UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche, Informatiche
e Matematiche

CORSO DI LAUREA IN INFORMATICA

Analisi e Valutazione Sperimentale di Tecniche di Similarità Semantica tra Frasi in un Contesto Multilingue

Marco Paradisi

Tesi di Laurea

Relatore:

Ing. Riccardo Martoglia

Anno Accademico 2018-2019

Ringrazio L'Ing. Riccardo Martoglia per la pazienza e la disponibilità in questo periodo di emergenza mondiale.

Grazie a Gabriele, senza il quale questo viaggio non sarebbe mai stato intrapreso.

Grazie a Giulia e Matteo che mi hanno supportato (e sopportato) in questi lunghi anni di studio.

Infine un ringraziamento ad Enrico, che mi ha sempre spronato e che mi ha dato il tempo necessario a terminare il mio percorso.

PAROLE CHIAVE

NLTK
SEMANTIC SIMILARITY
WORDNET
MULTIWORDNET
DISAMBIGUATION

INDICE DEI CONTENUTI

INDICE DELLE IMMAGINI

INTRODUZIONE

La nascita e l'evoluzione del web ha cambiato profondamente il modo in cui comunichiamo. Le informazioni sono condivise istantaneamente, i social media ne sono pieni, e ci si trova a far fronte a volumi inimmaginabili di testo da cui estrapolare soltanto ciò che interessa.
In sostanza la quantità di dati che si generano quotidianamente è maggiore di quella che si è in grado di gestire.
Da qui la necessità di uno strumento che compia sui testi quelle operazioni che erano fino a poco tempo fa deputate soltanto agli umani.
Il Natural Language Processing permette di processare il testo grezzo, filtrarlo, modellarlo in schemi confrontabili tra loro, e per questo risulta importantissimo.

Nella presente tesi sarà discusso ed approfondito il ruolo della similarità di significato tra frasi analizzando diversi approcci e rapportandoli ad un contesto multilingue.

La presente tesi è suddivisa in due parti ed è strutturata in quattro capitoli. Nella prima parte sono spiegati gli argomenti e gli strumenti utilizzati durante la fase di ricerca, mentre la seconda parte è relativa al progetto e alle verifiche sperimentali. Il documento è così strutturato:

Parte I: Il caso di studio

Capitolo 1 - Natural Language Processing

dove vengono presentate le tecniche atte a processare il testo che verranno utilizzate in questo scritto

Capitolo 2 - Librerie Python per NLP e Dizionari

dove vendono presentate le librerie e gli strumenti utilizzati e le funzioni che implementano quanto visto nel capitolo 1.

Capitolo 3 - Progettazione

dove vengono presentati gli algoritmi implementati per fornire la semantic similarity

Capitolo 4 - Prove sperimentali

dove vengono mostrati i risultati dell'applicazione degli algoritmi del capitolo 3.

Conclusioni

dove si descrivono le conclusioni a cui si è giunti dopo i test svolti.

Parte I

Il caso di studio

Capitolo 1

Natural Language Processing

1.1 Introduzione a NLP

NLP, ovvero elaborazione del linguaggio naturale, (dall'inglese Natural Language Processing), è il processo di processing automatico, mediante un computer, delle informazioni scritte o parlate in una determinata lingua. [1]
Più in generale, come suggerisce il nome, il Natural Language Processing si riferisce al trattamento informatico (computer processing) del linguaggio naturale, per qualunque scopo, senza considerare quanto approfonditamente viene fatta l'analisi. Per linguaggio naturale si intende la lingua che usiamo nella vita di tutti i giorni, come l'Inglese, l'Italiano e lo Spagnolo, ed è sinonimo di linguaggio umano, principalmente per poterlo distinguere dal linguaggio formale, come ad esempio i linguaggi di programmazione.
Si noti che il linguaggio naturale è la forma di comunicazione umana più comune, e non solo nella sua versione parlata, anche quella scritta è cresciuta in modo esponenziale con l'avvento e il radicamento dei social media.
Rispetto ad un linguaggio formale, il linguaggio naturale è più complesso, contiene sottintesi e ambiguità, il che lo rende più difficile da elaborare [2]

Per questo motivo il processo di elaborazione viene suddiviso in fasi, schematizzate per risultare vicine a quelle che si possono incontrare nel processo di elaborazione di un linguaggio di programmazione:

- analisi lessicale: scomposizione di un'espressione linguistica in parole (Tokenization);
- analisi grammaticale: associazione delle parti del discorso a ciascuna parola nel testo (PoS Tagging);
- analisi semantica: assegnazione di un significato (semantica) alla struttura sintattica e, di conseguenza, all'espressione linguistica.
-

Nell'analisi semantica la procedura automatica che attribuisce all'espressione linguistica un significato tra i diversi possibili è detta disambiguazione (o disambiguation)
Ciascuno dei suddetti concetti (Tokenization, PoS Tagging e disambiguation) sarà analizzato nel capitolo 1.3.

1.2 Il caso in esame

Scopo di questo documento è l'analisi e la valutazione sperimentale di "Tecniche di Similarità Semantica tra Frasi in un Contesto Multilingue".
Ovvero si cerca, partendo da un set predefinito di frasi, di ottenere un indice di similarità semantica tra ciascuna di esse.
Partendo dalla formula della similarità di base elaborata nell'articolo :

> "*Facilitate IT-Providing SMEs in Software Development: a Semantic Helper for Filtering and Searching Knowledge*
> *Ing. Riccardo Martoglia - 2011 [4]"*

si è cercato prima di estenderla ad altre lingue oltre all'inglese, dopodiché si è proceduto a confrontarla con altre metodologie per il calcolo della similarità semantica tra frasi, utilizzando dapprima un differente Dizionario nell'ambito della stessa libreria (NLTK) e successivamente una differente libreria, chiamata spaCy.

1.3 Text Wrangling

Perché il testo possa essere valutato per definire un indice di similarità semantica, indipendentemente dalla libreria che si vuole utilizzare, è necessario eseguire un'operazione di Text Wrangling, ossia un pre-processing eseguito sul testo grezzo (sulla frase da comparare nel caso in esame) atto ad ottenere un testo che sia leggibile dalla macchina e formattato secondo uno standard.

I processi di Text Wrangling che sono stati applicati sono:

- Tokenization;
- Stemming oppure Lemmatization;
- Rimozione delle stopword;
- PoS Tagging.

1.3.1 Tokenization

Il processo di Tokenization consiste nel dividere una stringa in una lista di sub-stringhe.
Ad esempio è possibile applicare un tokenizer per trovare e dividere le parole dalla punteggiatura in una stringa

Un esempio pratico :

stringa = ''Università di Modena e Reggio Emilia.''

stringa = ['Università', 'di', 'Modena', 'e', 'Reggio', 'Emilia', '.']

1.3.2 Stemming

Lo stemming è il processo di riduzione della forma flessa di una parola alla sua forma radice. La “radice“ non corrisponde necessariamente al lemma della parola, dove per lemma intendiamo la radice morfologica: solitamente basta che le parole siano riferite alla stessa radice, ad esempio che:

cantare
cantai
cantò

riconducano alla radice → *cant*

anche se quest'ultimo non è una valida radice per la parola (non si trova sul dizionario insomma), in quanto non è una “vera parola”. Il processo di stemming è utilizzato nei motori di ricerca per l'espansione di interrogazioni e in molti problemi di elaborazione del linguaggio naturale.

Esempio di stemming in lingua inglese:

Saying → Say

Says → Say

Said → Say

↑

Radici comuni per “Say”

am, are, is → be

qui sopra ad ulteriore esempio i verbi *am, are, is* vengono ricondotti all’infinito *be*

1.3.3 Lemmatization

La lemmatizzazione è il processo di riduzione di una forma flessa di una parola alla sua forma canonica, detta appunto lemma. In NLP, la lemmatizzazione è quella fase di processing del testo che determina in modo automatico il lemma di una data parola. Il processo può coinvolgere altre attività di elaborazione del linguaggio, quali ad esempio l'analisi grammaticale.
In molte lingue, le parole appaiono in diverse forme flesse.
Per esempio, in italiano il verbo *saltare* può apparire come *salta, saltò, saltando* e così via. La forma canonica, *saltare*, è il lemma della parola.
Quindi è la parola che si andrebbe a ricercare all'interno di un dizionario.

Stemming e Lemmatization generano entrambi la radice della parola interessata.
La differenza sta nel fatto che lo stemming può produrre un risultato che non sia una "vera" parola , mentre il lemma prodotto dalla lemmatization è sempre una parola del linguaggio prescelto.

Gli algoritmi di stemming sono in generale più veloci, mentre nella lemmatization si usa un corpus (vedi capitolo 2.1) per produrre un lemma e tale operazione risulta più lenta.
Inoltre è necessario indicare di quale parte del discorso fa parte la parola (verbo, nome, aggettivo…) per ottenere il lemma corretto, il PoS (Part Of Speech) Tagger si occupa proprio di eseguire un'analisi grammaticale sulle varie parole, associando a ciascuna un Tag in base al risultato (verbo, nome, aggettivo,…).

1.3.4 PoS Tagging

Le parole possono essere raggruppate in base al Part of Speech (PoS), letteralmente “parte del discorso”.
La grammatica tradizionale prevede poche tipologie di PoS (sostantivo, verbo, aggettivo, preposizione, avverbio, …) .
Modelli più recenti invece considerano un numero maggiore di PoS:
- 45 per il modello Penn Treebank;
- 87 per il modello Brown corpus.

Il PoS di una parola fornisce informazioni fondamentali per determinare il ruolo della parola stessa e di quelle vicine nella frase, risulterà quindi importante per applicare correttamente la Lemmatization.
Sapere se una parola è un pronome personale (io, tu,..) o un pronome possessivo (mio, tuo,…) permette di definire meglio quali parole è più probabile trovare nelle vicinanze .

Le 4 maggiori classi di parole, previste da quasi tutti i linguaggi, sono:
- sostantivi,
- verbi,
- avverbi,
- aggettivi.

Sono stati definiti alcuni insiemi di tag da utilizzare per il PoS tagging, il Penn Treebank e gli altri menzionati sopra sono riferiti alla lingua inglese. A titolo di esempio, il Penn Treebank tagset viene riportato in Figura 1.1:

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	*+, %, €*
CD	Cardinal number	*one, two*	TO	“to”	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, uh, oops*
EX	Existential ‘there”	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past particip.	*eaten*
JJR	Adj. comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj. superlative	*biggest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1,2,3*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, singular/mass	*dog, snow*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*dogs*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singul.	*Marco*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Alps*	#	Pound sign	*#*
PDT	Predeterminer	*all, both*	“	Left quote	“
POS	Possessive ending	*’s*	”	Right quote	”
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	*([{ <*
PP$	Possessive pronoun	*my, your*	)	Right parenthesis	*)] } >*
RB	Adverb	*never, often*	,	Comma	*,*
RBR	Adverb, comparative	*faster*	.	Sentence-final pun	*. ! ?*
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punt.	*: ; … -*
RP	Particle	*up, on ,off*			

FIGURA 1.1 - PENN TREEBANK TAGSET

1.3.5 Stopword

Le stopword sono parole considerate poco significative per l'analisi del testo perché possono essere usate spesso all' interno delle frasi.
Nell'ambito NLP, le stopword sono parole che vengono filtrate e separate dalle parole che trasportano effettivamente l'informazione.
La maggior parte delle stopword non hanno un vero significato se isolate dal testo per questo devono essere ignorate da programmi come, ad esempio, i motori di ricerca.
Le stopword variano da lingua a lingua. In lingua italiana possono essere considerate stopword gli articoli, le congiunzioni, le preposizioni.

Per quanto riguarda l'inglese, si riporta a titolo di esempio una lista delle stopword considerate tali dal corpus di NLTK:

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

1.3.6 Disambiguation - Introduzione

Il processo di Disambiguation punta ad identificare quale sia il significato di una parola in un determinato contesto.

Nell'ambito NLP, la disambiguazione del senso delle parole (WSD o Word Sense Disambiguation) è un problema relativo all'identificazione del senso di una parola usato in una frase. La soluzione a questo problema influisce su altre scritture informatiche, come il miglioramento della pertinenza dei motori di ricerca

Il cervello umano è abbastanza abile nel chiarimento delle parole, ovvero svolge in modo automatico e in maniera efficiente il processo di disambiguazione (in base al contesto è semplice capire se la parola portiere si riferisce al portiere di una squadra di calcio o ad un portiere d'albergo).

Il linguaggio umano si è sviluppato in un modo che riflette le innate capacità del cervello. In informatica, per lungo tempo è stata una sfida sviluppare la capacità dei computer di eseguire l'elaborazione del linguaggio naturale e l'apprendimento automatico.
Nel cap. 2.6 verranno descritte le tecniche implementate per l'applicazione di questa tecnica.

