



d1.unito.it

DIPARTIMENTO
DI INFORMATICA

TLN-LAB

utilizzo di risorse
lessicografiche per la concept
similarity e la WSD

Daniele Radicioni

credits

- the following slides have been mostly built on materials from:
 - M. Lesk. Automatic Sense Disambiguation using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *Proceedings of the 5th International Conference on Systems Documentation*, 1986.
 - Tanveer Siddiqui and U.S.Tiwary, *Natural Language Processing and Information Retrieval*, Oxford University, 2008.

conceptual similarity with WordNet



Daniele Radicioni - TLN

conceptual similarity with WN

- dati in input due termini, il task di conceptual similarity consiste nel fornire un punteggio numerico di similarità che ne indichi la vicinanza semantica.
 - ad esempio, la similarità fra i sensi *car* e *bus* potrebbe essere 0.8 in una scala $[0, 1]$, in cui 0 significa che i sensi sono completamente dissimili, mentre 1 significa identità.
- per risolvere il task di conceptual similarity è possibile sfruttare la struttura ad albero di WordNet.

input

- l'input per questa esercitazione è costituito da coppie di termini contenute nel file *WordSim353* (disponibile nei formati *.tsv* e *.csv*)
 - il file contiene 353 coppie di termini utilizzati come testset in varie competizioni internazionali
 - a ciascuna coppia è attribuito un valore numerico $[0, 1]$, che rappresenta la similarità fra gli elementi della coppia.

consegna

- l'esercitazione **consiste nell'implementare tre misure di similarità** basate su WordNet.
- per ciascuna di tali misure di similarità, calcolare gli **indici di correlazione di Spearman and gli indici di correlazione di Pearson** fra i risultati ottenuti e quelli 'target' presenti nel file annotato.



WIKIPEDIA
The Free Encyclopedia

Article

[Talk](#)

Pearson correlation coefficient

From Wikipedia, the free encyclopedia

Definition [\[edit \]](#)

Pearson's correlation coefficient is the [covariance](#) of the two variables divided by the product of their [standard deviations](#). The form of the definition involves a "product moment", that is, the mean (the first [moment](#) about the origin) of the product of the mean-adjusted random variables; hence the modifier *product-moment* in the name.

For a population [\[edit \]](#)

Pearson's correlation coefficient when applied to a [population](#) is commonly represented by the Greek letter ρ (rho) and may be referred to as the *population correlation coefficient* or the *population Pearson correlation coefficient*. Given a pair of random variables (X, Y) , the formula for ρ ^[7] is:

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \quad (\text{Eq.1})$$

where:

- cov is the [covariance](#)
- σ_X is the [standard deviation](#) of X
- σ_Y is the standard deviation of Y



WIKIPEDIA
The Free Encyclopedia

Article

Talk

Spearman's rank correlation coefficient

From Wikipedia, the free encyclopedia

Definition and calculation [[edit](#)]

The Spearman correlation coefficient is defined as the [Pearson correlation coefficient](#) between the [rank variables](#).^[3]

For a sample of size n , the n [raw scores](#) X_i, Y_i are converted to ranks $\text{rg } X_i, \text{rg } Y_i$, and r_s is computed from:

$$r_s = \rho_{\text{rg}_X, \text{rg}_Y} = \frac{\text{cov}(\text{rg}_X, \text{rg}_Y)}{\sigma_{\text{rg}_X} \sigma_{\text{rg}_Y}}$$

where

- ρ denotes the usual [Pearson correlation coefficient](#), but applied to the rank variables.
- $\text{cov}(\text{rg}_X, \text{rg}_Y)$ is the [covariance](#) of the rank variables.
- σ_{rg_X} and σ_{rg_Y} are the [standard deviations](#) of the rank variables.

Wu & Palmer

$$cs(s_1, s_2) = \frac{2 \cdot depth(LCS)}{depth(s_1) + depth(s_2)}$$

- la misura di similarity di Wu & Palmer si basa sulla struttura di WordNet
- *LCS* è il primo antenato comune (Lowest Common Subsumer) fra i sensi s_1 e s_2 ; e $depth(x)$ è una funzione che misura la distanza fra la radice di WordNet e il synset x .

Wu & Palmer, 1994

<https://arxiv.org/pdf/cmp-lg/9406033.pdf>

Daniele Radicioni - TLN

Shortest Path

$$\text{sim}_{\text{path}}(s_1, s_2) = 2 \cdot \text{depthMax} - \text{len}(s_1, s_2)$$

- for a specific version of WordNet, *depthMax* is a fixed value.
- the similarity between two senses (s_1, s_2) is the function of the shortest path $\text{len}(s_1, s_2)$ from s_1 to s_2 .
 - if $\text{len}(s_1, s_2)$ is 0, $\text{sim}_{\text{path}}(s_1, s_2)$ gets the maximum value of $2 * \text{depthMax}$.
 - if $\text{len}(s_1, s_2)$ is $2 * \text{depthMax}$, $\text{sim}_{\text{path}}(s_1, s_2)$ gets the minimum value of 0.
- thus, the values of $\text{sim}_{\text{path}}(s_1, s_2)$ are between 0 and $2 * \text{depthMax}$.

