Tecnologie del Linguaggio Naturale

Francesco Sannicola, Matteo Parisi

Prima-Seconda esercitazione

Introduzione

La prima e seconda esercitazione riguardano il calcolo della consistenza tra definizioni effettuate da noi studenti. Il task svolto a lezione prevedeva la selezione di 4 concetti, tra tutti quelli proposti dagli studenti, e di descrivere tali parole mediante definizioni. Ogni persona ha effettuato l'annotazione in modo indipendente cioè in base alla sua conoscenza del concetto.

Sono stati scelti 4 concetti di cui 2 generici e 2 specifici. Ogni concetto generici o specifici può essere a sua volta astratto o concreto:

- GENERICO ASTRATTO: **courage**
- GENERICO CONCRETO: **paper**
- SPECIFICO ASTRATTO: **apprehension**
- SPECIFICO CONCRETO: **sharpener**

Il file *definizioni.csv* contiene 30 definizioni per ogni concetto.

Sviluppo

- ***remove_stopwords(words_list)***: prende in input la frase ed elimina tutti le sstopwords contenute nel file *stop_words_FULL.txt*
- ***remove_punctuation(sentence)***: data una frase, verrà restituita una frase senza punteggiatura.
- ***tokenize_sentence(sentence)***: viene effettuato lo splitting della frase sfruttando gli spazi e, successivamente, ogni parola viene portata al suo lemma.
- ***get_signature(sense)***: attraverso l'uso dei metodi precedentemente descritti e delle API di Wordnet viene calcolata la **signature**, ovvero la concatenazione delle parole pre-processate delle definizioni e i termini provenienti dagli esempi in WN per un determinato senso.
- ***get_definitions(file)*** legge il file *definizioni.csv* e restituisce un dizionario dove la chiave è rappresentata dal termine e il valore da una lista di **BoW** correlata alle definizioni. Vengono utilizzati i metodi menzionati in precedenza.
- ***cosine_sim(def1, def2)*** : prende in input due insiemi di parole corrispondenti a due definizioni distinte, crea due vettori numerici, uno per *def1* e uno per *def2*, che conterranno 1 se la parola in analisi appartiene al set, 0 altrimenti. Ciò viene fatto per ogni parola che appare nelle due definizioni. Il calcolo della similarità del coseno viene effuata proprio tra questi due vettori numerici. La similarità è uguale al rapporto del prodotto scalare tra i vettori e il prodotto delle loro norme.

$$cosine(v_1, v_2) = \frac{v_1 \bullet v_2}{||v_1|| \cdot ||v_2||}$$

- ***compute_result(definitions_words)***: prende in input il dizionario di liste contenente le definizioni processate e calcola la similarità tra tutte le coppie di definizioni dello stesso concetto. Restituirà il valore medio delle similrità calcolate per ogni concetto.

- ***most_frequent_words(definitions)***: calcola le parole più frequenti all'interno delle definizioni di un concetto. In particolare, per ogni concetto, calcola quelle parole che appaiono almeno nel 50% delle definizioni.

Risultati

I risultati ottenuti sono in linea con quanto ci aspettavamo: i concetti più astratti sono descritti mediante delle definizioni poco simili. Al contrario i concetti concreti, *paper* e *sharpener*, vengono annotate attraverso definizioni più semplici e in linea tra di loro.

Nel dettaglio:

Concetto	Similarità definizioni
Courage	0.2105
Paper	0.2925
Apprehension	0.0830
Sharpener	0.3863

I risultati ottenuti dipendono principalmente dalla presenza di parole ripetute nelle varie definizioni. Il metodo *most_frequent_words()* ci fa notare la presenza di un maggior numero di parole ripetute tra le definizioni del concetto *Sharpener*, *Courage* e *Paper* condividono il numero di parole ripetute tra definizioni.

Concetto	Parole ripetute
Courage	ability, fear
Paper	write, material
Apprehension	
Sharpener	sharpen, pencil, tool

Terza esercitazione

Introduzione

Tra le tre proposte è stato scelto il task di **caratterizzazione delle definizione in Wordnet**. L'obiettivo è lo studio e ricerca di vari pattern all'interno dell'ontologia di Wordnet. Questi pattern fanno riferimento alle definizioni, in particolare della loro lunghezza, e alle relazioni tipiche di Wordnet (iperonimia e iponimia).