Capitolo 2

Librerie Python per NLP e Dizionari

In questo capitolo verranno descritte le librerie Python per NLP e le principali tecnologie utilizzate

2.1 Natural Language ToolKit (NLTK)

Impossibile riferirsi al Natural Language Processing in Python senza parlare della libreria NLTK. È la più famosa libreria NLP di Python, e ha portato ottimi risultati.
NLTK è responsabile di aver affrontato e risolto molti problemi di analisi del testo ed è anche per questo molto popolare in ambito accademico e per la ricerca.
Sul suo sito web, NLTK sostiene di essere una "*libreria incredibile per giocare con il linguaggio naturale*".
NLTK ha oltre 50 corpora e lessici, 9 stemmers e decine di algoritmi da scegliere. Questo è anche uno dei principali difetti di NLTK.

Per Corpora si intende:

> *una raccolta di dati linguistici, compilati come testi scritti o come trascrizione di un discorso registrato. Lo scopo principale di un corpus è verificare un'ipotesi sul linguaggio - ad esempio, per determinare come varia l'uso di un particolare suono, parola o costruzione sintattica. La linguistica del corpus tratta i principi e la pratica dell'uso dei corpora nello studio delle lingue. Un corpus computerizzato è un grande corpo di testi leggibili dalla macchina.[12]*

Un corpus di testo è quindi semplicemente un grande corpo di testo.

È stata sviluppata dai ricercatori Steven Bird ed Edward Loper al Department of Computer and Information Science dell'Università della Pennsylvania.[3]
NLTK punta a supportare la ricerca e l'insegnamento dell'NLP e di altri campi correlati, come la linguistica, le scienze cognitive, l'intelligenza artificiale, l'information retrieval, e il machine learning.

Le procedure supportate da NLTK comprendono “classificazione,” “tokenization”, “stemming”, “tagging” e “disambiguation”.

2.2 SpaCy

SpaCy[5] è una libreria open source per l'elaborazione del linguaggio naturale, scritta in Python e Cython.[6]
La libreria attualmente implementa modelli statistici di reti neurali in inglese, italiano, spagnolo e molte altre lingue.
A differenza della suite NLTK, che è molto utilizzata nel campo della ricerca e della didattica, SpaCy è maggiormente mirata alla realizzazione di software destinato alla produzione.

La sua filosofia è quella di presentare il migliore algoritmo per ogni scopo e risulta inoltre molto veloce nell’esecuzione.

2.2.1 Word Embedding

SpaCy utilizza come approccio il “word embedding”, che si può così definire:
una rappresentazione distribuita delle parole che permette di memorizzare le informazioni, sia semantiche che sintattiche, delle stesse.
Ciò avviene partendo da un corpus non annotato e costruendo uno spazio vettoriale in cui i vettori delle parole sono più vicini se le parole compaiono negli stessi contesti linguistici, cioè se sono riconosciute come semanticamente più simili (secondo l'ipotesi della semantica distribuzionale).
Ossia: più le parole sono simili (nel senso del significato) e più i loro vettori risulteranno vicini.

Nell’ambito NLP, si tratta di un insieme di tecniche di modellazione in cui parole o frasi di un vocabolario vengono mappate in vettori di numeri reali. In pratica consiste in un'operazione matematica di immersione in conseguenza della quale uno spazio costituito da una dimensione per parola viene trasformato in uno spazio vettoriale continuo di dimensione molto inferiore. Queste tecniche trovano applicazione nello studio della vicinanza semantica del discorso, in particolare nel mondo della semantica distribuzionale. [13]

Suddetta mappatura viene generata attraverso le reti neurali, modelli probabilistici, e rappresentazione esplicita riferita la contesto in cui si trova la parola.

Uno degli algoritmi più popolari proposti per l'apprendimento del word embedding è il GloVe[7] (Global Vectors) che è anche quello adottato dalla libreria SpaCy.

GloVe è un modello per la rappresentazione di parole distribuite.
Tale modello si traduce in un algoritmo di apprendimento per ottenere rappresentazioni vettoriali di parole, mappando le parole in uno spazio

significativo in cui la distanza tra le parole è correlata alla somiglianza semantica. È stato sviluppato come progetto open source alla Stanford University.

GloVe può essere utilizzato per trovare relazioni tra parole come sinonimi, codici postali, relazioni prodotto-azienda,… . Viene anche utilizzato dal modello SpaCy per creare incorporamenti di parole semantiche / vettori di caratteristiche.

2.2.3 L' "oggetto" nlp

Utilizzando la libreria spaCy in Python, dopo aver scaricato e installato un modello linguistico, si può caricarlo tramite l'istruzione spacy.load().
Ciò restituirà un oggetto Language contenente tutti i componenti e i dati necessari per elaborare il testo solitamente chiamato "nlp".
Chiamare l'"oggetto nlp" su una stringa di testo restituirà un documento (doc).

Anche se un documento viene elaborato, diviso in singole parole e successivamente annotato, contiene ancora tutte le informazioni del testo originale, come ad esempio i caratteri corrispondenti agli spazi bianchi.
È sempre possibile ottenere l'offset di un token nella stringa originale o ricostruire l'originale unendo i token e il loro spazio bianco finale.

In questo modo, non viene mai persa alcuna informazione durante l'elaborazione.

Da questo semplice esempio si può notare come da un singolo oggetto "doc" si possono estrarre tutte le informazioni relative alla frase (Lemma, Tagging, …):

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
for token in doc:
    print(token.lemma_, token.pos_, token.dep_)

RUN
```

```
Apple PROPN nsubj
be AUX aux
look VERB ROOT
at ADP prep
buy VERB pcomp
U.K. PROPN compound
startup NOUN dobj
for ADP prep
$ SYM quantmod
1 NUM compound
billion NUM pobj
```

FIGURA 1.2 - ESTRAZIONE DI INFORMAZIONI IN SPACY

Quando nlp viene invocato su di un testo, SpaCy prima tokenizza il testo per produrre un oggetto Doc. Il documento viene quindi elaborato in diversi passaggi: questa viene anche definita "pipeline di elaborazione".
La "pipeline" utilizzata dai modelli predefiniti è composta da un tagger, un parser e un riconoscimento di entità. Ogni componente della pipeline restituisce il documento elaborato, che viene quindi passato al componente successivo. [14]

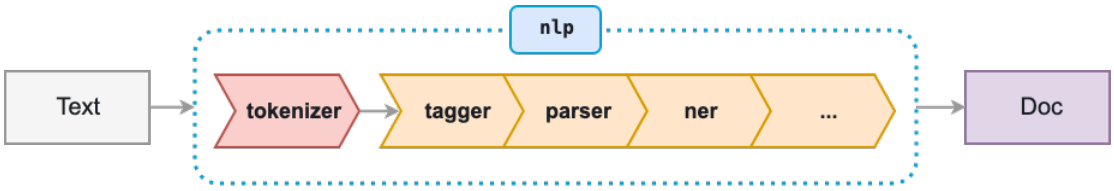


FIGURA 1.3 - PIPELINE DELL'OGGETTO NLP IN SPACY

Il parser assegnerà le "etichette" (label) di dipendenza, per NER invece si intende "Entity Recognizer" e si tratta del componente che rileva ed etichetta le entità nominate.
Il risultato finale sarà il documento contenete tutte le informazioni entro un unico oggetto.

2.3 NLTK vs SpaCy

Si vanno ora ad elencare, per maggiore chiarezza, le principali differenze tra le due librerie in oggetto.

NLTK è una libreria di elaborazione delle stringhe. Prende stringhe come input e restituisce stringhe o elenchi di stringhe come output, mentre SpaCy utilizza un approccio orientato agli oggetti. Quando si analizza un testo, SpaCy restituisce un oggetto documento (doc) le cui parole e frasi sono oggetti stessi.

SpaCy al contrario di NLTK non è orientato alla ricerca. Questo ha portato a decisioni di progettazione diverse rispetto a NLTK che nasce come piattaforma per l'insegnamento e la ricerca.
La differenza principale è che SpaCy offre quando possibile un solo algoritmo per una specifica funzione, evitando di avere più algoritmi che svolgono lo stesso lavoro.
SpaCy ha il supporto per i vettori di parole mentre NLTK no.
Nella tabella 2.1 vengono messe in evidenza e confrontate le principali funzionalità di spaCy e NLTK [5]

TABELLA 2.1 NLTK E SPACY - FUNZIONALITÀ

	spaCy	NLTK
Linguaggio di programmazione	Python	Python
Modelli di reti neurali	✅	❌
Vettori di parole integrati	✅	❌
Supporto multilingua	✅	✅
Tokenization	✅	✅
PoS tagging	✅	✅
Analisi delle dipendenze	✅	❌
Entity recognition	✅	✅
Entity linking	✅	❌

Ai fini di questo documento comunque le funzionalità necessarie (Tokenization, Stemming, Lemmatization, Stopword Removal, PoS Tagging) sono presenti in entrambe le librerie.

SpaCy non fornisce una vera e propria funzione di disambiguation anche se la funzionalità di ENTITY LINKING è così definita[5]: “Disambiguazione di entità testuali a identificatori univoci in una Knowledge Base”.

Verranno valutati nella Parte II i risultati sperimentali a confronto di tali funzionalità.

2.4 Wordnet

Wordnet[8] viene utilizzato in questo documento per la ricerca di sinonimi e per la disambiguazione dei termini utilizzando la libreria NLTK.
Di seguito sarà spiegato nel dettaglio il suo significato e come sia strutturato.

WordNet è un database semantico-lessicale per la lingua inglese che si propone di organizzare, definire e descrivere i concetti espressi dai vocaboli
È stato elaborato dal linguista George Miller presso l'Università di Princeton.
I termini sono raggruppati in gruppi, chiamati "synset" (dalla contrazione di synonym set) che contengono le parole di significato simile e le relazioni dei loro significati. All'interno dei “synset” le differenze di significato sono numerate e definite.

WordNet divide il significato di parola in due concetti: la “Word Form”, la forma scritta, e la “Word Meaning” ovvero il concetto espresso da tale parola.
Le relazioni che intercorrono tra un lemma ed il suo significato sono quindi la base della classificazione dei termini in WordNet.

WordNet assomiglia superficialmente a un thesaurus, in quanto raggruppa le parole in base ai loro significati.
Tuttavia ci sono alcune importanti differenze.

Innanzitutto in WordNet non vengono messe in relazione le parole ma i loro significati.
Ne consegue che se le parole si trovano molto vicine nella rete sono semanticamente non ambigue.
In seconda istanza, WordNet identifica le relazioni semantiche tra le parole, mentre i raggruppamenti di parole in un thesaurus non seguono alcun modello esplicito diverso dal significato di somiglianza.

In WordNet troviamo 4 macro-categorie lessicali :

- nomi,
- verbi,
- aggettivi,
- avverbi.

I nomi sono legati da 2 tipologie di relazioni:

- Relazioni lessicali: si instaurano tra word-forms (sia tra forme contenute nello stesso synset sia esterne).

  synomymy vs. antonymy

  Synonymy: due word-form sono sinonime se sostituendo l'una con l'altra non si cambia il valore di verità di una frase.
  Antonymy: due word-form sono antonime se il loro significato è opposto.

- Relazioni semantiche: si instaurano tra word-meaning.

  hyponymy vs. hyperonymy
  meronymy vs. holonymy

  Hyponymy : relazione "is a" (sottoinsieme).
  Hyperonymy: relazione inversa dell'hyponymy (famiglia di appartenenza)
  Meronymy: relazione "part of" (componente di).
  Holonymy: relazione inversa di meronymy

Alcuni aggettivi possono essere in relazione synonimy vs. antonymy mentre i verbi possiedono la relazione di <u>entailment</u> *(Il verbo A "entails" il verbo B, se lo svolgimento del primo implica lo svolgimento del secondo). [15]*

2.5 Sentence Similarity

Il fine ultimo di questo documento è il confronto fra diverse tecniche di Similarità tra frasi.

Verranno utilizzate formule di base più o meno articolate che verranno comunque descritte nella Parte II, ma oltre ad esse sarà utilizzata la funzione similarity della libreria SpaCy che verrà ora approfonditA per una miglior comprensione.

2.5.1 Similarity in SpaCy

La somiglianza è determinata confrontando i vettori di parole o "word embeddings" (vedi Cap.2.2.1).

Esempio di vettore per la parola: BANANA [5]

