Progetto TLN

(Prof. Di Caro)

a.a. 2020/21

Silvestro Stefano Frisullo 832813

Aldo Bushaj 847091

Antonino Bushaj 847013

**Esercizio 1.1**

Dati i seguenti concetti:

* Coraggio (Generico / Astratto)
* Carta (Generico / Concreto)
* Inquietudine (Specifico / Astratto)
* Temperino (Specifico / Concreto)

La consegna richiede di calcolare le similarità tra le definizioni proposte dei vari concetti.

Innanzitutto abbiamo fatto un preprocessing dei dati rimuovendo le stop words, tokenizzando e lemmatizzando i termini.

Per calcolare la similarità abbiamo impiegato la seguente formula, dove def1 e def2 sono due definizioni:

| len(def1.intersection(def2)) / min(len(def1), len(def2)) |
| --- |

Ossia calcoliamo la cardinalità dell’intersezione tra le due definizioni e normalizziamo sulla lunghezza minima tra i due insiemi.

Otteniamo:

| Astratto - Generico: 0.2025916803503011 Astratto - Specifico: 0.12072796934865905 Concreto - Generico: 0.2811384783798581 Concreto - Specifico: 0.37766283524904154 |
| --- |

Possiamo concludere che le definizioni usate per descrivere il coraggio e inquietudine, in quanto concetti astratti, usino un vocabolario più ampio dei concetti concreti, che risultano avere un grado di similarità consistentemente più alto.  
Inoltre si evince come le definizioni siano sostanzialmente molto diverse fra di loro, e questo è giustificabile col fatto che le persone descrivono i concetti in modo molto diverso seppur questi siano parole condivise e comuni.

Confrontando con le definizioni di wordnet otteniamo:

| Con utilizzo di gloss e esempi di WordNet: Astratto - Generico: 0.2 Astratto - Specifico: 0.0 Concreto - Generico: 0.0625 Concreto - Specifico: 0.0 |
| --- |

Dal confronto con Wordnet notiamo come la similarità scende ancora di più, questo è giustificabile dal fatto che le definizioni di wordnet sono formulate da parlanti nativi che utilizzano un vocabolario più fine complesso di quello delle nostre definizioni.

**Esercizio 1.2**

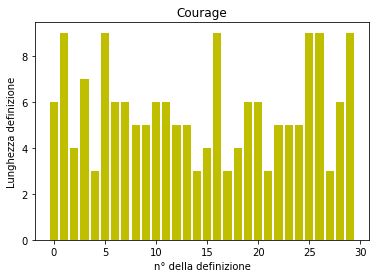
Per dare una spiegazione ai valori di similarità ottenuti nell’esercizio precedente stampiamo le prime 3 parole più usate per ogni concetto:

| ['ability', 'fear', 'face'] ['material', 'write', 'use'] ['fear', 'anxiety', 'feel'] ['pencil', 'sharpen', 'tool'] |
| --- |

**Esercizio 1.3**

L’esercizio chiede di dare una caratterizzazione delle definizioni in WordNet attraverso uno studio di forma e di relazione.  
Per studiare la forma abbiamo mostrato mediante grafici la lunghezza di ogni definizione di ogni concetto, mentre per lo studio delle relazioni abbiamo cercato gli iperonimi diretti e di primo livello dei concetti nelle loro definizioni.

Esempio di grafico della lunghezza delle definizioni di Courage



In output diamo la lista degli iperonimi diretti e di primo livello di ogni concetto trovati su wordnet:

| Courage: count:0 [] Paper: count:28 ['material', 'material', 'material', 'product', 'material', 'piece', 'material', 'material', 'material', 'material', 'material', 'material', 'product', 'material', 'material', 'material', 'material', 'medium', 'material', 'material', 'material', 'material', 'material', 'product', 'material', 'material', 'material', 'material'] Apprehension: count:13 ['expectation', 'fear', 'fear', 'fear', 'emotion', 'fear', 'fear', 'fear', 'fear', 'fear', 'fear', 'fear', 'emotion'] Sharpener: count:0 [] |
| --- |

Possiamo osservare come in alcuni casi gli iperonimi trovati siano zero, questo indica che o vi erano iperonimi di livelli ancora superiori oppure non erano in relazione di iperonimia. Inoltre il corpus presenta vari errori di scrittura e alcune frasi in italiano che vengono ignorate, quindi è comprensibile che con alcuni concetti non si trovino iperonimi.