Gli studi effettuati sono 4:

1. Calcolo della lunghezza media delle definizioni appartenenti ad ogni categoria presente in Wordnet, ovvero nomi, verbi, aggettivi e avverbi.
2. Calcolo delle lunghezze delle definizioni di tutti gli iperonimi di un determinato concetto.
3. Calcolo distanza del concetto in analisi dal nodo radice e confronto con le distanze ottenute tra le parole più significative della definizione della parola in analisi e il nodo root più vicino.
4. Calcolo overlap tra la definizione del concetto in analisi e le definizioni di iperonimi ed iponimi. Viene fatta una media aritmetica degli score di similarità degli iperonimi e degli iponimi in modo da ottenere due valori da poter confrontare. Inizialmente era stato deciso di

utilizzare gli score Rogue e Bleu per questo tipo di task. Purtroppo, queste due metriche non sono adatte al calcolo della similarità (overlap) tra due frasi bensì per valutare riassunti e traduzioni automatiche. La versione definitiva prevede l'utilizzo di un modello basato sulle *reti transformer*: il testo sarà prima mappato in uno spazio vettoriale e poi sarà effettuata la similarità del coseno direttamente nel nuovo spazio (Link Github).

Sviluppo

- **remove_stopwords(words_list)**, **remove_punctuation(sentence)**, **tokenize_sentence(sentence)**, **get_signature(sense)**: implementazione analoga all'esercizio precedente.
- **avg_len_section_definitons()**: implementazione primo studio. Viene calcolata la lunghezza media di ogni definizione per sezione.
- **all_hypernym_paths(word)**: implementazione secondo studio. Data una parola viene considerato ogni suo significato e per ogni suo significato vengono calcolati i suoi iperonimi. Per ogni iperonimo fino al nodo radice vengono calcolate le lunghezze delle definizioni.
- **distance_root(word)** e **calculate_distance_root(synset)**: implementazione terzo studio. Richiede in input la parola e viene restituito un dizionario di dizionari contenente come chiavi i vari significati del concetto in analisi e come valore le parole associate alla definizione del concetto con la relativa distanza (minima) dalla radice.
- **definition_overlap(word)**: implementazione quarto studio. Viene calcolato uno score che indica quanto le definizioni di iperonimi e iponimi sono simili alla definizione del concetto in analisi. Sono state utilizzate il module *sentence_transformers* per il calcolo dei vettori embedded corrispondenti alle varie definizioni e la libreria *scipy.spatial* utilizzata per il calcolo della distanza del coseno. I vettori embedded vengono computati a partire da un modello basato sulla rete transformer BERT, pre-addestrato sulla lingua inglese e ottimizzato per vari task. I valori restituiti saranno due per ogni synset del concetto in analisi: essi corrispondono al valore medio di somiglianza tra le definizioni di tutti gli iperonimi/iponimi e la definizione della parola in analisi.

Risultati

- **Primo Studio**

Categoria	Lunghezza media definizioni
Nomi	11,47
Verbi	6,14
Aggettivi	7,23
Avverbi	5,02

Notiamo che i nomi vengono descritti utilizzando un quantitativo maggiore di parole. A seguire i verbi, aggettivi e avverbi.

- **Secondo studio**

A seguire il risultato ottenuto per un solo senso e per ogni parola in analisi.

```
Concept:  Courage
[(Synset('entity.n.01'), 17), (Synset('abstraction.n.06'), 11),
(Synset('attribute.n.02'), 9), (Synset('trait.n.01'), 7),
(Synset('character.n.03'), 18), (Synset('spirit.n.03'), 9),
(Synset('courage.n.01'), 15)] # a partire dal synset 'courage.n.01'

Concept:  Paper
[(Synset('entity.n.01'), 17), (Synset('physical_entity.n.01'), 6),
(Synset('matter.n.03'), 7), (Synset('substance.n.01'), 11),
(Synset('material.n.01'), 12), (Synset('paper.n.01'), 15)] # a partire dal synset
'paper.n.01', ci sono altri 8 sensi per Paper

Concept:  Apprehension
[(Synset('entity.n.01'), 17), (Synset('abstraction.n.06'), 11),
(Synset('attribute.n.02'), 9), (Synset('state.n.02'), 10),
(Synset('feeling.n.01'), 7), (Synset('emotion.n.01'), 3), (Synset('fear.n.01'),
20), (Synset('apprehension.n.01'), 4)] # a partire dal synset
'apprehension.n.01', ci sono altri 3 sensi per Apprehension

Concept:  Sharpener
[(Synset('entity.n.01'), 17), (Synset('physical_entity.n.01'), 6),
(Synset('object.n.01'), 12), (Synset('whole.n.02'), 11),
(Synset('artifact.n.01'), 7), (Synset('instrumentality.n.03'), 13),
(Synset('implement.n.01'), 12), (Synset('sharpener.n.01'), 14)] # a partire dal
synset 'sharpener.n.01'
```