```
BANANA.VECTOR

array([2.02280000e-01, -7.66180009e-02,  3.70319992e-01,
       3.28450017e-02, -4.19569999e-01,  7.20689967e-02,
      -3.74760002e-01,  5.74599989e-02, -1.24009997e-02,
       5.29489994e-01, -5.23800015e-01, -1.97710007e-01,
      -3.41470003e-01,  5.33169985e-01, -2.53309999e-02,
       1.73800007e-01,  1.67720005e-01,  8.39839995e-01,
       5.51070012e-02,  1.05470002e-01,  3.78719985e-01,
       2.42750004e-01,  1.47449998e-02,  5.59509993e-01,
       1.25210002e-01, -6.75960004e-01,  3.58420014e-01,
       # ... and so on ...
       3.66849989e-01,  2.52470002e-03, -6.40089989e-01,
      -2.97650009e-01,  7.89430022e-01,  3.31680000e-01,
      -1.19659996e+00, -4.71559986e-02,  5.31750023e-01], dtype=float32)
```

FIGURA 2.1 - IL VETTORE "BANANA"

I vettori di parole consentono di importare conoscenze dal testo non elaborato nel modello.

La conoscenza è rappresentata come una tabella di numeri, con una riga per termine nel vocabolario. Se due termini vengono utilizzati in contesti simili, l'algoritmo che apprende i vettori dovrebbe assegnare loro righe abbastanza simili, mentre le parole utilizzate in contesti diversi avranno valori piuttosto diversi. Ciò consente di utilizzare i valori di riga assegnati alle parole come una sorta di dizionario, per ottenere informazioni sul significato delle parole nel testo.

2.6 Disambiguation

È stata studiata una ricca varietà di tecniche, dai metodi basati su dizionario che utilizzano le conoscenze codificate in risorse lessicali, ai metodi supervisionati di apprendimento automatico in cui un classificatore viene addestrato per ogni parola distinta su un corpus di esempi annotati manualmente dal senso.

In questo documento le due strategie che verranno applicate saranno:

Per la libreria NLTK l'algoritmo di Lesk:

L'algoritmo di Lesk si basa sul presupposto che le parole in un determinato "intorno" (sezione del testo) tenderanno a condividere un argomento comune. Una versione semplificata dell'algoritmo di Lesk è di confrontare la definizione del dizionario di una parola ambigua con i termini contenuti nel suo vicinato. Le versioni sono state adattate per utilizzare WordNet.
Verrà qui utilizzata la versione semplificata nella quale il significato corretto di ogni parola in un determinato contesto è determinato individualmente, trovando il senso che si sovrappone maggiormente tra la sua definizione del dizionario e il contesto dato. Piuttosto che determinare simultaneamente i significati di tutte le parole in un determinato contesto, questo approccio affronta ogni parola individualmente,

indipendentemente dal significato delle altre parole che si verificano nello stesso contesto.

Per la libreria spaCy:

Per radicare le entità nominate nel "mondo reale", spaCy fornisce funzionalità per eseguire il cosiddetto Entity Linking, che risolve un'entità testuale in un identificatore univoco da una base di conoscenza (Knowledge Base).
Gli script di elaborazione che spaCy fornisce utilizzano identificatori WikiData, ma è possibile creare la propria Knowledge Base e addestrare un nuovo modello di Entity Linking utilizzando quella base di conoscenza personalizzata.
Tale operazione va comunque al di là degli scopi di questo scritto.

2.7 Google Translator

Google Traduttore[10] è un servizio di traduzione automatica multilingue sviluppato da Google LLC.
Verrà utilizzato in questo contesto per generare delle soluzioni al problema della similarità in lingue diverse dall'inglese, traducendo i dataset dalla lingua prescelta all'inglese, dataset che verranno poi passati al software scritto per valutare l'indice di similarità tra le frasi.

Le altre tecniche che verranno usate per valutare la similarità tra frasi in lingue diverse dall'inglese saranno presentate nella Parte II di questo trattato.

Parte II

Progetto e Valutazione

Capitolo 3

Progettazione

In questo capitolo verranno presentati gli algoritmi che sono alla base di tutto il lavoro svolto e le scelte progettuali che sono state fatte per implementarli
Tutte le soluzioni proposte offrono in output un indice di similarità SIMMETRICO (A sym B = B sym A) e NORMALIZZATO per offrire un risultato compreso tra zero e uno, dove zero corrisponde ad una misura di similarità nulla o molto bassa, e uno corrisponde ad una misura di similarità molto alta .

3.1 Overview

Nella seguente tabella verranno riassunte tutte le metodologie applicate per ottenere il cosiddetto "indice di similarità tra frasi", per ogni metodologia sono segnate le tecniche di pre-processing usate e con quale libreria sono state implementate .

Si noti che il passaggio di tokenization è imprescindibile dal flusso di lavoro in tutte le metodologie e come l'aver scelto di utilizzare la tecnica di lemmatization al posto dello stemming abbia portato a dover necessariamente implementare il PoS tagging in tutte le soluzioni proposte.
La tecnica di disambiguation servirà a mettere in luce come l'utilizzo dei semplici sinonimi (o meglio synset) sia per certi aspetti troppo grossolano ed una grande approssimazione, in quanto non viene tenuto conto del contesto.

TABELLA3.1 - OVERVIEW

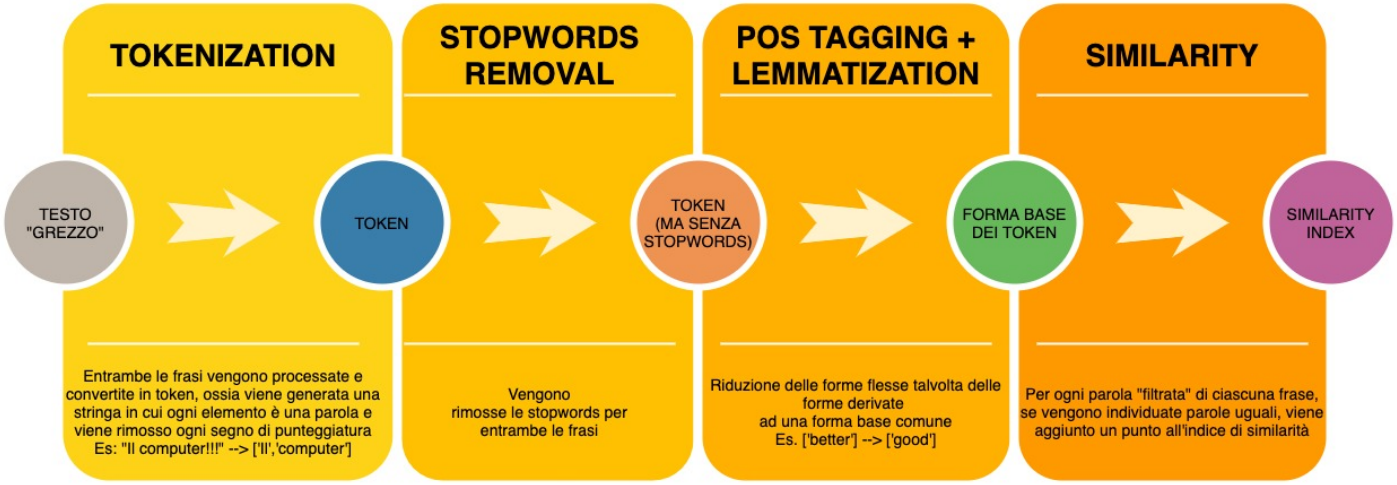
TECNICHE UTILIZZATE						
	Tokenization	PoS Tagging	Lemmatization	Rimozione Stopword	Disambiguation	Similarity
Baseline (NLTK)	✅ NLTK	✅ NLTK (spaCy per Italiano e Spagnolo)	✅ NLTK (spaCy per Italiano e Spagnolo)	✅ NLTK	❌	Solo parole uguali
Baseline(NLTK)+ Synonym	✅ NLTK	✅ NLTK (spaCy per Italiano e Spagnolo)	✅ NLTK (spaCy per Italiano e Spagnolo)	✅ NLTK	❌	Parole uguali e sinonimi
Baseline(NLTK)+ Synonym + Word Sense Disambiguation	✅ NLTK	✅ NLTK (spaCy per Italiano e Spagnolo)	✅ NLTK (spaCy per Italiano e Spagnolo)	✅ NLTK	✅ NLTK (Algoritmo di Lesk)	Parole uguali con stesso significato
SpaCy + stopword_rem + Entity Linking	✅ spaCy	✅ spaCy	✅ spaCy	✅ NLTK	✅ spaCy (Entity linking)	✅ spaCy

3.2 Algoritmo Baseline(NLTK)

La libreria su cui si appoggia questo metodo è la libreria NLTK

Per visualizzare meglio il flusso di lavoro eseguito con questo metodo viene prima fornito uno schema esplicativo, che si provvederà poi a descrivere nel dettaglio.

FIGURA 3.1 - ALGORITMO BASELINE (NLTK)



- Per ogni frase si esegue una prima fase di **tokenization**, in cui viene rimossa la punteggiatura e la frase (che è fondamentalmente una stringa) viene convertita in una lista di parole

  “Università di Modena !!!” —> [‘Università’,’di’,’Modena’]

Il codice per questa operazione in Python con NLTK è:

```
tokenizer = RegexpTokenizer(r'\w+')
frase1 = tokenizer.tokenize(frase1)
frase2 = tokenizer.tokenize(frase2)
```

- In seconda istanza vengono rimosse dalla lista tutte le stopword per la lingua selezionata, ovviamente le cosiddette stopword non sono uguali tra una lingua e l'altra.

```
frase1 = [w for w in frase1 if not w in stopwords.words("english")]
frase2 = [w for w in frase2 if not w in stopwords.words("english")]
```

- In seguito viene applicata sui token rimanenti la funzione di Lemmatization che porta ognuna delle parole alla propria forma base.

Questo il codice sorgente impiegato:

```
lemmatizer = WordNetLemmatizer()

frase1 = ([lemmatizer.lemmatize(w, tagger(w)) for w in frase1])

frase2 = ([lemmatizer.lemmatize(w, tagger(w)) for w in frase2])
```

Tale codice merita un paio di commenti:
innanzitutto si noti come il lemmatizer utilizzato sia direttamente collegato con il database lessicale Wordnet. Questo è necessario per trovare la forma base di una parola a partire da un dizionario.
Inoltre si pone in evidenza l'utilizzo della funzione
tagger() che restituisce, a partire da una parola, il PoS Tag corretto, facendo sempre riferimento al database di Wordnet.
Tale funzione servirà al lemmatizer per ricondurre la parola alla forma base corretta, in caso contrario la funzione lemmatize() assumerebbe che ciascun termine che le viene passato sia un nome.

Viene ora riportato il codice della funzione tagger()[11]

```
def tagger(word):

    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {        "J": wordnet.ADJ,
                        "N": wordnet.NOUN,
                        "V": wordnet.VERB,
                        "R": wordnet.ADV
                }

    return tag_dict.get(tag, wordnet.NOUN)
```

- L'ultimo step è il calcolo della Similarity, che in questa versione Baseline(NLTK) verrà effettuata con il seguente algoritmo:

```
for w1 in frase1:
    for w2 in frase2:
        if (w1 == w2):
            similarity = similarity + 1.0
```

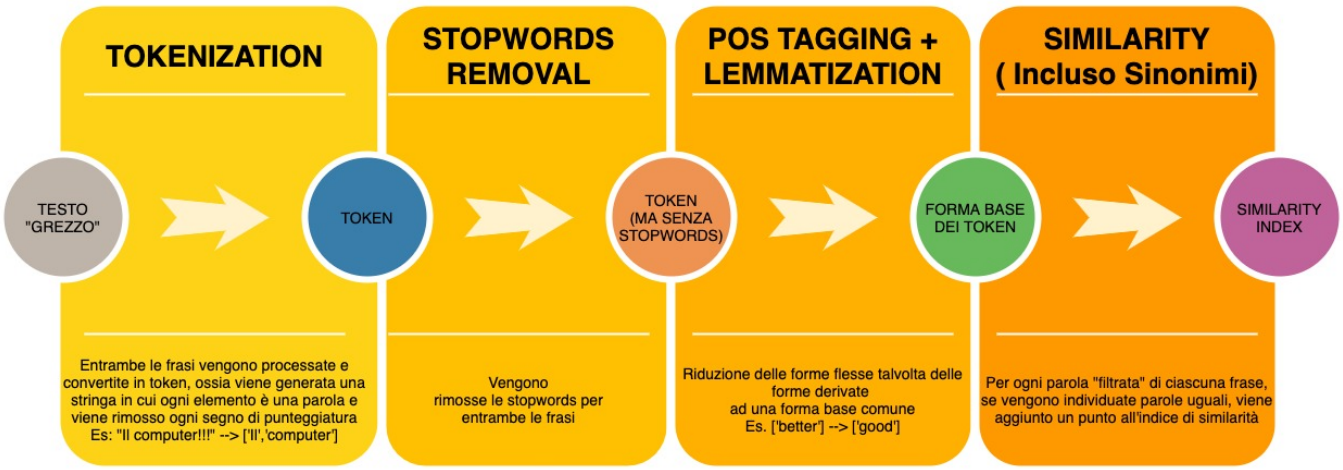
Ovvero, per ogni parola di ogni frase, se al netto di tutte le operazioni di pre-processing che sono state fatte vengono trovate parole uguali, per ogni parola uguale si aggiunge uno all'indice di similarità.
Infine tale indice viene normalizzato tra 0 e 1, per uniformare il risultato ottenuto con gli altri metodi proposti

I risultati ottenuti con questo algoritmo verranno presentati e discussi nel prossimo capitolo.

3.3 Algoritmo Baseline(NLTK)+ Synonym

Tale algoritmo è una estensione del precedente

FIGURA 3.2 - ALGORITMO BASELINE(NLTK)+SYNONYM



I primi tre step sono identici alla soluzione precedente, l'unica differenza sta nel calcolo dell'indice di similarità, che estende la ricerca della parola tra i suoi sinonimi. Ovvero: Se la parola w1 è contenuta tra i sinonimi della parola w2, viene aggiunto uno all'indice di similarità:

```
for w1 in frase1:
    for w2 in frase2:
        w2_syn = []
        for syn in wordnet.synsets(w2):
            for l in syn.lemmas():
                w2_syn.append(l.name())
        if (w1 == w2) or (w1 in w2_syn):
            similarity = similarity + 1.0
```

In sostanza è lo stesso algoritmo di prima, ma tiene conto anche delle parole che tra loro sono sinonimi, sfruttando i summenzionati synset di wordnet.
Risulta molto importante sottolineare che questa strada considera tutti i possibili sinonimi di tutti i possibili significati di un termine, senza

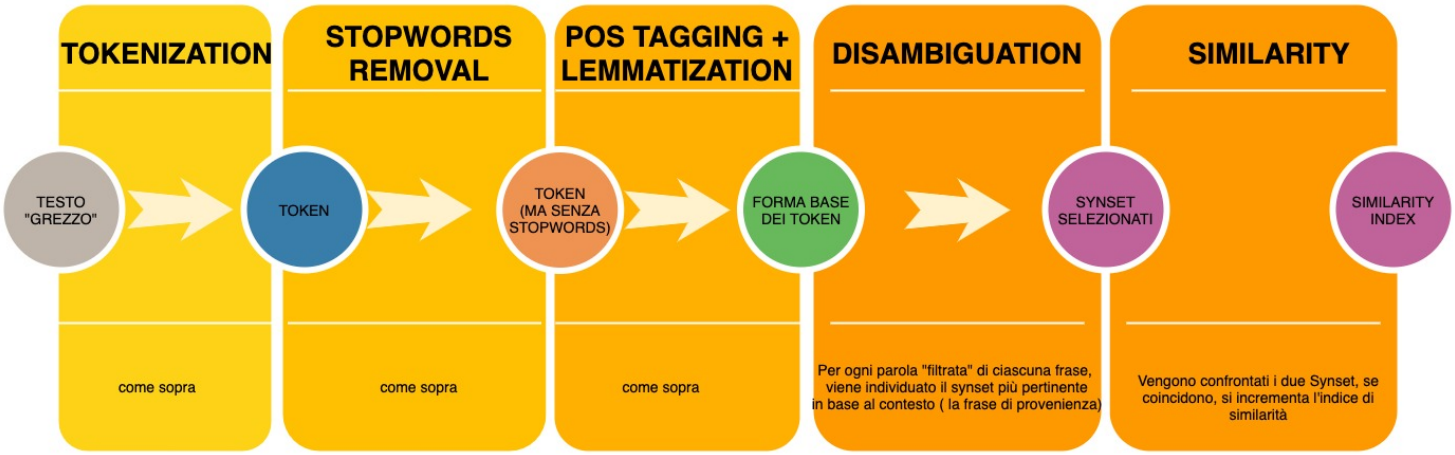
cercare di capire quale sia il significato corretto nel contesto della frase in cui viene impiegato (questo è invece fatto con il disambiguation)

I risultati ottenuti con questo algoritmo verranno presentati e discussi nel prossimo capitolo.

3.4 Baseline(NLTK)+ Synonym + WordSenseDisambiguation

Anche questa soluzione estende la precedente.
I synset analizzati nella soluzione precedente però contengono tutti i possibili significati di una data parola, senza tenere conto di omonimia o altre ambiguità. Per questo si è deciso di analizzare una soluzione che metta in luce appunto tali ambiguità.
Per questo, inoltre, la valutazione dell'indice di similarità è ora differente

FIGURA 3.3 - ALGORITMO BASELINE(NLTK)+SYNONYM+WORDSENSEDISAMBIGUATION



Viene eseguita un'operazione di disambiguation su entrambe le parole da valutare (per ogni coppia di parole nelle due frasi) per cui viene manutenuto solo il synset che l'algoritmo di Lesk ritiene più pertinente in base al contesto, ovvero la frase di provenienza.

Se i due synset coincidono, si incrementa di 1 l'indice di similarità.
Segue il codice implementato:

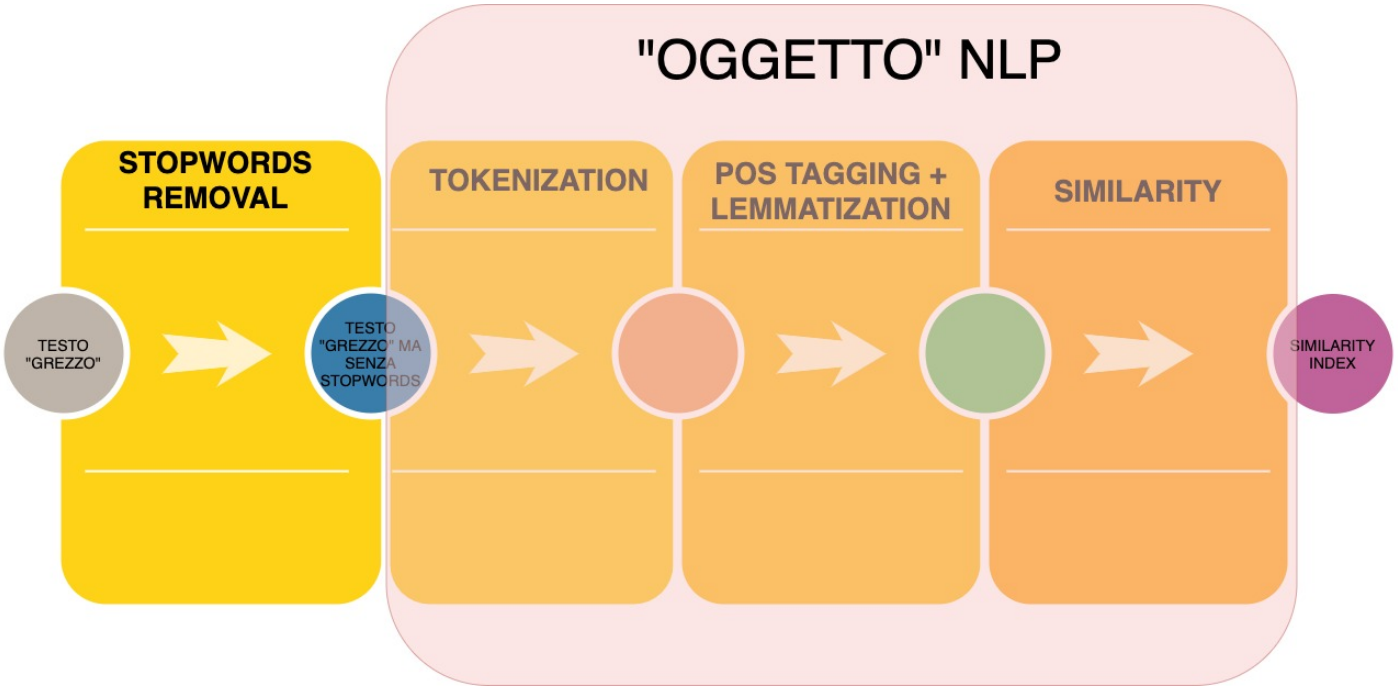
```
for w1 in frase1:
    for w2 in frase2:
        syn2 = lesk(context2, w2)
        syn1 = lesk(context1, w1)
        if syn2 and syn1:
            if (syn1 == syn2):
                similarity = similarity + 1.0
```

I risultati ottenuti con questo algoritmo verranno presentati e discussi nel prossimo capitolo.

3.5 SpaCy + stopword_rem + Entity Linking

Questa soluzione analizzata è stata testata così come fornita dalla libreria spaCy, con un'unica eccezione:
come si è visto nel cap.2.2.3, nell' "oggetto" nlp di spaCy vengono già effettuate molte delle operazioni necessarie al pre-processing delle frasi, ma non la rimozione delle stopword, che quindi è stata implementata appoggiandosi alla libreria NLTK.

Questo per garantire un ambiente di confronto quanto più uniforme possibile. Infatti è stato valutato sperimentalmente che, come era lecito aspettarsi, evitare la rimozione delle stopword avrebbe falsato il risultato finale.



Il codice qui risulta molto essenziale per la similarity:

```
nlp = spacy.load("modulo_da_caricare_a_seconda_della_lingua")
nlp1 = nlp(frase1)
nlp2 = nlp(frase2)
punteggio = nlp1.similarity(nlp2)
```

3.6 Modifiche apportate per il contesto Multilingua

Tutte le soluzioni descritte sopra sono state applicate anche alle altre lingue considerate oltre all'inglese, ossia italiano e spagnolo.

Di seguito vengono mostrati gli accorgimenti necessari ad adattare ciascuna soluzione alla lingua prescelta.

3.6.1 Baseline(NLTK)

Le stopword da eliminare vanno in questo contesto caricate nella lingua prescelta :

```
frase1 = [w for w in frase1 if not w in stopwords.words("italian")]
frase2 = [w for w in frase2 if not w in stopwords.words("italian")]
```

oppure

```
frase1 = [w for w in frase1 if not w in stopwords.words("spanish")]
frase2 = [w for w in frase2 if not w in stopwords.words("spanish")]
```

L'assenza nella libreria NLTK di un lemmatizer per la lingua italiana e per la lingua spagnola, hanno portato a sostituire tale modulo con uno stemmer per la versione Baseline(NLTK)

Tale scelta è applicabile soltanto al summenzionato algoritmo, in quanto la sostituzione del lemmatizer con lo stemmer andrebbe a impedire il corretto riconoscimento dei lemmi da parte del database lessicale (MultiwordNet). Ad esempio il verbo "andare" viene trasformato dallo

stemmer nella radice “and”, che non ha significato nella lingua italiana e per la quale non è possibile individuare i synset.
La versione dell’algoritmo Baseline(NLTK) non necessita di interfacciarsi al database lessicale, quindi si può attuare la sostituzione.

segue il codice dello stemmer in esame:

```
sb = SnowballStemmer(“lingua_del_dataset”)
temp1 = []
temp2 = []
for w in frase1:
    temp1.append(sb.stem(w))
for w in frase2:
    temp2.append(sb.stem(w))
frase1 = temp1
frase2 = temp2
```

Segue che non sarà possibile passare direttamente il dataset in lingua italiana e spagnola ai due algoritmi “Baseline(NLTK)+ Synonym + WordSenseDisambiguation” e “Baseline(NLTK)+ Synonym”.
Si provvederà quindi ad esaminare il comportamento dei due algoritmi passando loro detti dataset tradotti in lingua inglese con l’ ausilio del traduttore di Google.

3.6.2 Modifiche ad algoritmo SpaCy + stopword_rem + Entity Linking

La libreria spaCy supporta molte lingue tra cui italiano e spagnolo.
Per utilizzare la lingua prescelta è sufficiente eseguire il comando