**Esercizio 1.4**

L’esercizio chiede di scegliere un verbo transitivo con almeno 2 argomenti, recuperare un corpus n con più di 1000 istanze in cui viene usato, effettuare parsing e disambiguazione, usare i supersensi di Wordnet e aggregare i risultati in cluster semantici.

Il verbo che abbiamo scelto è *run*, che può contare su un ventaglio di significati diversi, uno dei più ampi della lingua inglese.

Il corpus di frasi è stato recuperato da Sketch Engine e conta 2700 frasi.  
Inoltre utilizziamo la libreria [SpaCy](https://spacy.io/) per l’analisi sintattica a dipendenze, in particolare per esplorare i child e i parent di ogni token per cercare soggetto e oggetto di *run*

if verb in token.text : #se il verbo cercato è nel testo del token

for child in token.children : #per ogni figlio sintattico del token

#se la relazione sintattica tra il token e il figlio è nsubj https://spacy.io/usage/linguistic-features vedere immagine in fondo

if child.dep\_ == "nsubj" and subj == "":

subj = child.text #se il padre è in relazione di soggetto e non l'ho ancora trovato lo aggiungo a subj

if child.dep\_ == "dobj" and obj == "" :

obj = child.text #se il figlio è in relazione di oggetto e non l'ho ancora trovato lo aggiungo a obj

In seguito con l’algoritmo di Lesk possiamo disambiguare il verbo e lo assegniamo al cluster corrispondente.

In output abbiamo i cluster con la coppia <soggetto, oggetto> : contatore di frequenza.

| {'<noun.substance,noun.act>': 21, '<noun.substance,noun.quantity>': 4, '<noun.substance,noun.group>': 23, '<noun.person,noun.group>': 37, '<noun.object,noun.communication>': 1, '<noun.act,noun.act>': 2, '<noun.group,noun.artifact>': 20, '<noun.group,noun.object>': 1, '<noun.group,noun.process>': 1, '<noun.substance,noun.artifact>': 26, '<adj.all,noun.group>': 2, '<noun.person,noun.person>': 10, '<noun.person,noun.state>': 2, '<noun.person,noun.act>': 7, '<noun.cognition,noun.communication>': 6, '<noun.group,noun.attribute>': 5, '<noun.quantity,noun.communication>': 4, '<noun.group,noun.time>': 1, '<adj.all,noun.person>': 1, '<noun.object,noun.state>': 1, '<noun.location,noun.artifact>': 4, '<noun.cognition,noun.person>': 1, '<adj.all,noun.cognition>': 1, '<noun.communication,noun.event>': 1, '<noun.group,noun.shape>': 4} ... |
| --- |

Naturalmente l’output è tagliato perché avrebbe occupato troppo spazio nella relazione.

Possiamo concludere , in base ai risultati ottenuti, che l’ambiguità d’uso del verbo *run* nella lingua inglese sia rispecchiata dall’ampio numero di cluster individuati.

**Esercizio 1.5**

L’esercizio prevede di risolvere un task *content-to-form* , ossia riconoscere i concetti a partire dalle loro definizioni.

I concetti da individuare sono:

* Coraggio (Generico / Astratto)
* Carta (Generico / Concreto)
* Inquietudine (Specifico / Astratto)
* Temperino (Specifico / Concreto)

Dopo aver svolto una fase di preprocessing, ordino le parole di ogni definizione in base alla loro frequenza.  
Successivamente prendiamo il primo nome in ogni definizione per cercare il genus e applichiamo l’algoritmo di Lesk per la disambiguazione per ricavare il synset corrispondente.

Adesso calcoliamo lo score dell’intersezione tra le parole della definizione i-esima del concetto e la signature del synset del nome come il numero di parole in comune.