- **Terzo studio**

  Viene stampato per ogni senso del concetto in analisi, la distanza minima tra le parole più significative della definizione e il nodo radice.

```
Concept:  Courage
{Synset('courage.n.01'): {'Courage': 7, 'quality': 1, 'spirit': 5, 'enable': 2,
'face': 1, 'danger': 4, 'pain': 4, 'fear': 1}}
# Courage : 7 vuol dire che il Synset('courage.n.01') ha distanza dal più vicino
nodo radice pari a 7.

Concept:  Paper
{Synset('paper.n.01'): {'Paper': 6, 'material': 1, 'cellulose': 9, 'pulp': 3,
'derive': 1, 'wood': 6, 'rag': 2, 'grass': 2}, ...

 Concept:  Apprehension
{Synset('apprehension.n.01'): {'Apprehension': 8, 'fearful': 1, 'expectation': 6,
'anticipation': 7}, ...

 Concept:  Sharpener
{Synset('sharpener.n.01'): {'Sharpener': 8, 'implement': 2, 'edge': 2, 'point':
2, 'sharper': 1}}
```

- **Quarto studio**

  Anche in questo caso consideriamo i risultati ottenuti su un singolo synset associato alla parola in analisi. Per una completa visione della soluzione rimandiamo al notebook.

Concetto	Synset considerato	Similarità media iperonimi	Similarità media iponimi
Courage	Synset('courage.n.01')	0.5595	0.6555
Paper	Synset('paper.n.01')	0.4182	0.544
Apprehension	Synset('apprehension.n.01')	0.7708	0.8047
Sharpener	Synset('sharpener.n.01')	0.733	0.6331

Tre risposte su quattro mostrano che le definizioni degli iponimi siano mediamente più vicine alla definizione del concetto in analisi. Questo vale anche per i sensi non mostrati nella tabella precedente. Importante sottolineare che la metà dei sensi, in questo caso di studio, non ha iponimi, quindi, non è stato possibile calcolare un valore di similarità.

Quarta esercitazione

Introduzione

La quarta esercitazione prevede lo studio di alcuni verbi dal punto di vista della **Teoria di Hanks**. Secondo Hanks il verbo è la radice del significato e non esistono espressioni senza verbo. Ad ogni verbo viene associata una valenza che indica il numero di argomenti necessari per il verbo. Possiamo differenziare il significato del verbo in base al numero di argomenti che possiede. Determinati il numero di argomenti di un verbo, bisogna specificarli mediante un certo numero di slot. Ogni slot può avere un certo numero di valori che lo riempiono, detti **filler**. Ogni filler può avere associati dei **tipi semantici** che rappresentano delle generalizzazioni concettuali strutturate come una gerarchia. Raggruppiamo i vari filler secondo alcuni types. Essi sono **gruppi semantici** generici o categorizzazioni/clusterizzazioni dei filler.

E' stato scelto di effettuare gli esperimenti su due verbi in particolare:

1. **eat**
2. **buy**

Inoltre sono state considerate solo le forme del verbo al presente, quindi: eat, eats, buy, buys.

I corpus sono stati estratti da *Sketch Engine* tramite il tool *Concordance*. Entrambi contengono circa 10000 frasi in inglese composte dal verbo in questione. Ogni frase descrive un contesto generico.

Sviluppo

- **remove_stopwords(words_list)**, **remove_punctuation(sentence)**, **tokenize_sentence(sentence)**, **get_signature(sense)**: implementazione analoga all'esercizio precedente.
- **search_obj(sentence, head_verb, pattern)**: data una frase elaborata come oggetto *spacy*, il verbo e un numero minore di 3 argomenti all'interno del pattern, viene calcolato l'oggetto della frase in base alla sua costruzione sintattica a dipendenze.

  Un esempio di pattern:

```
[('You', 'PRP', 'buy', 'nsubj'), ('products', 'NNS', 'buy', 'dobj'),
You can buy our products by PAYPAL Or Credit Card.
 ]
```

- **search_subj(sentence, head_verb, pattern)**: funzione analoga a quella precedente, ma per il calcolo del soggetto.
- **disambiguate_terms(pattern)** : una volta ottenuti i pattern all'interno del corpus si effettua la disambiguazione, ovvero viene associato un senso di Wordnet alle dipendenze di ogni pattern, attraverso l'algoritmo **Lesk**.
- **semantic_clusters(patterns)**: dato in input una lista di pattern vengono calcolati i cluster semantici. Viene utilizzata la funzione precedentemente descritta per l'assegnazione del senso alle due dipendenze. Attraverso l'attributo *lexname* otteniamo il supersenso associato ai due argomenti.

  Il calcolo dei cluster viene effettuato mediante l'ausilio del modulo *Counter* che crea un dizionario le cui chiavi corrispondono alle coppie *supersense1, supersense2* e il valore al numero di volte che appare quella coppia all'interno della lista di pattern. Una volta fatto ciò è semplice attribuire una percentuale di appartenenza ai cluster calcolati implicitamente con *Counter*. Infine viene restituita una struttura ordinata in base a tale percentuale.
- **find_patterns()**: in seguito all'inizializzazione di *spacy* e al parsing del file XML del corpus tramite *minidom*, vengono invocate le funzioni per il calcolo dell'oggetto e del soggetto di ogni frase. Vengono salvati esclusivamente i pattern contenenti esattamente due argomenti.