```
spacy.load(lingua_prescelta)
```

dove “lingua prescelta” è, più specificatamente, il modello statistico predefinito disponibile per la lingua.(vedi capitolo 2.2)

3.6.3 Utilizzo di Google Translator

Come ulteriore verifica sperimentale si è deciso di valutare l'applicazione degli algoritmi costruiti per la lingua inglese ai dataset italiano e spagnolo, traducendoli preventivamente con Google Translator.
Tale scelta è risultata obbligata per gli algoritmi "Baseline(NLTK)+ Synonym" e "Baseline(NLTK)+ Synonym + Word Sense Disambiguation", ma è stata poi estesa anche alle altre soluzioni per testarne l'efficacia. I risultati ottenuti con questo metodo verranno presentati e discussi nel prossimo capitolo.

Capitolo 4

Prove sperimentali

Vengono qui presentate le diverse prove sperimentali effettuate e i risultati ottenuti

4.1 Dataset

Per testare al meglio gli algoritmi sono state scelte 8 frasi da un pool di frasi differenti, ciascuna contenente un "trabocchetto lessicale", come ad esempio parole omonime con significati diversi.
Lo scopo ultimo sarà valutare come le diverse scelte di progettazione fatte impattino sulla similarità "attesa".
Come anticipato nel capitolo 2, il cervello umano è abilissimo nel comprendere i diversi significati in base al contesto, per questo il metro con il quale gli algoritmi si dovranno confrontare è la valutazione soggettiva dell'autore del grado di similarità tra le diverse frasi proposte.

Tale valutazione verrà espressa secondo tre livelli

- ROSSO - grado di similarità nullo o molto basso
- GIALLO - grado si similarità medio, le frasi sono in qualche modo correlate
- VERDE - grado di similarità alto: le frasi risultano molto “vicine” a livello semantico

Ad ogni algoritmo testato si provvederà poi ad assegnare un range da associare ai valori ottenuti per ricondurli al codice colore summenzionato Il numero di match con il modello “desiderato” verrà poi valutato secondo la formula dell’ efficacia:

$$\text{Efficacy} = \sum_{i=1}^{n} \left(1 - \left| ga_i - gr_i \right| \right)$$

dove la sommatoria è fatta sugli n confronti, ga_i è il grado atteso del confronto i, gr_i è il grado rilevato del confronto i. I gradi varranno 0 per il rosso, 0.5 per il giallo e 1 per il verde

Il valore ottenuto riportato in percentuale valuterà la bontà dell’algoritmo utilizzato.

4.2 Prove con dataset in lingua inglese

Il dataset utilizzato per le prove in lingua inglese è il seguente:

1. *”forwards are the players on a football team who play nearest to the opposing team's goal “,*
2. *”in soccer, the traditional role of a striker is to score goals;”,*
3. *"a hockey season that ended far too early “,*
4. *"in baseball, a batter or hitter is a person whose turn it is to face the pitcher.”,*
5. *"the cat is a domestic species of small carnivorous mammal. “,*
6. *"Bass Guitars ranging from exclusive starter packs and affordable basses, to premium instruments from the likes of Fender”,*
7. *"the tiger is the largest species among the Felidae . It is most recognisable for its dark vertical stripes.”,*
8. *"Popular wild sea bass is a sustainable and delicious white fish”,*

Tali frasi hanno parole in comune come “bass” nella 6 e 8 che però hanno due significati completamente diversi, 4 parlano di sport, 3 di animali una di musica.
Tutto considerato si è compilato una “maschera” con cui ora si procederà a confrontare i risultati sperimentali ottenuti dall’implementazione dei vari algoritmi.

Elementi comuni, differenze e ambiguità verranno messe in luce nei commenti alle tabelle che seguono.

Verranno ora mostrate le tabelle, commentate, riferite a ciascun algoritmo e riportanti i risultati delle prove sperimentali effettuate:

	ALGORITMO Baseline(NLTK)															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0,16		0		0		0		0		0		0
2						0		0		0		0		0		0
3								0		0		0		0		0
4										0		0		0		0
5												0		0,16		0
6														0		0,13
7																0
8																

TABELLA 4.1 ALGORITMO BASELINE(NLTK)
ROSSO: [0,0.1] GIALLO:(0.1,0.15] VERDE:(0.15,1]

Vengono qui messe in luce le parole uguali, la similarità è infatti maggiore di 0 solo in corrispondenza di due frasi che abbiano parole uguali, si noti come per parole uguali l'algoritmo rilevi anche goal-goals. Questo è dovuto al lemmatizer che riconduce le parole alla forma base, eliminando in questo caso il plurale

Efficacy (vedi 4.1): 85%

	ALGORITMO Baseline(NLTK)+ SYNONYM															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0,16		0,2		0,13		0		0		0		0
2						0,2		0,16		0		0		0		0
3								0		0		0		0		0
4										0		0		0		0
5												0		0,16		0
6														0		0,25
7																0
8																

TABELLA 4.2 ALGORITMO BASELINE(NLTK)+ SYNONYM
ROSSO: [0,0.1] GIALLO:(0.1,0.15] VERDE:(0.15,1]

In questa prova vengono rilevati più termini, perché la ricerca è stata estesa a tutti i synset di quella parola. Ad esempio viene registrata la somiglianza tra “striker” e “hitter”, fra “end” e “goal”, fra “end” e “play”. È facile capire come si creino in questo modo moltissime “false similarità”

Efficacy (vedi 4.1): 83%

	ALGORITMO Baseline(NLTK)+ SYNONYM + WORD SENSE DISAMBIGUATION															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0,16		0		0		0		0		0		0
2						0		0		0		0		0		0
3								0		0		0		0		0
4										0		0		0		0
5												0		0,16		0
6														0		0
7																0
8																

TABELLA 4.4 ALGORITMO BASELINE(NLTK)+ SYNONYM + WORD SENSE DISAMBIGUATION
ROSSO: [0,0.1] GIALLO:(0.1,0.15] VERDE:(0.15,1]

Se da un lato si vede come la disambiguation abbia eliminato correttamente le “false similarità” di tabella 4.2, ci si rende conto di come in questo caso sia applicata in modo troppo rigido, impedendo all’algoritmo di rilevare, ad esempio, le relazioni esistenti tra la frase 1 e la frase 2.

Terminati questi primi test con la libreria NLTK si mette in luce come i risultati appaiano:
Tutti e tre gli algoritmi si comportano bene quando si tratta di rilevare una NON similarità tra le frasi, cosa da tenere in conto.
Si mette in evidenza anche il fatto che la similarità calcolata applicando l’algoritmo di Lesk risulti da un lato limitante, dall’altro metta in luce egregiamente solo i concetti che effettivamente sono semanticamente correlati.
Il confronto tra i risultati ottenuti verrà effettuato nel capitolo 4.5

Efficacy (vedi 4.1): 87%

	SpaCy + STOPWORD_REM + ENTITY LINKING															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0,79		0,72		0,66		0,32		0,37		0,46		0,39
2						0,68		0,64		0,36		0,44		0,49		0,47
3								0,64		0,39		0,40		0,49		0,48
