simili le definizioni relativi agli iperonimi rispetto alle definizioni per i concetti analizzati. Per lo svolgimento e' stato utilizzato il modulo *sentence_transformer* che permette di calcolare rappresentazioni vettoriali che riguardano frasi, parafrasi ecc.. Fornisce una serie di modelli pre-addestrati dove la cui frase (o il testo) viene incorporato all'interno dello spazio vettoriale in modo che le frasi simili siano vicini tra loro e che venga calcolata con velocita' la similarita' del coseno. Il modello utilizzato per la rappresentazione vettoriale e' il *all-MiniLM-L6-v2*. Il valore restituito sara' uno per ogni synset del concetto in analisi:

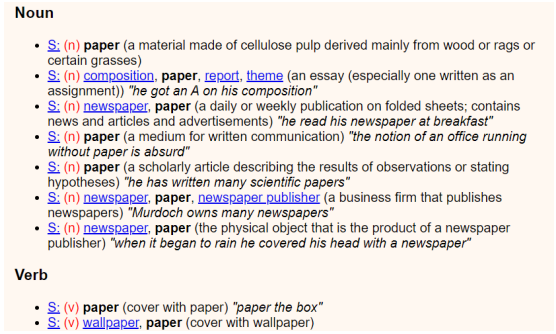
essi corrispondono al valore medio di somiglianza tra le definizioni di tutti gli iperonimi e la definizione della parola in analisi.

2.3 Risultati Ottenuti

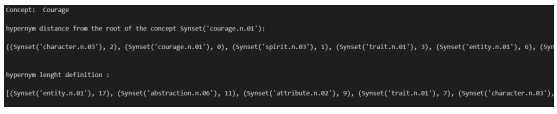
I risultati ottenuti sono suddivisi sulla base delle due parti svolte durante l'esercitazione:



Come possiamo vedere dalla figura sopra, per ogni concetto che analizziamo, ne riportiamo la rispettiva lunghezza della definizione (presa da WordNet):



Per quanto riguarda il secondo studio, vogliamo mostrare i risultati che riguardano la lunghezza della definizione per ciscun iperonimo del concetto analizzato e anche la rispettiva distanza tra la radice e il termine :



Stesso ragionamento e' stato fatto per la ricerca degli antonimi, anche se ne esiste solo uno per il concetto *Courage*:

```
------------------------

Concept:  Courage
Antonym : [(Lemma('cowardice.n.01.cowardice'), 1)]

------------------------

Concept:  Paper
Antonym : []

------------------------

Concept:  Apprehension
Antonym : []

------------------------

Concept:  Sharpener
Antonym : []
```

Se andiamo a dare un occhiata su WordNet vediamo che i risultati coincidono esattamente :

Noun

- S: (n) **courage**, courageousness, bravery, braveness (a quality of spirit that enables you to face danger or pain without showing fear)
  - *direct hyponym* / *full hyponym*
  - *attribute*
  - *direct hypernym* / *inherited hypernym* / *sister term*
  - ***antonym***
    - W: (n) cowardice [Opposed to: courage] (the trait of lacking courage)
  - *derivationally related form*

Infine vogliamo mostrare i risultati ottenuti dall'ultimo studio:

```
Concept:  Paper
concetto Synset('paper.n.01') :
Average Similarity for hypernyms : 0.2250049710273742 7

concetto Synset('composition.n.08') :
Average Similarity for hypernyms : 0.4687023162841797

concetto Synset('newspaper.n.01') :
Average Similarity for hypernyms : 0.5600683093070984

concetto Synset('paper.n.04') :
Average Similarity for hypernyms : 0.49245941638946533

concetto Synset('paper.n.05') :
Average Similarity for hypernyms : 0.23190630972385406

concetto Synset('newspaper.n.02') :
Average Similarity for hypernyms : 0.7873624563217163

concetto Synset('newspaper.n.03') :
Average Similarity for hypernyms : 0.31530842185020447

concetto Synset('paper.v.01') :
Average Similarity for hypernyms : 0.5753957033157349

concetto Synset('wallpaper.v.01') :
Average Similarity for hypernyms : 0.40710264444351196
```

Dove vediamo che solo alcune definizioni degli iperonimi sono mediamente vicine alla definizione del concetto *Paper* analizzato in questione.