Risultati

Per entrambi i corpus (eat e buy) sono stati trovati circa 200 cluster semantici.

Nelle due prossime tabelle verranno mostrati i 10 cluster più frequenti per ogni verbo in analisi.

- **Buy**: 2709 pattern in 167 cluster

Cluster	Percentuale di appartenenza
('noun.quantity', 'noun.artifact')	11.43%
('noun.quantity', 'noun.cognition')	7.12%
('noun.group', 'noun.artifact')	4.71%
('noun.quantity', 'noun.communication')	3.71%
('noun.person', 'noun.artifact')	3.71%
('noun.act', 'noun.substance')	2.81%
('noun.group', 'noun.communication')	2.31%
('noun.substance', 'noun.artifact')	2.31%
('noun.quantity', 'noun.person')	2.11%
('noun.quantity', 'noun.act')	2.01%

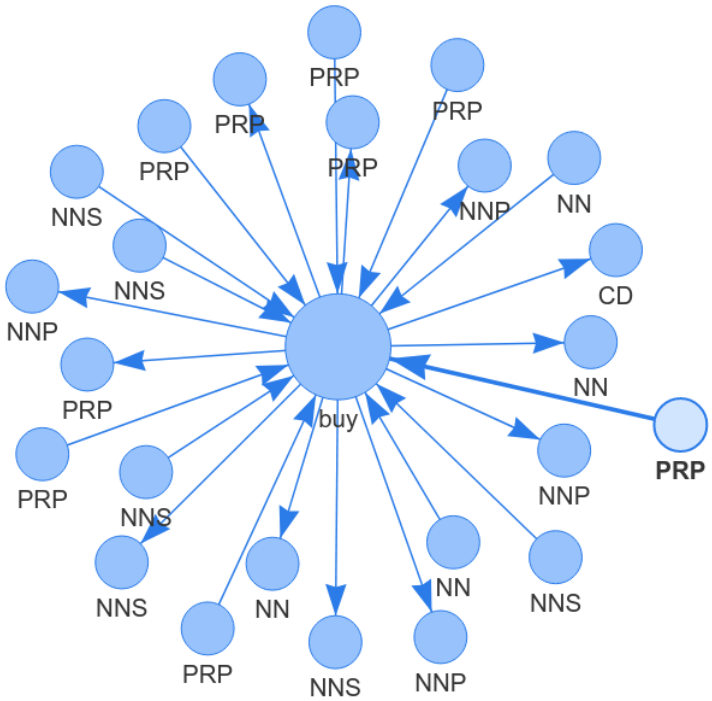
- **Eat**: 2902 pattern in 202 cluster

Cluster	Percentuale di appartenenza
('noun.quantity', 'noun.food')	8.32%
('noun.quantity', 'noun.cognition')	5.52%
('noun.group', 'noun.food')	4.84%
('noun.person', 'noun.food')	4.16%
('noun.quantity', 'noun.artifact')	2.72%
('noun.substance', 'noun.food')	2.72%
('noun.quantity', 'noun.quantity')	2.29%
('noun.group', 'noun.cognition')	2.04%
('noun.person', 'noun.cognition')	1.61%
('noun.group', 'noun.act')	1.53%

