Esercitazione 1 : Lesk Algorithm con WordNet

Marco Corona

Luglio 2017

Indice

1 Introduzione

Lo scopo dell'esercizio risulta quello di implementare l'algoritmo di Lesk per il word disambiguation problem. L'algoritmo sceglie il miglior senso per una parola target all'interno di una frase, calcolando per ogni senso possibile un valore di overlap tra il *contesto* (ovvero l'elenco di termini che compaiono nella frase) e la *firma* del senso, dove per firma intendiamo l'insieme di termini che compaiono nella glossa che descrive l'uso di quella parola quando viene utilizzata con quel significato (descrizione ed esempi di utilizzo).
Il valore di overlap rappresenta la cardinalità dell'intersezione tra questi due insiemi, dove ogni parola però può essere pesata a seconda dell'influenza che ricopre. La pesatura avviene tramite l'utilizzo di una IDF (Inverse document function) nel seguente modo: sia G con $|G| = n$ l'insieme di tutte le frasi che compongono le glosse di una certa parola target, se un certo termine i compare in k frasi diverse con $k \leq n$ allora ottieniamo il valore :

$$idf(i) = log\left(\frac{n}{k}\right) = log(n) - log(k) \tag{1}$$

Presi l'insieme C dei termini del contesto e S dei termini della firma associata ad un significato, otteniamo l'overlap calcolando :

$$overlap(C, S) = \sum_{i \in C \wedge S} idf(i) \tag{2}$$

L'algoritmo di lesk selezione in miglior senso S* per la parola w come :

$$S^* = \arg\max_S overlap(C, S) \tag{3}$$

2 Costruzione del dizionario e lemmatizzazione

Per poter eseguire il lesk è stata implementata un struttura dati dizionario la quale contiene le features indicizzate tramite la stringa del termine che rappresentano.
Una volta che è stata richiesta la parola target da disambiguare per prima cosa vengono prelevati tutti i possibili sensi di quella parola sulla base del suo ruolo sintattico, ovvero si determina la part-of-speech che occupa la parola nella frase (utilizzando la funzione di POS tagging della libreria Stanford NLP).
Tutti i sensi della parola sono reperiti utilizzando la risorsa WordNet la quale contiene al suo interno oggetti detti Synset i quali rappresentano uno dei possibili sensi con cui una determinata parola può essere intesa. All'interno di questi synset recuperiamo la definizione e gli esempi di utilizzo della parola.
Una volta recuperato, queste frasi vengono poi annotate, sempre tramite Stanford NLP, associando ad ogni termine il proprio lemma.
I lemmi diventano poi features aggiungendo alcune informazioni quali il numero di testi in cui esso compare. Per rendere più semplice il calcolo dell'overlap, ad ogni synset posto in esame, viene associato, tramite una mappa, l'insieme dei lemmi che sono presenti all'interno della firma.

3 Filtraggio delle stop words

Per tentare di rimarcare le differenze tra i vari sensi alcune parole all'interno delle frasi sono state opportunamente filtrate in quanto, si tratta di parole talmente comuni che non aiutano a discernere tra i sensi ma piuttosto tendono a "livellare" tutti i risultati; queste parole sono dette stopword. Queste parole sono state elencate all'interno di un file e sono utilizzate durante la fase di costruzione del dizionario per discernere se alcune parole possono diventare features per l'algoritmo

4 Struttura del codice

.

- `DocumentAnnotator` classe utilizzata per annotare delle sentence richiamando le librerie di Stanford NLP
- `WordDictionary` : classe per la costruzione di una dizionario sulla base dei documenti passati in input; sono presenti tutte le istanze di `Feature` facilmente accessibili tramite HashMap.

- `Feature` : questa classe gestisce tutte le informazioni relative ad una singola feature, ovvero a quale lemma corrisponde e in quanti testi prelevati dai synset essa compare
- `FileUtilities` : racchiude una collezione di metodi statici per leggere scrivere su file, oltre che parsificare il nome del file al fine di recuperare la classe di appartenenza. `StringUtilities` : racchiude una collezione di metodi statici per eseguire alcune operazioni sulle stringhe
- `Logging`: questa classe serve per costruire un file di log, impostando il livello di verbosità necessario
- `StopWords` : classe che gestisce il caricamento da più files di un elenco di StopWords, che verranno usate come filtro nella costruzione del dizionario
- `LeskAlgorithm` : classe che contiene l'implementazione dell'algoritmo di Lesk
- `SenseSignature`: classe che contiene una mappa dei synset recuperati per la disambiguazione e la loro signature associata, ovvero l'elenco di lemmi che sono stati trovati all'interno delle glosse dei vari synset
- `WordNetUtils`: classe avente lo scopo di interagire con la risorsa di Wordnet e recuperare sia i synset sia l'elenco di testi recuperabili per la disambiguazione

5 Risultati

A causa della povertà riscontrata nelle risorse messe a disposizione da Wordnet, purtroppo le prestazioni dell'algoritmo di Lesk sono risultate insoddisfacenti. Infatti il numero di esempi per synset è molto piccolo ed inoltre costituito da frasi molto brevi.
Per questo motivo, nella maggior parte dei casi l'intersezione tra la firma e il contesto della frase risulta vuota; non trovando informazioni con cui discernere l'algoritmo restituisce di solito il senso più comune della parola quando rappresenta una certa part-of-speech. Affinché l'algoritmo di Lesk possa "ingranare" e iniziare a discriminare il senso più corretto è necessario fornire un corpus di esempi e documenti molto più ampio.