3 Quarta Esercitazione

3.1 Introduzione

La quarta esercitazione prevede lo studio di acluni verbi dal punto di vista della Teoria di Hank. Secondo Hank il verbo e' la radice dek significato e non esistono espressioni senza verbo. Ad ogni verbo viene associata una valenza che indica gli argomenti che sono necessari per il verbo. Possiamo differenziare il significato del verbo in basa al numero di argomenti che possiede. Determinato il numero di argomenti per un certo verbo, bisogna specificarli mediante un certo numero di slot; ogni slot e' formato da un certo numero di valori che lo riempieno, detti *filler*. Ogni filler puo' avere associati dei tipi semantici che rappresentano delle generalizzazioni strutturate come una gerarchia.

3.2 Sviluppo

Per lo svolgimento e' stato utilizzato il seguente verbo :

- *Read,Reads,Readen*

Per recuperare le n frasi in cui il verbo viene utilizzato, e' stato utilizzato il corpus brown: vengono recuperate una serie di frasi dal corpus che contengono o il verbo presente *read e reads*, oppure la sua forma al present perfect *readen* Poiche' si tratta di recuperare una serie di frasi dal corpus, lavoreremo con una serie di contesti diversi a seconda del ruolo che viene svolto dal verbo in ciascuna frase. Per fare cio', effettuo l'algoritmo di lesk che permette di disambiguare. L'algoritmo di Lesk si basa sul fatto che, all'interno di una regione testuale, un insieme di parole condividono un significato simile. Verra' restituito il *best_sense* per una determinata parola a partire da un insieme di contesti presi da esempi per quella parola.

Per ogni frase, mediante l'ausilio di *spacy*, effettuando il pos e il parsing, ottengo i vari token collegati tramite le dipendenze sintattiche. Iterando sui vari token, e' possibile estrarre il soggetto e il complemento oggetto relativi al verbo di una frase. Soggetto e complemento oggetto vengono poi mappati con i relativi super sensi di WordNet. I supersensi di soggetto e complemento oggetto rappresentano i filler dei verbi. Le coppie dei supersensi sono contenute all'interno di cluster semantici opportunamente rappresentati. Sucessivamente, vengono calcolati per entrambi i filler i supersensi piu' frequenti.

3.3 Risultati Ottenuti

Vediamo che prima di tutto possiamo mostrare una parte del soggetto e del complemento oggetto estratto da ciascuna frase:

```
['You', 'read', 'various', 'guesses', 'on', 'how', 'many', 'Americans', 'are', 'camping', '.']

subj : You      obj: guesses
['Again', ',', 'the', 'oil', 'man', 'must', 'read', 'the', 'meters', 'at', 'such', 'intervals', 'as', 'he', 'finds', 'best', '.']

subj : man      obj: meters
['She', 'read', "Maitland's", 'Dark', 'Ages', ',', '``', 'which', 'I', 'enjoyed', 'very', 'much', "''", ';', ';']

subj : She      obj: Ages
```

La cui coppia di cluster semantici e' pari a :

```
Coppie Semantiche with count:
[('noun.substance', 'noun.artifact'), 1]
[('noun.person', 'adj.all'), 1]
[('noun.substance', 'noun.communication'), 4]
[('noun.substance', 'adv.all'), 1]
[('noun.person', 'noun.cognition'), 2]
[('noun.person', 'verb.social'), 1]
[('noun.location', 'noun.cognition'), 1]
[('adj.all', 'noun.group'), 1]
[('noun.person', 'noun.group'), 1]
[('noun.cognition', 'noun.person'), 1]
[('noun.quantity', 'noun.communication'), 2]
[('noun.substance', 'noun.attribute'), 1]
[('noun.person', 'noun.quantity'), 2]
[('noun.person', 'noun.time'), 1]
[('noun.person', 'noun.person'), 1]
[('noun.person', 'noun.communication'), 5]
[('noun.substance', 'noun.location'), 1]
[('noun.person', 'noun.act'), 1]
[('noun.substance', 'noun.person'), 1]
[('noun.quantity', 'noun.cognition'), 1]
[('noun.person', 'adv.all'), 1]
```

dove possiamo notare che per il verbo *read*, abbiamo che la coppia di cluster semantici piu' frequente e' quella relativa alla coppia *person:communication*, con una frequenza pari a 5.