Leacock & Chodorow

$$\text{sim}_{LC}(s_1, s_2) = -\log \frac{\text{len}(s_1, s_2)}{2 \cdot \text{depthMax}}$$

- when s_1 and s_2 have the same sense, $\text{len}(s_1, s_2)=0$. in practice, we add 1 to both $\text{len}(s_1, s_2)$ and $2 \cdot \text{depthMax}$ to avoid $\log(0)$.
- thus the values of $\text{sim}_{LC}(s_1, s_2)$ are in the interval $(0, \log(2 \cdot \text{depthMax} + 1)]$

termini vs. sensi

- **attenzione**: l'input è costituito da coppie di *termini*, mentre la formula utilizza *sensi*.
- per calcolare la similarity fra 2 termini immaginiamo di prendere la **massima similarity fra tutti i sensi del primo termine e tutti i sensi del secondo termine**.
 - l'ipotesi è cioè che i due termini funzionino come contesto di disambiguazione l'uno per l'altro.
 - nella formula c sono i concetti che appartengono ai synset associati ai termini w_1 e w_2 .

$$\text{sim}(w_1, w_2) = \max_{c_1 \in s(w_1), c_2 \in s(w_2)} [\text{sim}(c_1, c_2)]$$

Daniele Radicioni - TLN

WSD



Daniele Radicioni - TLN

Word Sense Disambiguation

Word sense disambiguation (WSD) is an *open problem* of natural language processing, which comprises the process of *identifying which* sense of a word (i.e. *meaning*) *is used in any given sentence*, when the word has a number of distinct senses (polysemy).

WSD

- disambiguating word senses has the potential to improve many natural language processing tasks, such as [machine translation](#), [question-answering](#), [information retrieval](#), and [text classification](#).
- in their most basic form, WSD algorithms take [as input a word in context](#) along with a fixed inventory of [potential word senses](#), and return as output the correct word sense for that use.

what is WSD

WordNet Sense	Spanish Translation	Roget Category	Target Word in Context
bass ⁴	lubina	FISH/INSECT	... fish as Pacific salmon and striped bass and...
bass ⁴	lubina	FISH/INSECT	... produce filets of smoked bass or sturgeon...
bass ⁷	bajo	MUSIC	... exciting jazz bass player since Ray Brown...
bass ⁷	bajo	MUSIC	... play bass because he doesn't have to solo...

Extracting Feature Vectors

“ If one examines the words in a book, one at a time as through an **opaque mask with a hole in it** one word wide, then it is obviously impossible to determine, one at a time, the meaning of the words. [...] But **if one lengthens the slit in the opaque mask**, until one can see not only the central word in question but also say N words on either side, then if N is large enough one can unambiguously decide the meaning of the central word. [...]

The practical question is : “*What minimum value of N will, at least in a tolerable fraction of cases, lead to the correct choice of meaning for the central word?*”

Nancy, Ide, and Veronis Jean. Word sense disambiguation: The state of the art. *Computational Linguistics* 24.1 (1998): 1-40.

Daniele Radicioni - TLN

feature vectors

- to **extract useful features from such a window**, a minimal amount of processing is first performed on the sentence containing the window.
 - this **processing** typically includes **part-of-speech (POS) tagging**, **lemmatization** or **stemming**, and in some cases **syntactic parsing** to reveal information such as head words and dependency relations.
 - context features relevant to the target word can then be extracted from this enriched input.
- a feature vector consisting of numeric or nominal values is used to encode this linguistic information as an input to most machine learning algorithms.

collocational vs. bag-of-words

- two classes of features are generally extracted: **collocational** features and **bag-of-words** features.
- a **collocation** is a word or phrase in a position-specific relationship to a target word (i.e., *exactly one word to the right, or exactly 4 words to the left, and so on*).

collocational features

- let us consider a case where we have to disambiguate the word *bass* in the following WSJ sentence:
 - *An electric guitar and bass player stand off to one side, not really part of the scene...*

collocational features

*An electric guitar and **bass** player stand off to one side, not really part of the scene...*

- example of a collocational feature-vector, extracted from a window of two words to the right and left of the target word, made up of the words themselves and their respective parts-of- speech, i.e.,