#Calcolo tra i nomi analizzati dalla lista delle parole, il synset con valore migliore

# ossia conto il numero di parole comuni tra le parole che definiscono il concetto

# e la signature del synset, che metto in intersection (più alto è meglio)

intersection = overlap\_lists(concept\_words, get\_signature\_of\_synset(syn))

A questo punto aggiorniamo il best-val trovato (ossia quello che massimizza lo score dell’intersezione tra le parole delle nostre definizioni del concetto e la signature del synset

# Se il synset ha valore bag-of-words migliore, lo uso come genus

if best\_val <= intersection :

best\_val = overlap\_lists(concept\_words,get\_signature\_of\_synset(syn))

best\_sense = syn

Se un synset non ha uno score superiore a best\_val, lo salviamo in backtracking\_sense per poter fare backtracking in seguito.  
A questo punto cerchiamo tra gli iponimi del genus, rappresentato da best\_sense, un synset che migliori lo score dell’intersezione. Se non otteniamo un miglioramento proviamo a fare lo stesso con backtracking\_sense.  
Se nessuno dei due tentativi migliora lo score, lasciamo il synset che c’era all’inizio.

L’output è:

| Synset('ability.n.01') the quality of being able to perform; a quality that permits or facilitates achievement or accomplishment  Synset('writing\_paper.n.01') paper material made into thin sheets that are sized to take ink; used for writing correspondence and manuscripts  Synset('feel.v.01') undergo an emotional sensation or be in a particular state of mind  Synset('slate\_pencil.n.01') a pencil of soft slate (or soapstone) used for writing on a slate |
| --- |

Concludiamo quindi che nel caso di Paper siamo riusciti a inferire il concetto esatto, nel caso di temperino ci siamo avvicinati perchè abbiamo inferito il concetto Slate Pencil ( in quanto nelle definizioni di temperino viene nominata spesso la matita), mentre per courage e apprehension abbiamo ottenuto dei risultati molto generali e nello specifico caso di courage supponiamo l’algoritmo sia stato troppo influenzato dalla presenza della parola “ability” in numerose definizioni del concetto.

**Esercizio 2.1**

L’esercizio richiede di risolvere un task di text segmentation, quindi di identificare tutti i cambi di contesto che vengono fatti nel testo. Abbiamo suddiviso il testo in un numero di sezioni arbitrario di lunghezza uguale (ossia stesso numero di frasi), a cui successivamente abbiamo applicato il preprocessing e per ogni frase abbiamo memorizzato la frequenza delle parole.

Dopodiché per ogni frase ( a due a due) calcoliamo la misura di coesione rispetto alla successiva, per individuare il *breakpoint*, ossia il punto in cui si prevede vi sia il cambio di contesto ( fine sezione attuale e inizio nuova sezione).

La misura di coesione implementata determina l’overlap di parole tra le frasi ( cioè le parole in comune tra le due frasi) , successivamente sommiamo per ogni parola comune la media della frequenza della parola nella prima frase e nella seconda, e normalizziamo sulla dimensione dell’overlap.

# è una misura di similarità che tiene conto delle frequenze delle parole per ogni frase.

# calcola l'overlap delle due frasi.

# Per ogni parola dell'overlap somma la media delle frequenze della parole all'interno delle due frasi

# Normalizza la somma sulla lunghezza dell'intersezione

def similarity(s1, s2):

sim = 0

overlap = set.intersection(set(s1.keys()), set(s2.keys()))

if len(overlap) > 0:

for word in overlap:

sim += (s1[word] + s2[word]) / 2 # media fra le due frequenze

#qui si normalizza sulla lunghezza delle intersezioni

sim /= len(overlap)

return sim

Abbiamo testato il programma sull’argomento pasta, in particolare sul primo paragrafo dell’articolo di [Wikipedia.org versione inglese](https://en.wikipedia.org/wiki/Pasta), che gli autori dividono in 5 sezioni. I risultati sono abbastanza buoni, anche se non rispettano precisamente la divisione data dagli autori: ad esempio la prima frase viene considerata una sezione a sé stante, e questo è comprensibile visto che prova a dare una definizione enciclopedica della pasta.  
Output:

| Sezione 1 : 0 - 0 Sezione 2 : 1 - 5 Sezione 3 : 6 - 10 Sezione 4 : 11 - 12 Sezione 5 : 13 - 18 |
| --- |