4										0,44		0,46		0,54		0,50
5												0,35		0,76		0,65
6														0,47		0,61
7																0,71
8																

TABELLA 4.5 ALGORITMO SPACY + STOPWORD_REM + ENTITY LINKING
ROSSO: [0,0.60) GIALLO:[0.60,0.75) VERDE:[0.75,1]

La soglia qui applicata risulta ovviamente differente, ma una volta aggiustata i risultati sono ottimali; vediamo come l'algoritmo rilevi praticamente tutte le relazioni così come erano state previste, con qualche possibile eccezione.
A titolo sperimentale si era provato a non rimuovere le stopword e applicare l'algoritmo. Il risultato, che si era rivelato molto più impreciso, ha dimostrato come la rimozione delle stopword sia un passo fondamentale nel processo di analisi della similarità tra due frasi.

Efficacy (vedi 4.1): 98%

4.3 Prove con dataset in lingua italiana

Il dataset utilizzato per le prove in lingua italiana è il seguente:

1. *"il rombo del razzo era assordante",*
2. *"il rombo è un pesce che vive nei mari profondi",*
3. *"gli uccelli acquatici come le anatre hanno penne impermeabili ",*
4. *"Il missile entra nell'atmosfera del pianeta con un angolo sbagliato",*
5. *"il cane è il miglior amico dell'uomo ",*
6. *"le piume delle oche sono estremamente efficaci per trattenere il calore",*
7. *"il portiere dell'albergo ci mostrò le camere, indicando con un martello",*
8. *"il portiere parò un calcio di rigore che sembrava impossibile",*

Anche qui si trovano diverse parole omonime, con significati diversi come "rombo" in 1 e 2 ne è solo un esempio, 4 parlano di animali. 2 di razzi. Il mix eterogeneo ci permette di effettuare diverse prove.

Si noti che l'assenza del lemmatizer per la lingua italiana nella libreria NLTK ha portato a vagliare i due algoritmi "Baseline(NLTK)+ Synonym" e "Baseline(NLTK)+ Synonym + Word Sense Disambiguation" passando loro il dataset preventivamente tradotto con l'ausilio del traduttore della suite Google.
Per completezza, al dataset tradotto, è stato applica anche l'algoritmo SpaCy + Stopword_Rem + Entity Linking. Seguono i risultati ottenuti:

	ALGORITMO Baseline(NLTK)															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0,3		0		0		0		0		0		0
2						0		0		0		0		0		0
3								0		0		0		0		0
4										0		0		0		0
5												0		0		0
6														0		0,16
7																0
8																

TABELLA 4.6 ALGORITMO BASELINE(NLTK)- ITA
ROSSO: [0,0.1] GIALLO:(0.1,0.15] VERDE:(0.15,1]

come per la lingua inglese, vengono correttamente rilevate e segnalate le parole uguali, che in questo caso sono in numero minore.
La presenza dello stemmer al posto del lemmatizer non ha influito sul rilevamento delle stesse.

Efficacy (vedi 4.1): 82%

	ALGORITMO TRANSLATOR + Baseline(NLTK)+ SYNONYM															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0		0		0		0		0		0		0
2						0		0,2		0		0		0		0
3								0		0		0,2		0		0
4										0		0		0		0
5												0		0		0
6														0		0
7																0
8																

TABELLA 4.7 ALGORITMO BASELINE(NLTK)+ SYNONYM - ITA
ROSSO: [0,0.1] GIALLO:(0.1,0.15] VERDE:(0.15,1]

In questa prova vengono rilevati più termini, anche la corrispondenza penna-piuma di 3-6 viene rilevata.

Efficacy (vedi 4.1): 89%

	ALGORITMO TRANSLATOR + Baseline(NLTK)+ SYNONYM + WORD SENSE DISAMBIGUATION															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0		0		0		0		0		0		0
2						0		0		0		0		0		0
3								0		0		0,2		0		0
4										0		0		0		0
5												0		0		0
6														0		0
7																0
8																

TABELLA 4.8 ALGORITMO BASELINE(NLTK)+SYNONYM + WORD SENSE DISAMBIGUATION - ITA
ROSSO: [0,0.1] GIALLO:(0.1,0.15] VERDE:(0.15,1]

Come prima vediamo come l'algoritmo rilevi la correlazione presente tra alcune delle frasi, ma come già visto per la lingua inglese, questo algoritmo, applicando in maniera troppo severa la disambiguation, non raggiunge ancora risultati ottimali.

Efficacy (vedi 4.1): 92%

	SpaCy + STOPWORD_REM + ENTITY LINKING															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0,55		0,35		0,47		0,65		0,52		0,52		0,61
2						0,70		0,50		0,56		0,73		0,63		0,63
3								0,44		0,43		0,63		0,56		0,55
4										0,76		0,59		0,68		0,69
5												0,59		0,66		0,82
6														0,73		0,72
7																0,69
8																

TABELLA 4.9 ALGORITMO SPACY + STOPWORD_REM + ENTITY LINKING - ITA
ROSSO: [0,0.65) GIALLO:[0.65,0.70) VERDE:[0.70,1]

vediamo come l'algoritmo rilevi alcune delle le relazioni così come erano state previste, con molti più errori rispetto a quanto atteso.
Il motivo di questo è da individuarsi nel modulo linguistico di default per la lingua italiana che viene caricato da spaCy.
Nel modello non sono stati caricati vettori di parole, pertanto il risultato del metodo Doc.similarity si baserà su tagger, parser e NER (Entity Recognition), che potrebbero non fornire utili giudizi di somiglianza.

Come ulteriore test si provvede a processare il dataset con Google Translator (vedi cap. 3.6.3) e successivamente a passarlo all'algoritmo SpaCy + STOPWORD_REM + ENTITY LINKING in lingua inglese:

Efficacy (vedi 4.1): 67%

	TRANSLATOR +ALGORITMO SpaCy + STOPWORD_REM + ENTITY LINKING															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0,30		0,28		0,54		0,20		0,34		0,27		0,30
2						0,48		0,48		0,43		0,50		0,38		0,35
3								0,33		0,37		0,71		0,36		0,27
4										0,49		0,56		0,55		0,51
5												0,42		0,61		0,49
6														0,50		0,42
7																0,46
8																

TABELLA 4.10 ALGORITMO TRANSLATOR + SPACY + STOPWORD_REM + ENTITY LINKING - ITA
ROSSO: [0,0.40) GIALLO:[0.40,0.70) VERDE:[0.70,1]

Otteniamo un risultato nettamente più valido del precedente, dimostrando come la libreria spaCy si riveli al momento la scelta migliore ma soltanto caricando un modello linguistico adeguato.
Il risultato più basso rispetto, per esempio, a quello del test precedente è da imputarsi al fatto che in questo dataset si sono deliberatamente scelte frasi con una similarità più "lontana", ovvero con un contesto semantico non così vicino come nell'esempio in lingua inglese o in quello che seguirà in lingua spagnola. Tale differenza è maggiormente percepita dalla libreria spaCy che valuta settorialmente la distanza semantica tra le due frasi.

Efficacy (vedi 4.1): 78%

4.4 Prove con dataset in lingua spagnola

Il dataset utilizzato per le prove in lingua spagnola è il seguente:

1. *"Ramon es un jugador de fútbol, es el blanco de todos los chistes.",*
2. *"la novia tenía un vestido blanco",*
3. *"El temporal causó varias muertes.",*
4. *"El préstamo fue temporal, el banco fue muy claro al respecto",*