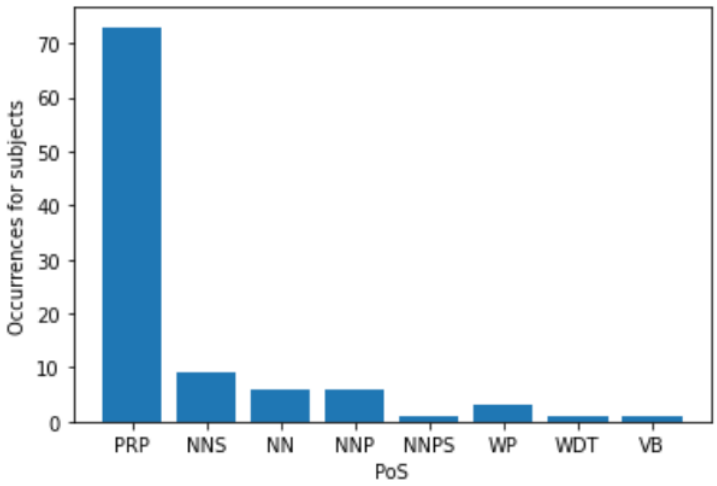
Task extra

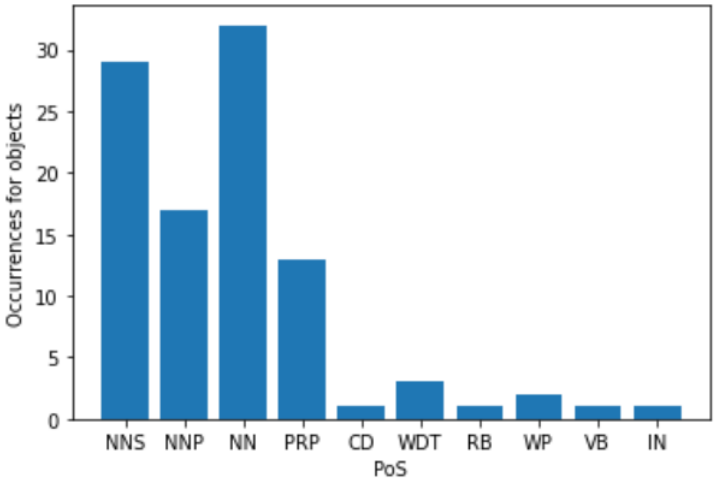
A partire dallo stesso corpus di dati testuali su cui è stato testato il principio di Hanks, è stato creato un Knowledge Graph utile alla visualizzazione di alcune informazioni contenute nel testo. Calcolati i pattern del tipo *(soggetto, oggetto, frase)*, visti prima, tramite *spacy* è stato possibile estrarre il loro part of speech. Fatto questo si è potuto procedere alla creazione del KG tramite il modulo *pyvis*: la rete risultate ha come nodi tutti i PoS trovati sia per i soggetti sia per gli oggetti. Tutti questi nodi *dependents* sono in relazione con un unico nodo che rappresenta il verbo (buy o eat). Gli archi sono etichettati con la tipologia di relazione: nel nostro caso la relazione *verbo-soggetto* e *verbo-oggetto*.

L'immagine mostra nodi e archi del KG riguardanti il PoS di 25 frasi del corpus *buy*. Gli archi uscenti dai nodi che non sono il verbo corrispondono al soggetto, i nodi con soli archi entranti rappresentano gli oggetti.



Ottenuto il grafo di conoscenza è stato possibile calcolare delle statistiche utili in fase di disambiguazione. In particolare sono state contate le occorrenze di ogni tag per soggetti ed oggetti. Le immagini sottostanti mostrano una distribuzione calcolata a partire da 250 frasi.





I due grafici mostrano la frequenza dei PoS per soggetti (primo grafico) e oggetti (secondo grafico). Dal primo si evince come, stopwords escluse, i PoS che occorrono più frequentemente nel ruolo di *subject* sono principalmente i pronomi, ma troviamo anche nomi (al singolare o non), aggettivi e verbi (entrambi in forma sostantivata). Pronomi non così frequenti, invece, nel ruolo di oggetti, notiamo un maggior numero di *NN*, di *NNS* e *NNP*. Da evidenziare anche la presenza di valori numerici (*CD*) e avverbi (*RB*).

Quinta esercitazione

Introduzione

La quinta esercitazione prevede la sperimentazione del content-to-form, cioè cercare di risalire al synset di un concetto indirizzando la ricerca in WordNet attraverso i **genus** dello stesso e usare approcci di overlapping delle parole per sfruttare il meccanismo di **differentia**.

Secondo il principio **Genus-Differentia definition** un concetto può essere descritto secondo due elementi principali:

- Genus: una definizione esistente che viene utilizzata come porzione di una nuova definizione; tutte le definizioni con lo stesso genus sono considerate membre di quel genus.
- Differentia: la porzione di definizione che non viene data dal genus e che rende più specifico e caratterizzato un concetto.

Sviluppo

- **remove_stopwords(words_list)**, **remove_punctuation(sentence)**, **tokenize_sentence(sentence)**, **get_signature(sense)**: implementazione analoga all'esercizio precedente.
- **get_definitions(file)**: stessa implementazione del primo esercizio: legge un file *.csv* e restituisce un dizionario con le definizioni per ogni concetto in analisi.
- **get_genus_list(definitions_word)**: ottenute le definizioni dei concetti, utilizziamo ancora il modulo *Counter* per ottenere un dizionario le cui chiavi sono le parole appartenenti alla defininizione e i valori rappresentano la frequenza di comparsa di quella parola. Utilizzeremo come *genus* le 5 parole che appaiono più frequentemente dato che, grazie a loro, è possibile creare un *intorno semantico*.
- **get_candidates(genus_list, definitions_word)**: per ogni genus ottengo il suo miglior iperonimo, ovvero quello che va a massimizzare l'overlap tra la *signature* (gloss + esempi) dell'iperonimo e il BoW delle definizioni associate al concetto in analisi.