4 Quinta Esercitazione

4.1 Introduzione

La quinta esercitazione prevede la sperimentazione del content-to-form, cioe' cercare di risalire al synset di un concetto indirizzando la ricerca in WordNet attraverso i genus. Alla base, vi e' il principio *Genus-Differentia definition*, secondo il quale un concetto puo' essere definito da due elementi principali :

- *Genus* : parte di una definizione esistente che viene utilizzata come porzione per una nuova definizione; tutte le definizioni con lo stesso genus vengono raggruppate secondo lo stesso genus
- *Differentia*: porzione della definizione che non viene dal genus e che rende piu' specifico un concetto.

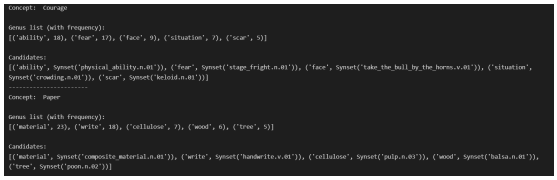
4.2 Sviluppo

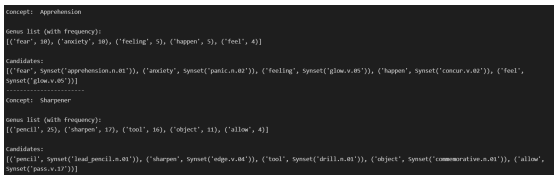
Sono state implementate le seguenti operazioni :

- Operazioni di Text Cleaning e preprocessing grazie al solito svoglimento di rimozione delle stopwords, rimozione della punteggiatura e tokenizzazione delle varie frasi.
- Una volta che sono state ottenute tutte le definizioni dei concetti, grazie al metodo *get_definitions(file)* svolto nell'esercitazione precedente, posso applicare il metodo di ricerca del genus. Prima di tutto, utilizzo il modulo *Counter* per ottenere un dizionario in cui le chiavi sono le parole che appartengono alla definizione e i valori rappresentano la frequenza di comparsa di quella parola. Impostiamo il valore numerico 5 come il numero di termini piu' importanti da estrarre per un concetto. La funzione mi restituira' una lista ordinata dei 5 termini piu' importanti (ovvero quelli piu frequenti)
- Per ogni genus, ottengo un possibile candidato; per ogni genere ottengo una lista di iponimi di tutti i synset relativi al termine genus. Verra' restituito quel synset che si avvicina di piu' alle definizioni del concetto; questo approccio permette di massimizzare l'overlap tra la signature dell'iperonimo e il BoW delle definizioni associate al concetto in analisi

4.3 Risultati Ottenuti

I risultati ottenuti dal task sono i seguenti :





Come possiamo vedere, solo in un unico caso troviamo una corrispondenza esatta tra il miglior senso presente nella lista genus e il senso del concetto in analisi. Parliamo proprio del concetto *Apprehension* e il suo genus *fear*: entrambi sono mappati al *Synset('apprehension.n.01'))*. Questo puo' essere dovuto da un legame semantico debole che c'e' tra il concetto preso in analisi e l'intorno ottenuto attravero i genus e i lori iperonimi. Inoltre, il documento *defs.csv* e' un file che e' stato rappresentato da noi studenti, quindi alcune definizioni possono essere lontane da cio' che WordNet intende per quel determinato concetto.

5 Sesta Esercitazione

La sesta esercitazione prevedeva la realizzazione di un semplice algoritmo di summarization. Il task consiste nell'effettuare un riassunto automatico a parte da un testo in input. Il riassunto viene creato estraendo parti di testo rilevanti (interi paragrafi nel testo) in base a un punteggio che viene assegnato a ciascun paragrafo a seconda delle parole rilevanti che ci sono all'interno di ciascun paragrafo. Il risultato ottenuto offre una compressione del testo del 40%.