5. *"El Banco de España alerta del riesgo de exclusión financiera de los mayores",*
6. *"al menos 4 personas muertas deja tormenta Gloria en España",*
7. *"el matrimonio es un compromiso importante",*
8. *"En el fútbol, el defensor tiene el papel de obstruir a los atacantes.",*

In questo dataset le relazioni semantiche sono mantenute più strette (sposa-matrimonio) (fútbol-fútbol)

	ALGORITMO Baseline(NLTK)															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0,3		0		0		0		0		0		0,2
2						0		0		0		0		0		0
3								0,25		0		0,25		0		0
4										0,2		0		0		0
5												0,13		0		0
6														0		0
7																0
8																

TABELLA 4.11 ALGORITMO BASELINE(NLTK)- SPA
ROSSO: [0,0.2) GIALLO:[0.2,0.3) VERDE:(0.3,1]

Per come è stato costruito il dataset, anche questa volta, il risultato dell'algoritmo Baseline(NLTK)è quello atteso, le parole uguali vengono correttamente rilevate.
Vengono rilevate soltanto le parole uguali, ma essendo le frasi appaiate come significato, tale significato non viene ovviamente rilevato.

Efficacy (vedi 4.1): 83%

	ALGORITMO TRANSLATOR + Baseline(NLTK)+ SYNONYM															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0		0		0		0		0		0		0,2
2						0		0		0		0		0		0
3								0		0		0,25		0		0
4										0,25		0		0		0
5												0,1		0		0
6														0		0
7																0
8																

TABELLA 4.7 ALGORITMO BASELINE(NLTK)+SYN - SPA
ROSSO: [0,0.1) GIALLO:[0.1,0.2) VERDE:[0.2,1]

In questa soluzione vengono rilevate più corrispondenze, il motivo risiede anche che, traducendo il dataset, si perdono quei " trabocchetti lessicali" presenti nel dataset stesso, evitando falsi positivi. L'allargamento del contesto semantico ai sinonimi permette di rilevare le similarità tra le frasi come si era voluto intendere, raggiungendo una percentuale di riscontri molto più alta.

Efficacy (vedi 4.1): 94%

	ALGORITMO TRANSLATOR + Baseline(NLTK)+ SYNONYM + WORD SENSE DISAMBIGUATION															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0		0		0		0		0		0		0,2
2						0		0		0		0		0		0
3								0		0		0		0		0
4										0,1		0		0		0
5												0		0		0
6														0		0
7																0
8																

TABELLA 4.8 ALGORITMO TRANSLATOR + BASELINE(NLTK)+ SYNONYM + WORD SENSE DISAMBIGUATION - SPA
ROSSO: [0,0.1] GIALLO:(0.1,0.15] VERDE:(0.15,1]

Come prima vediamo come l'algoritmo rilevi la correlazione presente tra alcune delle frasi, ma l'applicazione stretta dell'algoritmo di disambiguation rimuove due delle corrispondenze semantiche tra le frasi.

Efficacy (vedi 4.1): 92%

	SpaCy + STOPWORD_REM + ENTITY LINKING															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0,65		0,65		0,81		0,62		0,67		0,72		0,84
2						0,54		0,79		0,70		0,59		0,71		0,62
3								0,65		0,70		0,71		0,54		0,67
4										0,73		0,66		0,77		0,78
5												0,76		0,59		0,64
6														0,50		0,60
7																0,73
8																

TABELLA 4.12 ALGORITMO SPACY + STOPWORD_REM + ENTITY LINKING - SPA
ROSSO: [0,0.65) GIALLO:[0.65,0.70) VERDE:[0.70,1]

In questo caso, come era stato nei test per la lingua italiana, vengono correttamente individuate le frasi che erano state considerate come accoppiate. Restano molte “false similarità” dovute al modello linguistico, che anche per la lingua spagnola appare carente.

Efficacy (vedi 4.1): 57%

	TRANSLATOR + SpaCy + STOPWORD_REM + ENTITY LINKING															
	FRASE 1		FRASE 2		FRASE 3		FRASE 4		FRASE 5		FRASE 6		FRASE 7		FRASE 8	
	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO	VALORE ATTESO	VALORE RILEVATO
1				0,35		0,49		0,50		0,46		0,60		0,42		0,74
2						0,27		0,28		0,26		0,42		0,37		0,31
3								0,51		0,56		0,68		0,45		0,59
4										0,71		0,54		0,52		0,57
5												0,57		0,54		0,55
6														0,48		0,58
7																0,53
8																

TABELLA 4.13 ALGORITMO TRANSLATOR + SPACY + STOPWORD_REM + ENTITY LINKING - SPA
ROSSO: [0,0.60) GIALLO:[0.60,0.65) VERDE:[0.65,1]

Vediamo come anche se non tutti i risultati combacino con le nostre aspettative, i divari tra i valori siano più netti e in generale la bontà della soluzione appaia pressoché ottimale.
L'utilizzo di un dataset dove le frasi sono appaiate con una stretta correlazione di significato viene ripagato da spaCy che rileva correttamente la quasi totalità delle corrispondenze.

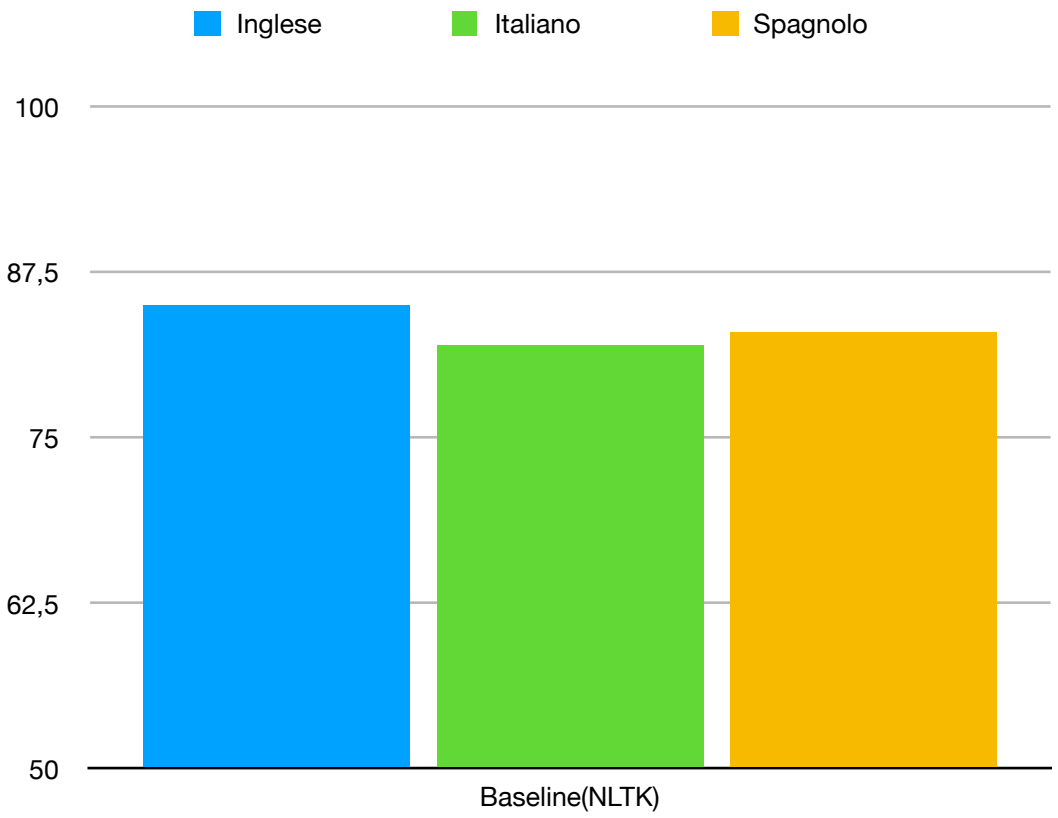
Efficacy (vedi 4.1): 94%

4.5 Grafici dei risultati

Vengono qui riportati una serie di grafici riassuntivi che per ogni algoritmo mostrano quanto efficiente si è mostrato nei test effettuati, in rapporto al numero di rilevazioni in linea con le aspettative come mostrato nella formula del cap. 4.1. Si metteranno altresì in luce alcune considerazioni con l'aiuto dei suddetti grafici.
Sull'asse delle ordinate si indicherà sempre l'indice di Efficacy (vedi 4.1) riportato in percentuale.

FIGURA 4.1 - GRAFICO ALGORITMO BASELINE(NLTK)

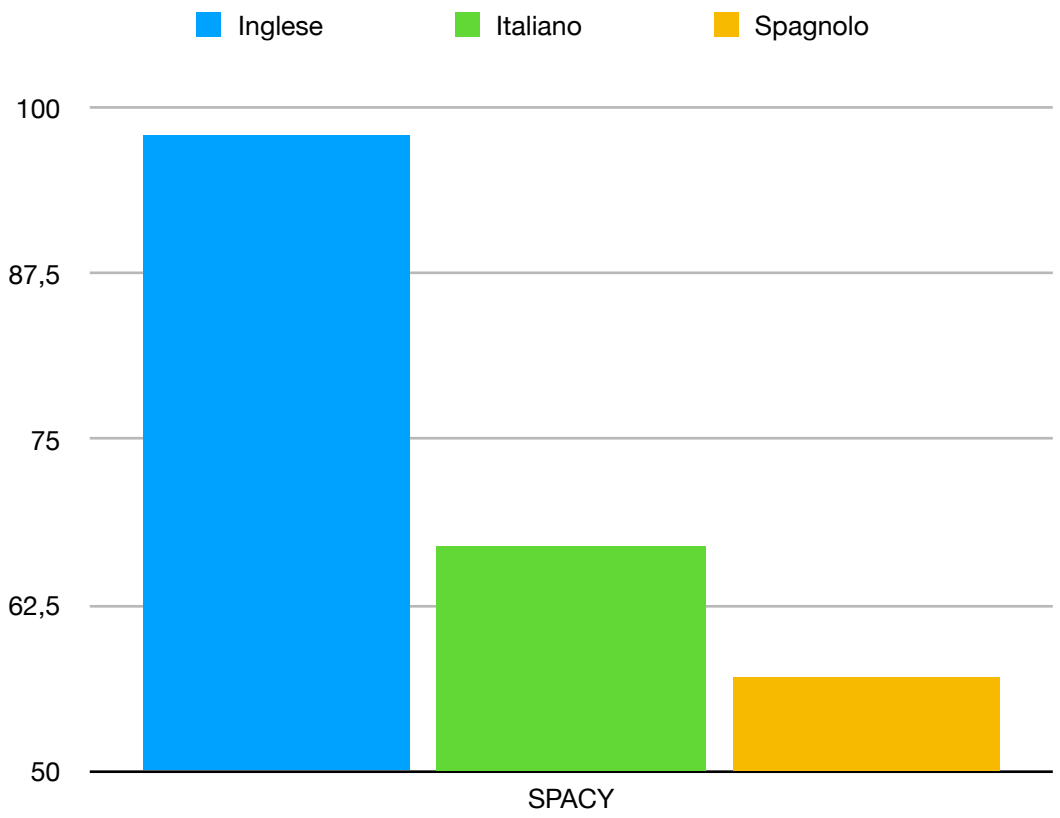
Risulta evidente come l'applicazione dell'algoritmo Baseline(NLTK) dia, a grandi linee, gli stessi risultati, indipendentemente dalla lingua scelta.



Questo perché tale algoritmo non utilizza particolari strumenti lessicali fatta esclusione per la rimozione delle stopword e il lemmatizer, che però in questo contesto risulta intercambiabile con lo stemmer utilizzato per italiano e spagnolo come spiegato in 4.3

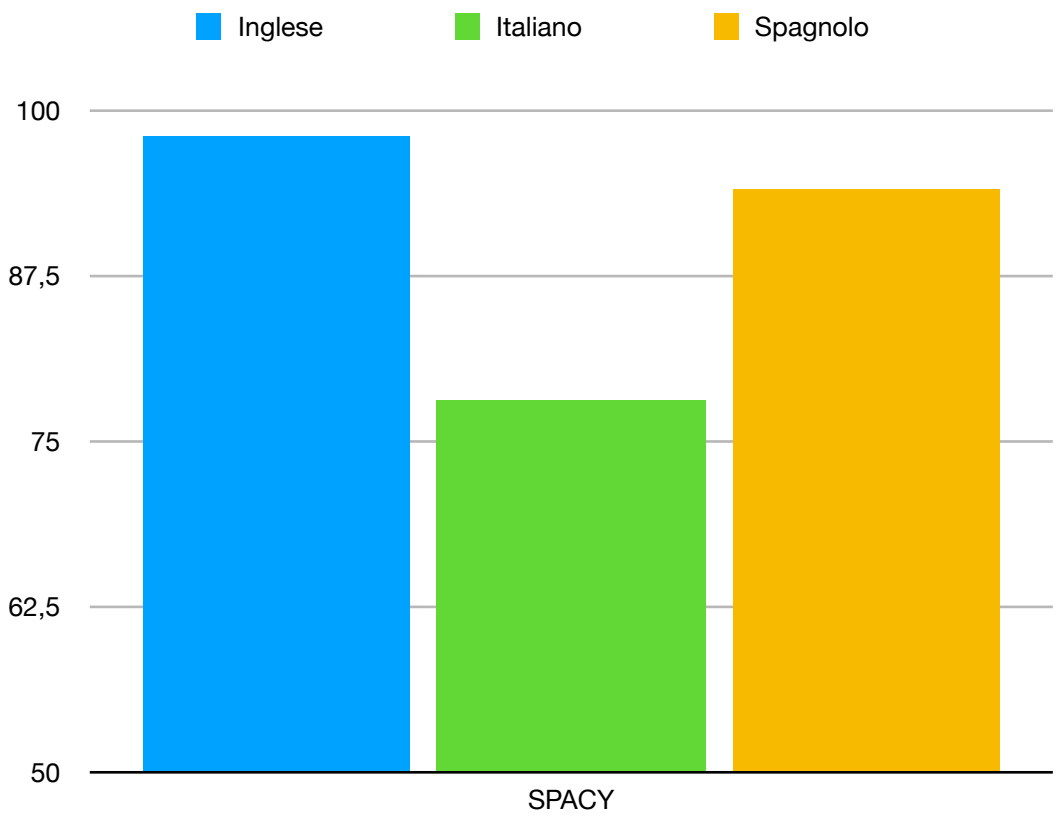
Risulta ormai chiaro che spaCy sia la libreria che ha dato i migliori risultati nell'analisi della similarità semantica tra frasi in questo scritto.
Tuttavia si è visto come le sue prestazioni calino bruscamente se non viene utilizzato un modello linguistico contenente word vector (vettori di parole), come nel caso dell'italiano o dello spagnolo.

FIGURA 4.2 GRAFICO ALGORITMO SPACY



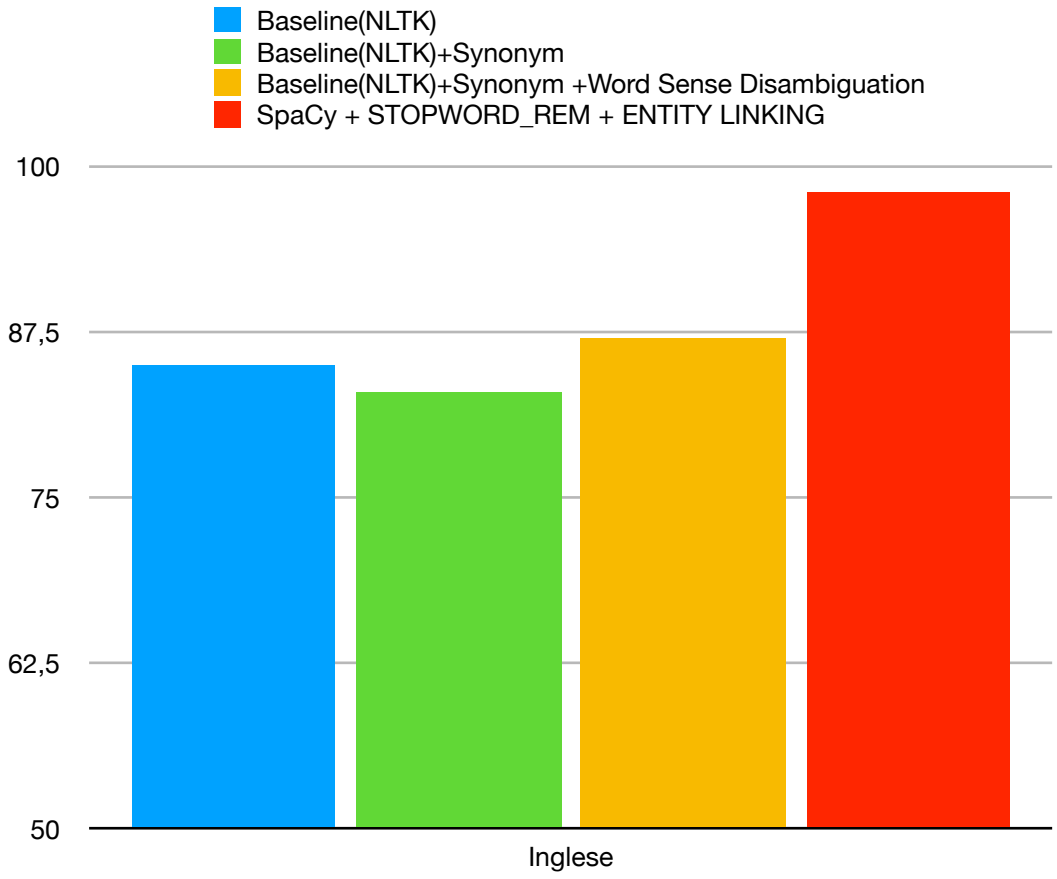
L'utilizzo di un traduttore per riportare i dataset in lingua inglese ha però permesso di colmare egregiamente questo divario, e di mettere in luce, come desiderato, la bontà della soluzione in rapporto alla similarità semantica tra frasi.
Infatti la scelta di un dataset italiano dove le frasi risultano semanticamente più "distanti" è ben mostrato nel grafico in figura 4.3

FIGURA 4.3 GRAFICO ALGORITMO TRANSLATOR + SPACY



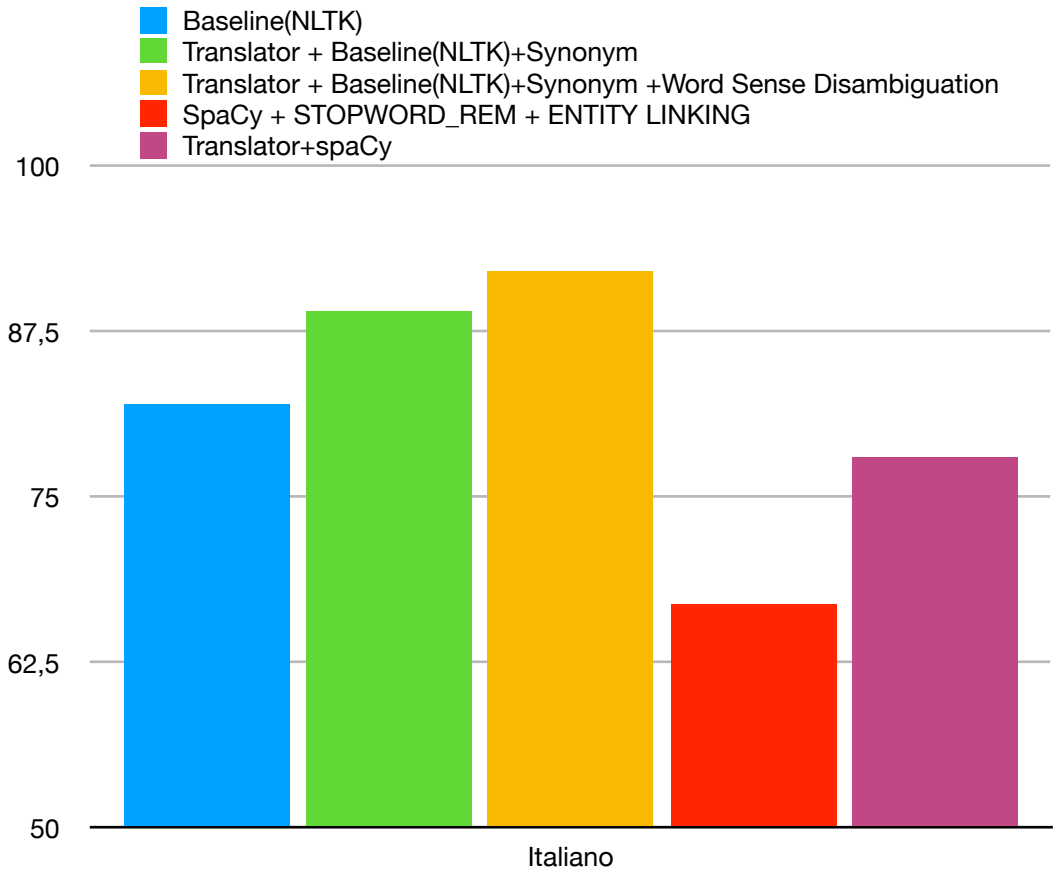
Verranno ora analizzate nello specifico le soluzioni che sono state applicate ad ogni singola lingua presa singolarmente

FIGURA 4.4 ALGORITMI PER LINGUA INGLESE



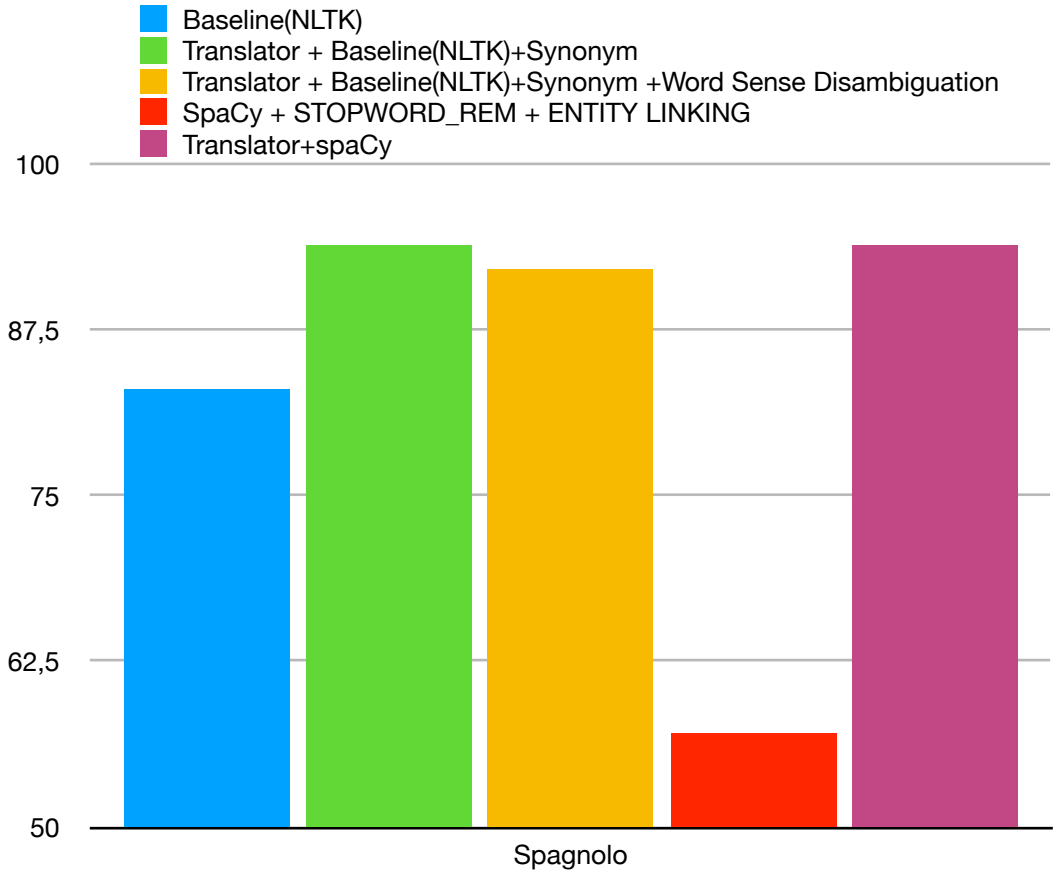
Si nota come la disambiguation porti un leggero miglioramento nell'efficacia degli algoritmi testati, ma Spacy risulti la soluzione più performante.