Risultati

```
----------------------
Concept:  Courage

Genus list (with frequency):
[('ability', 18), ('fear', 17), ('face', 9), ('situation', 7), ('scar', 5)]

Candidates:
[('ability', Synset('physical_ability.n.01')), ('fear', Synset('stage_fright.n.01')),
('face', Synset('take_the_bull_by_the_horns.v.01')), ('situation',
Synset('crowding.n.01')), ('scar', Synset('keloid.n.01'))]
----------------------
Concept:  Paper

Genus list (with frequency):
[('material', 23), ('write', 18), ('cellulose', 7), ('wood', 6), ('tree', 5)]

Candidates:
[('material', Synset('composite_material.n.01')), ('write', Synset('handwrite.v.01')),
('cellulose', Synset('pulp.n.03')), ('wood', Synset('balsa.n.01')), ('tree',
Synset('poon.n.02'))]
----------------------
Concept:  Apprehension

Genus list (with frequency):
[('fear', 10), ('anxiety', 10), ('feeling', 5), ('happen', 5), ('feel', 4)]

Candidates:
[('fear', Synset('apprehension.n.01')), ('anxiety', Synset('panic.n.02')), ('feeling',
Synset('glow.v.05')), ('happen', Synset('concur.v.02')), ('feel',
Synset('glow.v.05'))]
----------------------
Concept:  Sharpener

Genus list (with frequency):
[('pencil', 25), ('sharpen', 17), ('tool', 16), ('object', 11), ('allow', 4)]
```

```
Candidates:
[('pencil', Synset('lead_pencil.n.01')), ('sharpen', Synset('edge.v.04')), ('tool',
Synset('drill.n.01')), ('object', Synset('commemorative.n.01')), ('allow',
Synset('pass.v.17'))]
```

Possiamo notare che solo in un unico caso troviamo una corrispondenza esatta tra il miglior senso attribuito al *genus* e un senso del concetto in analisi. Parliamo di *Apprehension* e il suo genus *fear*: entrambi sono mappati al *Synset('apprehension.n.01')*. Questo è dovuto al debole legame semantico che si va a creare tra il concetto in analisi e l'intorno ottenuto attraverso i genus e i loro iperonimi. Inoltre le definizioni date da noi studenti risultano meno pragmatiche e precise delle signature presenti in Wordnet.

Sesto esercizio

Introduzione

La sesta esercitazione prevede la realizzazione del task di **summarization**. Il task consiste nell'effettuare un riassunto automatico a partire da un testo in input. La risorsa utilizzata per svolgere il riassunto è NASARI in formato embedded, ovvero una rappresentazione vettoriale di synset Babelnet. L'approccio utilizzato è detto *estrattivo* e di tipo *statistico*: il riassunto viene creato estraendo parti di testo rilevanti (interi paragrafi o frasi) in base al tasso di compressione utilizzato. Il riassunto ottenuto sarà *indicativo*, ossia fornirà un'idea sul contenuto e *informativo*, ovvero conterrà tutti gli elementi rilevanti del documento iniziale. Il titolo del documento rappresenta un modo semplice ed efficace per capire di cosa tratterà il documento. A partire da esso saranno calcolati dei topics che ci consentiranno di scartare quei paragrafi che sono distanti semanticamente distanti da essi.

Sviluppo

- **read_nasari(file)**: richiede in input il path al file NASARI contenente i vettori embedded e restituisce in output un dizionario del tipo {babel_id: {term:score}...}.
- **read_doc(file)**: legge il documento da riassumere. Viene restituita una lista contenente l'insieme dei paragrafi.
- **calculate_rank(vector, nasari_vector)**: calcola il rank di uno specifico vettore. Nel nostro caso verrà restituito la posizione che ha il vettore all'interno del vettore di nasari.
- **weighted_overlap(nasari_vector_1, nasari_vector_2)**: Implementazione della formula di *Weighted Overlap* tra due vettori di nasari. Maggiore è l'overlap è più simili saranno i vettori.

$$WO(v_1, v_2) = \frac{\sum_{q \in O}(rank(q, v_1) + rank(q, v_2))^{-1}}{\sum_{i=1}^{|O|}(2i)^{-1}}$$

  Dove *O* è l'insieme di dimensioni in comune tra i due vettori.
- **bag_of_word_approach(text)**: metodo che effettua la rimozione delle stop-words e della punteggiatura. I termini rimanenti vengono ridotti nel loro lemma.
- **get_topic_from_title(document, nasari)**: vengono calcolati i topic di un documento considerando le parole più significative del titolo. In output avremo la rappresentazione vettoriale di questi termini.
- **text_to_nasari(text, nasari)**: calcola una rappresentazione vettoriale delle BoW dei termini di ciascun paragrafo del testo.
- **calculate_lines_to_keep(doc_paragraphs, percentage)**: dati i paragrafi di un documento e la percentuale di riduzione viene computato il numero di paragrafi da tenere.