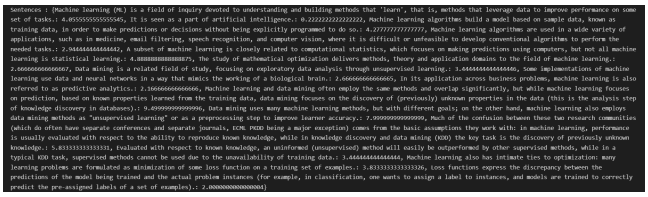
5.1 Sviluppo

Per l'implementazione del seguente task ho deciso di riportare in formato .txt una pagina di Wikipedia che trattava di Machine Learning. Sucessivamente sono state effettuate le seguenti implementazioni :

- Dato il testo, effettuo le solite operazioni di rimozione delle stopword, punteggiatura, oltre grazie al supporto fornito dalla libreria spacy che permette di effettuare le opportune operazioni di estrazione ed elaborazione dei testi analizzati.
- Successivamente, posso passare all'operazione di lettura del mio documento. Per fare cio' utilizzo il modulo glob che mi permette di recuperare file o percorsi corrispondenti ad un modello specificato.
- Una volta aperto il mio documento, conto le parole piu' frequenti all'interno del mio .txt: la frequenza delle parole mi sara' poi utile per fare la summarization finale.
- Per ciascun elenco di parole, vado a dividere ogni parola per il massimo valore di frequenza ottenuto. Questi punteggi espressi per ogni parola mi servono poi per capire se una determinata parola e' presente o no in un paragrafo e, se e' presente, probabilmente quel paragrafo avra' un rilevanza maggiore rispetto ad un altro.
- Posso creare dei punteggi per ogni frase: il punteggio fa riferimento a quanto l'algoritmo considera rilevante o no quel paragrafo in relazione al contesto su cui sta operando.
- Tramite il modulo *nlargest* prendo i paragrafi con i punteggi migliori, ci applico una compressione del 40% e ne restituisco il riassunto finale
- Applico una metrica di score denominata *rouge_score* per visualizzare i risultati.

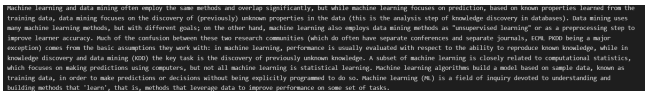
5.2 Risultati Ottenuti

Come abbiamo detto in precedenza, il risultato ottenuto e' stato definito a partire da una compressione del testo del 40%. Prima vediamo i punteggi associati ad ogni paragrafo :



Vediamo che alcune frasi hanno uno score molto elevato pari a 9.49, il che vuol dire che all'interno del paragrafo vi e' un maggior numero di parole chiave rilevanti per la summarization. Altri paragrafi hanno punteggi molto bassi come 0.22 perche' considerate prive di informazioni utili.

I risultati ottenuti portano alla seguente compressione:



Dove sono state riportate le frasi con lo score piu' alto.

Infine e' stata utilizzata una metrica Rouge per poter visualizzare gli effettivi punteggi ottenuti dal task di summarization. Tale metrica consiste in un insieme di metriche per valutare la sintesi automatica dei testi e le traduzioni automatiche. Generalmente funzioni confrontando un riassunto o una traduzione prodotta automaticamente con una serie di summaries (tipicamente prodotte dall'uomo). Le metriche coinvolte in questo processo di valutazione sono:

- Recall
- Precision
- F-1 score

Vogliamo ricordare che *Rouge-1*, *Rouge-2* e *Rouge-l* misurano l'overlap di unigrammi, bigrammi ed l-grammi presenti nel riassunto e nel documento originale. Come possiamo notare per la metrica Rogue, man mano che aumentiamo il tasso di compressione, subisce una variazione piu' o meno significativa. Poiche' e' la recall che misura il numero di parole sovrapposte in relazione al numero totali di parole presente nel documento originale, questa risulta essere la misura piu' indicativa della metrica.