FIGURA 4.5 ALGORITMI PER LINGUA ITALIANA



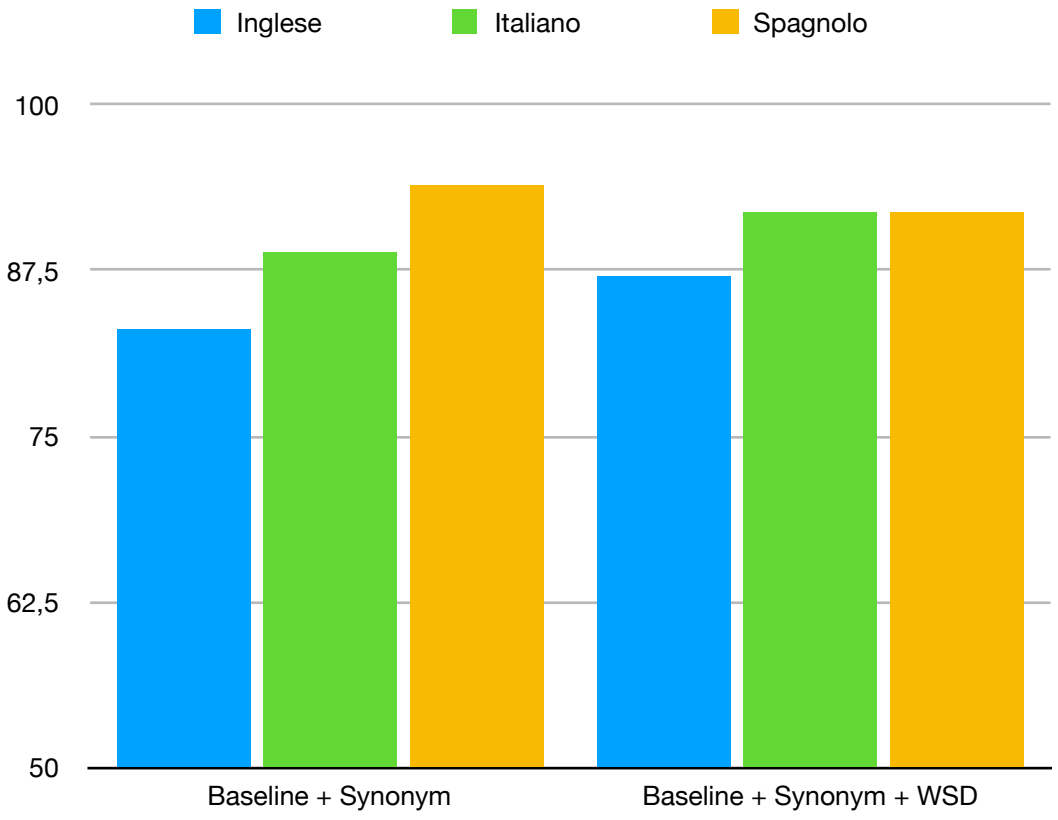
Come si può vedere. la scelta del dataset ha influito non tanto sulle scelte fatte dalla libreria NLTK, che ha correttamente individuato i termini relazionati e li ha selezionati, quanto invece sull'efficacia della libreria SpaCy, che ricercando la "vicinanza" semantica delle frasi, avendo scelto frasi troppo debolmente correlate ha mostrato una Efficacy minore a quanto atteso.

FIGURA 4.6 ALGORITMI PER LINGUA SPAGNOLA



Avendo selezionato un dataset di frasi più strettamente correlate tra loro, si vede come il risultato di Spacy (quando applicato alla lingua inglese) sia il migliore.

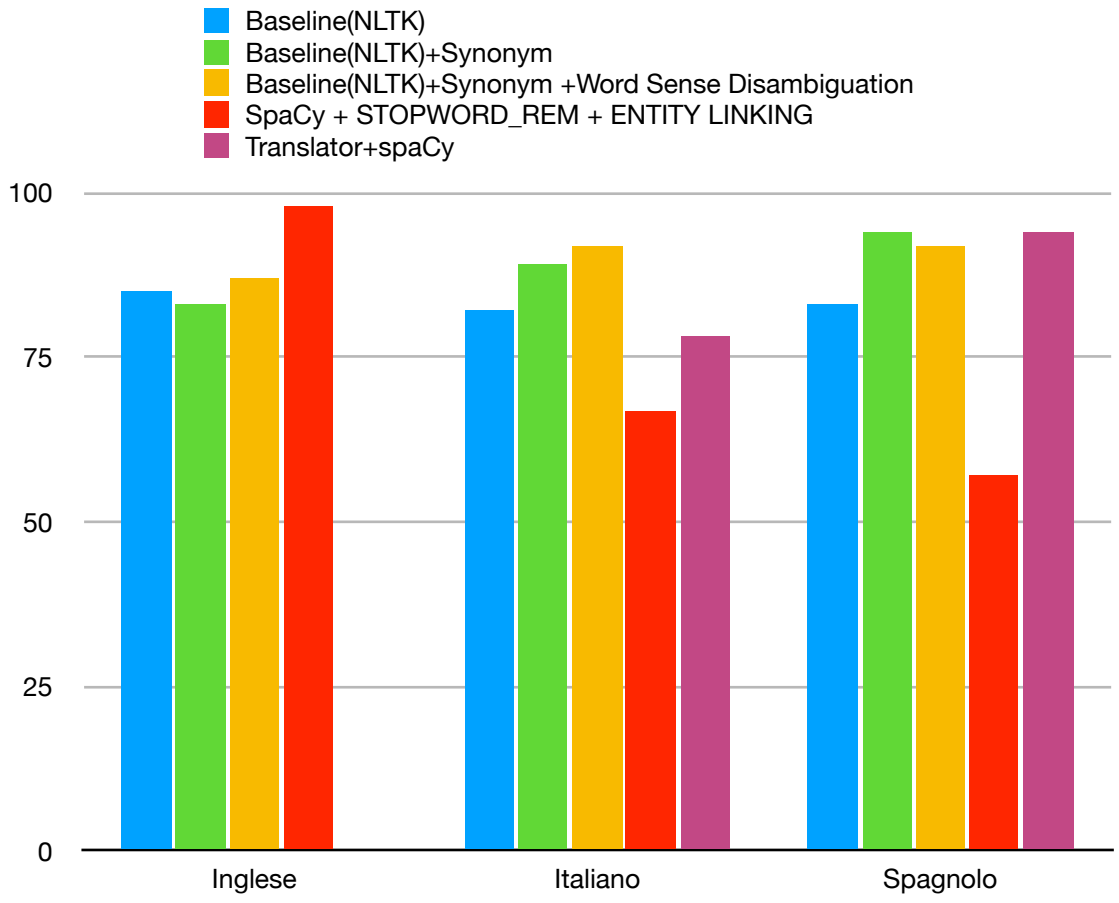
FIGURA 4.7 GRAFICO ALGORITMO BASELINE + SYNONYM



Per completezza si riporta il grafico dei restanti algoritmi presi singolarmente, si noti come le soluzioni in Italiano e Inglese siano implementate con l'utilizzo del traduttore.

Viene infine riportato un grafico riassuntivo che mostra l'efficacia di ogni singolo algoritmo in rapporto alla lingua.
Gli algoritmi Baseline(NLTK)+ Synonym e Baseline(NLTK)+ Synonym + Word Sense Disambiguation in lingua italiana e spagnola sono stati implementati con l'ausilio del traduttore.

FIGURA 4.4 GRAFICO RIASSUNTIVO



Le conclusioni che si possono trarre dal grafico in esame saranno analizzate, appunto, nella sezione "conclusioni"

CONCLUSIONI

Gli obiettivi di questa tesi erano quelli di studiare ed approfondire le conoscenze sulle tecniche di Semantic Similarity in un contesto multilingue paragonando due diverse librerie e tecniche .

Dopo aver analizzato le varie soluzioni, e come si evince chiaramente dal grafico in figura 4.4, la soluzione che applicava la libreria spaCy, per quanto riguarda la lingua Inglese è risultata nettamente superiore alle altre. In larga misura questo è sicuramente dovuto anche alla sua semplicità di implementazione e alla sua filosofia “un solo algoritmo per ogni funzionalità”.

Il supporto di più di 50 lingue vantato però sul sito della libreria stessa si è rivelato essere meno performante di quanto fosse lecito aspettarsi.

Il problema di fondo si trova nel modello computazionale che viene fornito per la lingua italiana e per la lingua spagnola.

La mancanza dei vettori di parole o word vectors ha fortemente impattato sui risultati che sono stati ottenuti e quindi è sensato pensare che, con un adeguato modello linguistico computazionale (almeno pari a quello per la lingua inglese) da passare alla libreria si possano ottenere gli stessi, ottimi, risultati nella verifica della similarità semantica tra frasi.

È vero altresì che l’utilizzo del modello inglese in concomitanza con lo strumento Google Translator per tradurre i dataset si è rivelato molto molto buono e quindi è da tenere in considerazione

Le altre soluzioni si sono rivelate in certo modo carenti nel contesto multilingue per l’assenza di un buon Lemmatizer per la lingua italiana spagnola all’interno della suite NLTK.

Con l’implementazione di tale strumento, e visto l’ottimo interfacciamento dei database di Multiwordnet con Wordnet è ragionevole pensare che si possano ottenere ottimi risultati anche con NLTK. MultiWordNet è un database lessicale multilingue strettamente

allineato con WordNet.
I synset italiani vengono creati in corrispondenza dei synset WordNet Princeton, ove possibile, e le relazioni semantiche vengono importate dai corrispondenti synset inglesi; cioè, supponendo che ci siano due synset in Wordnet e una relazione tra loro, la stessa relazione vale tra i synset corrispondenti in italiano. Mentre il progetto sottolinea l'utilità di un rigoroso allineamento tra reti di parole di lingue diverse, la gerarchia multilingue implementata è in grado di rappresentare vere idiosincrasie lessicali tra le lingue, come lacune lessicali e differenze di denotazione.
La mancanza di un lemmatizer in NLTK ha impedito il perseguirsi di questa strada e l'utilizzo di questo strumento, ma una delle migliorie possibili è certamente l'implementazione di tale lemmatizer per testare gli algoritmi "Baseline(NLTK)+ Synonym" e "Baseline(NLTK)+ Synonym + Word Sense Disambiguation" direttamente sul dataset in lingua originale e senza l'utilizzo di un traduttore.

Per la lingua inglese invece le soluzioni scelte si sono rivelate molto veloci ed affidabili, anche se il numero di opzioni possibili e strade praticabili per compiere un'operazione nella libreria NLTK è così alto da renderlo sia un punto di forza, che una grande debolezza di questa libreria. Per questo si è preferito l'utilizzo della libreria spaCy.

Anche nel caso di NLTK si è comunque visto che l'utilizzo degli algoritmi in lingua inglese accodati a Google Translator per tradurre le frasi porta a delle prestazioni paragonabili a quelle viste per la lingua inglese.

SpaCy però ha mostrato come l'utilizzo di un modello statistico ridotto, dove non siano stati caricati vettori di parole, porti il risultato del metodo Doc.similarity a basarsi solo su tagger, parser e NER (Entity Recognition), che potrebbero non fornire utili giudizi di somiglianza, o meglio, come si è riscontrato. possono fornire molti riscontri inesatti.

Anche in questo caso però si è visto come l'utilizzo dell'algoritmo in tandem con Google transistor abbia portato a risultati che rispecchiano le aspettative.

Da un certo punto di vista, però, utilizzare queste soluzioni porta ad essere dipendenti dalla qualità della traduzione, ma per contesti in cui esista un traduttore affidabile e performante (come è Google Translator per il linguaggio “colloquiale”) si conclude sia meglio fare affidamento su questi strumenti ed utilizzare le librerie analizzate nella lingua di origine.

Bibliografia/Sitografia

1 https://it.wikipedia.org/wiki/Elaborazione_del_linguaggio_naturale

2 https://www.linkedin.com/pulse/overview-natural-language-processing-wei-li

3 Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O'Reilly Media Inc.

4 https://www.isgroup.unimo.it/article/seke11.pdf

5 http://spacy.io

6 https://cython.org/

7 https://nlp.stanford.edu/projects/glove/

8 https://wordnet.princeton.edu/

10 https://translate.google.it/

11 https://www.machinelearningplus.com/nlp/lemmatization-examples-python/

12 (Crystal, David, 1992. Un dizionario enciclopedico di lingue e lingue Oxford: Blackwell.)

13 https://it.wikipedia.org/wiki/Word_embedding

14 https://spacy.io/usage/processing-pipelines

15"Introduction to WordNet: an on-line lexical database" – George A. Miller
"WordNet: a lexical database" – Marco Degemmis