- **reduce_document(doc_paragraphs_overlaps, lines_to_keep)**: il primo parametro è una struttura contenente l'ID del paragrafo, l'overlap medio tra i suoi termini e i topic (entrambi in formato vettoriale) e il suo testo. Una volta ordinati i paragrafi in base allo score, si crea una nuova struttura solo con i primi *n=lines_to_keep* paragrafi. Infine viene ristabilito l'ordine iniziale grazie all'ID di tipo auto-increment.
- **summarization(document, nasari, percentage)**: metodo principale che necessita in input del documento composto da titolo e paragrafi, il vettore Nasari e la percentuale di riduzione. Esso utilizza i metodi precedentemente descriitti. Possiamo riassumere il suo lavoro in:
    1. Calcolo dei topic utilizzando il titolo
    2. Per ogni paragrafo
        1. Ottiene la sua rappresentazione vettoriale
        2. Per ogni parola del paragrafo (rappresentata in forma vettoriale)
        3. Per ogni topic
            1. Calcola l'overlap tra il topic e la parola
            2. Somma tutti gli score ottenuti
        4. Calcola la media degli score della parola per ogni topic
        5. Somma le medie degli score di ogni parola ottenendo un punteggio cumulativo per il paragrafo
        6. Divide la somma cumulativa acquisita al passo precedente per il numero di paragrafi in modo da ottenere uno score medio per quel paragrafo
    3. Ricava il numero di paragrafi da tenere in base alla percentuale di riduzione e alla grandezza del documento
    4. Calcola e restituisce il documento ridotto
- **Nell'ultimo blocco** del notebook avviene l'analisi dei risultati ottenuti dall'algoritmo. Per il valutazione delle performance sono stati utilizzate due metriche distinte: **Bleu** e **Rogue**.

  Il BLEU originalmente veniva utilizzato per misurare la qualità di una traduzione fatta da una macchina confrontandola con quello che avrebbe fatto un essere umano esperto. Viene oggi utilizzata principalmente nella valutazione di riassunti automatici. Nel nostro caso calcola quanti 1-gram sono sopravvissuti applicando la summarization al documento originale (**precision**).
  La metrica del Rogue si basa sullo stesso principio del BLEU, ovvero calcola quanto la "traduzione" della macchina sia simile alla traduzione da parte di un agente umano. A differenza di BLEU controlla quanti 1-gram (2-grams e l-grams) contenuti nel documento creato dall'esperto sono presenti anche nel documento generato (**recall**).

Risultati

A seguire l'output restituito dall'esecuzione dell'algoritmo sul documento intitolato *Andy Warhol: Why the great Pop artist thought 'Trump is sort of cheap'*. Andy Warhol fu una figura predominante del movimento della Pop art nel XX secolo, oltre che ad un artista e produttore televisivo e cinematografico. Sono stati effetuati dei test anche su altri documenti i quali mostrato risultati paragonabili.

```
Andy-Warhol     Original lenght: 20


10 % redution   Summary lenght: 18
```

```
BLEU score:  0.8948393168143697
Rogue scores:  [{'rouge-1': {'f': 0.9290555756849036, 'p': 1.0, 'r':
0.8675105485232067}, 'rouge-2': {'f': 0.9136137444598561, 'p': 0.9834469328140214,
'r': 0.8530405405405406}, 'rouge-l': {'f': 0.9369369319568218, 'p': 1.0, 'r':
0.8813559322033898}}]


20 % redution   Summary lenght: 16

BLEU score:  0.7788007830714049
Rogue scores:  [{'rouge-1': {'f': 0.8523002372398267, 'p': 1.0, 'r':
0.7426160337552743}, 'rouge-2': {'f': 0.8376151187156861, 'p': 0.9829351535836177,
'r': 0.7297297297297297}, 'rouge-l': {'f': 0.8771626248332306, 'p': 1.0, 'r':
0.7812018489984591}}]


30 % redution   Summary lenght: 14

BLEU score:  0.6514390575310556
Rogue scores:  [{'rouge-1': {'f': 0.8030302982242884, 'p': 1.0, 'r':
0.6708860759493671}, 'rouge-2': {'f': 0.7896865472671786, 'p': 0.9836272040302267,
'r': 0.6596283783783784}, 'rouge-l': {'f': 0.8400357413299089, 'p': 1.0, 'r':
0.724191063174114}}]
```

Come prevedibile, man mano aumentiamo il tasso di compressione entrambe le metriche subiscono una riduzione più o meno significativa. Il Rogue sembra risentirne meno, soprattutto se consideriamo la precisione. Quest'ultimo è un dato fuorviante dato che va a calcolare quanti unigrammi, bigrammi e l-grams del riassunto sono presenti anche nel documento originale (tutti o quasi). La recall, invece, misura quanti unigrammi, bigrammi e l-grams contenuti nel documento originale sono presenti anche nel riassunto. La recall, quindi, risulta essere il dato più indicativo.

Settimo esercizio

Introduzione

Lo sesta esercitazione prevede l'implementazione del **topic modelling**, ovvero creare un modello statistico in grado di determinare gli argomenti o topic da una collezione di documenti. Non ci limiteremo a individuare il topic principale, ma cercheremo anche dei sotto-argomenti più specifici e distinti da quello principale.

Gli esperimenti sono stati condotti tramite alcuni corpus estrapolati da *Sketch Engine*. Gli unici corpus disponibili gratuitamente al download e che presentano una struttura e una composizione organizzata in documenti e paragrafi sono 3: *travelling*, *italian_cuisine*, *future_tenses*. Quest'ultimo contiene solo 10 documenti e non consente la creazione di un modello accurato. In questa analisi tratteremo il corpus *travelling*.