6 Settima Esercitazione

6.1 Introduzione

La settima esercitazione prevede l'implementazione del *topic modelling*, ovvero creare un modello statistico in grado di determinare gli argomenti o topic di una collezione di documenti. L'esperimento e' stato svolto andando ad estrapolare

un corpus da Sketch Engine; l'idea era quella di andare ad estrapolare un corpus che potesse avere una struttura e una composizione organizzata in documenti e paragrafi.

6.2 Sviluppo

Per lo sviluppo sono state implementate le seguenti funzionalita' :

- Solite operazioni di Text Cleaning e preprocessing svolte come nelle esercitazioni del blocco 1.
- Operazione di lettura del corpus: dato il mio file, leggo ogni documento e ogni paragrafo. In output avre' una lista di documenti a cui viene associata una lista di parole pre-processate.
- data la lista di documenti ottenuti dalla funzione precedente, possiamo procedere con lo sviluppo del topic modelling: creo, mediante l'ausilio di *corpora*, un dizionario che contiene il mapping tra le parole e i loro ID interi. A questo dizionario rimuovo tutti i token che non appaiono in meno di 3 documenti e tutti quei token che non appaiono in almeno il 60% dei documenti. Viene poi creata un lista di BoW per ogni termine nel documento. Infine, con *LdaModel* effetuo il training del modello *LDA (Latent DIrichlet Allocation)* il cui training e' effettuato a partire da:
  - *corpus_freq* : lista di BoW calcolata precedentemente
  - *num_topics* : numero di topics che verranno estratti dal corpus
  - *id2word*: rappresenta il mio dizionario precedentemente ottenuto ed e' usato per determinare la dimensione del vocabolario e per la stampa del topic
  - *passes*: numero di volte che attraverso il corpus in fase di training
  - *alpha* : probabilita' a priori per ogni topic
  - *chunksize*: numero di documenti da caricare in memoria alla volta. Il valore di default per chunksize=2000.

6.3 Risultati Ottenuti

Il corpus travel contiene 100 documenti dove, grazie al modello LDA ottenuto, abbiamo ottenuto i seguenti topics:

Notiamo che ogni Topic e' descritto mediante una serie di termini che sono ordinati in base alla loro significativita' all'interno del topic stesso.

Vediamo poi i topics appartenenti ai primi 20 documenti:

```
Doc 0 : [(5, 0.98751235)]
Doc 1 : [(2, 0.13711767), (5, 0.8607147)]
Doc 2 : [(0, 0.5192168), (1, 0.31337816), (3, 0.017647326), (4, 0.13945827)]
Doc 3 : [(1, 0.06805676), (2, 0.39220524), (5, 0.44496346), (7, 0.091738485)]
Doc 4 : [(3, 0.9985373)]
Doc 5 : [(2, 0.01055066), (3, 0.8591707), (9, 0.12990437)]
Doc 6 : [(3, 0.99202764)]
Doc 7 : [(0, 0.22088102), (3, 0.33655864), (5, 0.43885994)]
Doc 8 : [(5, 0.033344958), (9, 0.96604043)]
Doc 9 : [(0, 0.025432905), (5, 0.6630382), (9, 0.30522987)]
Doc 10 : [(0, 0.9990297)]
Doc 11 : [(0, 0.6552332), (4, 0.34283528)]
Doc 12 : [(4, 0.99257296)]
Doc 13 : [(7, 0.99716973)]
Doc 14 : [(2, 0.19703244), (7, 0.7341375), (8, 0.06538537)]
Doc 15 : [(0, 0.07200433), (2, 0.021269456), (3, 0.90590125)]
Doc 16 : [(0, 0.500172), (3, 0.49519813)]
Doc 17 : [(0, 0.26614466), (5, 0.17361982), (9, 0.5588081)]
Doc 18 : [(0, 0.34682715), (3, 0.65098876)]
Doc 19 : [(0, 0.99726015)]
Doc 20 : [(0, 0.30688792), (3, 0.011119908), (9, 0.68103606)]
```