**Esercizio 2.2**

L’esercizio chiede di sviluppare una soluzione di topic modeling partendo da un corpus a piacere e suggerendo di utilizzare la libreria Gensim.  
Abbiamo scelto di utilizzare un modello LDA( Latent Dirichlet Allocation) con approccio a Bag of Words.  
Il dataset da noi scelto per l’ampiezza dei topic trattati è reperibile su [kaggle](https://www.kaggle.com/therohk/million-headlines), di cui abbiamo utilizzato però solo le prime 10000 righe per ragioni di tempo.  
Innanzitutto facciamo preprocessing dei dati, in seguito da questi creiamo un dizionario Gensim da cui poi andremo a filtrare i token troppo comuni e troppo rari (che compaiono meno di 15 documenti, più del 50% del corpus ) e teniamo i primi 100000 più usati.

dictionary.filter\_extremes(no\_below=15, no\_above=0.5, keep\_n=100000)

A questo punto creiamo il corpus per il bag of words

bow\_corpus = [dictionary.doc2bow(doc) for doc in processed\_docs]

Successivamente addestriamo il modello LDA,scegliendo di identificare 5 topics, di fare 2 passate sul corpus e di usare 2 workers per sfruttare più core

lda\_model = gensim.models.LdaMulticore(bow\_corpus, num\_topics=5, id2word=dictionary, passes=2, workers=2)

Ora stampiamo tutti i topic individuati dal modello e le parole con peso maggiore all’interno di essi

| Topic: 0  Words: 0.028\*"iraq" + 0.018\*"crash" + 0.015\*"report" + 0.014\*"support" + 0.013\*"claim" + 0.013\*"kill" + 0.012\*"injur" + 0.011\*"dead" + 0.011\*"win" + 0.010\*"plan" Topic: 1  Words: 0.027\*"baghdad" + 0.024\*"govt" + 0.022\*"iraqi" + 0.019\*"council" + 0.016\*"urg" + 0.011\*"forc" + 0.010\*"iraq" + 0.010\*"suspect" + 0.010\*"claim" + 0.010\*"elect" Topic: 2  Words: 0.030\*"iraq" + 0.029\*"say" + 0.021\*"protest" + 0.019\*"plan" + 0.019\*"anti" + 0.012\*"concern" + 0.009\*"deni" + 0.008\*"forc" + 0.008\*"warn" + 0.007\*"help" Topic: 3  Words: 0.017\*"iraq" + 0.014\*"iraqi" + 0.012\*"warn" + 0.012\*"clash" + 0.011\*"miss" + 0.011\*"sar" + 0.011\*"water" + 0.011\*"murder" + 0.011\*"test" + 0.010\*"final" Topic: 4  Words: 0.034\*"polic" + 0.019\*"death" + 0.017\*"charg" + 0.015\*"face" + 0.012\*"court" + 0.011\*"saddam" + 0.010\*"council" + 0.010\*"troop" + 0.009\*"govt" + 0.009\*"report" |
| --- |

Infine testiamo il modello su un documento conosciuto e su di uno sconosciuto:

| #Documento Conosciuto Score:0.6399598 Topic: 0.034\*"polic" + 0.019\*"death" + 0.017\*"charg" + 0.015\*"face" + 0.012\*"court" + 0.011\*"saddam" + 0.010\*"council" + 0.010\*"troop" + 0.009\*"govt" + 0.009\*"report" |
| --- |

| #Documento Sconosciuto 'university kill health government guns'  Score: 0.5979681015014648 Topic: 0.021\*"say" + 0.020\*"kill" + 0.015\*"baghdad" + 0.015\*"iraqi" + 0.013\*"world" |
| --- |

Possiamo concludere che il modello, considerando anche il basso numero di frasi su cui è stato addestrato per ragioni di tempo, sia ragionevolmente sensibile e riesca a distinguere topic lessicalmente molto diversi tra loro.