Sviluppo

- **remove_stopwords(words_list)**, **remove_punctuation(sentence)**, **tokenize_sentence(sentence)**, **get_signature(sense)**: implementazione analoga ad esercizio precedente.
- **read_corpus(txt_file)**: dato il path del file contenente il corpus, viene letto ogni documento e ogni paragrafo. In output avremo, quindi, una lista di documenti a cui viene associata una lista di parole appartententi ad ognuno di essi.

- **topic_modelling(documents_words)**: data la lista di liste ottenute dalla funzione precedente, viene prima creato, con l'ausilio di *corpora*, un dizionario a cui viene associata una chiave intera ad ogni parola e successivamente vengono scartati quei token che non appaiono in meno di 3 e in più del 60% dei documenti (tramite *filter_extremes(no_below=5, no_above=0.6*). Viene creata una lista di BoW del tipo *(id_term, term_frequency)* per ogni termine nei documenti con il metodo *doc2bow*.

  Infine, la funzione *LdaModel* effettua il training del modello LDA (Latent Dirichlet Allocation). Le BoW vengono mappate in un nuovo spazio di dimensione minore. Possiamo vedere i topic del modello LDA come una distribuzione di probabilità sulle parole.

  Il training è effettuato a partire da:

  1. *corpus_idbow_freq*: la lista di BoW calcolata precedentemente
  2. *num_topics*: numero di topic che verranno estratti dal corpus (nel nostro caso 10)
  3. *id2word*: dizionario di (int, str) ottenuto con *corpora.Dictionary* e *filter_extremes* (primi due passi menzionati prima). Il dizionario verrà usato per determinare la dimensione del vocabolario e per la stampa dei topic.
  4. *passes*: numero di volte che attraverso il corpus in fase di training.
  5. *alpha*: probabilità a priori di ogni topic (nel nostro caso sono equiprobabili).
  6. *eta*: probabilità a priori di ogni parola (ogni termine è equiprobabile).

Risultati

Il corpus *travelling* contiene esattamente 100 documenti. Il modello LDA ottenuto a partire dai documenti restituisce 10 topic di questo tipo:

```
Topic 0 : [('clause', 0.045), ('example', 0.02), ('third', 0.02), ('perfect', 0.016), ('conditionals', 0.014)]
Topic 1 : [('travel', 0.017), ('money', 0.014), ('lot', 0.011), ('holiday', 0.011), ('dream', 0.01)]
Topic 2 : [('clause', 0.026), ('situation', 0.015), ('condition', 0.015), ('result', 0.015), ('happen', 0.015)]
Topic 3 : [('students', 0.065), ('exam', 0.035), ('book', 0.026), ('sb', 0.026), ('speak', 0.025)]
Topic 4 : [('word', 0.023), ('clause', 0.021), ('happen', 0.014), ('example', 0.013), ('noun', 0.013)]
Topic 5 : [('level', 0.02), ('teach', 0.016), ('video', 0.015), ('love', 0.015), ('esl', 0.015)]
Topic 6 : [('hotel', 0.038), ('book', 0.035), ('holiday', 0.025), ('beach', 0.023), ('travel', 0.016)]
Topic 7 : [('lesson', 0.038), ('student', 0.012), ('grammar', 0.01), ('learn', 0.01), ('language', 0.009)]
Topic 8 : [('clause', 0.015), ('condition', 0.013), ('happen', 0.012), ('result', 0.012), ('book', 0.01)]
Topic 9 : [('travel', 0.031), ('article', 0.026), ('word', 0.021), ('journey', 0.02), ('example', 0.017)]
```

Notiamo che ogni topic è descritto mediante 5 termini ordinati in base alla loro significatività all'interno del topic.

Infine, vengono mostrati i topics appartenenti ai primi 10 documenti (per una visione completa dei risultati vedere il *notebook*).

```
Doc 0 : [(1, 0.8516115), (9, 0.14705722)]
Doc 1 : [(1, 0.1281324), (3, 0.2067905), (6, 0.6648527)]
Doc 2 : [(1, 0.43143946), (7, 0.5667039)]
Doc 3 : [(0, 0.7186929), (3, 0.101707526), (7, 0.05541384), (9, 0.12384491)]
Doc 4 : [(7, 0.99982)]
Doc 5 : [(3, 0.08431996), (4, 0.04471661), (7, 0.46974307), (9, 0.40117958)]
Doc 6 : [(7, 0.99901193)]
Doc 7 : [(0, 0.7027754), (2, 0.11124558), (7, 0.18546453)]
Doc 8 : [(1, 0.99991614)]
Doc 9 : [(1, 0.999696)]
Doc 10 : [(2, 0.73135585), (4, 0.26853248)]
```