$[w_{i-2}, POS_{i-2}, w_{i-1}, POS_{i-1}, w_{i+1}, POS_{i+1}, w_{i+2}, POS_{i+2}]$

- would yield the following vector:

$[guitar, NN, and, CC, player, NN, stand, VB]$

bag-of-words approaches

- a bag-of-words means **an unordered set of words**, ignoring their exact position.
 - the simplest bag-of-words approach represents the context of a target word by a vector of features, each binary feature indicating **whether a vocabulary word w does or doesn't occur** in the context.

bag-of-words approaches

*An electric guitar and **bass** player stand off to one side, not really part of the scene...*

- for example a bag-of-words **vector** consisting of the 12 most frequent content words from a collection of bass sentences drawn from the WSJ corpus would have the following ordered word feature set:

*[fishing, big, sound, **player**, fly, rod, pound, double, runs, playing, **guitar**, band]*

- using these word features with a window size of 10, in the example would be represented by the following binary vector:

*[0,0,0, **1**,0,0,0,0,0,0, **1**,0]*

Daniele Radicioni - TLN

the Lesk Algorithm

- by far the most well-studied dictionary-based algorithm for sense disambiguation is the Lesk algorithm.

the Lesk Algorithm

```
1 function SimplifiedLesk(word,sentence)
2 returns best sense of word
3 best-sense  $\leftarrow$  most frequent sense for word
4 max-overlap  $\leftarrow$  0
5 context  $\leftarrow$  set of words in sentence
6 for all senses of word do
7   signature  $\leftarrow$  set of words in the gloss and examples of sense
8   overlap  $\leftarrow$  ComputeOverlap(signature,context)
9   if overlap > max-overlap then
10     max-overlap  $\leftarrow$  overlap
11     best-sense  $\leftarrow$  sense
12   end if
13 end for
14 return best-sense
```

the Lesk Algorithm

- as an example of the Lesk algorithm at work, consider disambiguating the word **bank** in the following context:
the bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

the Lesk Algorithm

the bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

- given the following two WordNet senses:

bank¹	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	“he cashed a check at the bank”, “that bank holds the mortgage on my home”
bank²	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	“they pulled the canoe up on the bank”, ...

the *Corpus* Lesk Algorithm

- the primary problem with either the original or simplified approaches, however, is that the **dictionary entries for the target words are short**, and may not provide enough chance of overlap with the context.
- one remedy is to **expand the list of words** used in the classifier to include words related to, but not contained in their individual sense definitions.
- the best solution, if any sense-tagged corpus data is available, is to add all the words in the labeled corpus sentences for a word sense into the signature for that sense (**Corpus Lesk algorithm**).

the *Corpus* Lesk Algorithm

- instead of just counting up the overlapping words, the Corpus Lesk algorithm also applies a weight to each overlapping word.
 - the weight is the inverse document frequency or IDF, a standard information-retrieval measure
- IDF measures how many different 'documents' (in this case glosses and examples) a word occurs in.
 - Since function words like *the*, *of*, *etc*, occur in many documents, their IDF is very low, while the IDF of content words is high.

the *Corpus* Lesk Algorithm

- the IDF_i score for a word i can be defined as

$$idf_i = \log \left(\frac{Ndoc}{nd_i} \right)$$

- where $Ndoc$ is the total number of 'documents' (glosses and examples) and nd_i is the number of these documents containing word i .

the *Corpus* Lesk Algorithm

- finally, it is possible to combine the Lesk and supervised approaches, by adding new Lesk-like bag-of-words features.
 - for example, the glosses and example sentences for the target sense in WordNet could be used to compute the supervised bag-of-words features instead of (or in addition to) the words in the context sentence for the sense.

example problem

- let us consider the three senses of the noun *ash* in WordNet, along with their definition.
 - *sense₁*: the residue that remains when something is burned ;
 - *sense₂*: any of various deciduous pinnate-leaved ornamental or timber trees of the genus *Fraxinus*;
 - *sense₃*: strong elastic wood of any of various ash trees; used for furniture and tool handles and sporting goods such as baseball bats.

example problem

- let us suppose we want to disambiguate the term *ash* occurring in the two contexts:
 - *context₁*: *The house was burnt to ashes while the owner returned.;*
 - *context₂*: *This table is made of ash wood.*

example problem

- *context₁*: The house was burnt to ashes while the owner returned;
 - *context₂*: This table is made of ash wood.
- using the number of words that the contexts have in common with the sense definitions:

	<i>s₁</i>	<i>s₂</i>	<i>s₃</i>
<i>c₁</i>	1	0	1
<i>c₂</i>	1	0	2

tools

- Find APIs and interfaces to WordNet at the URL
<https://wordnet.princeton.edu/related-projects>

testing corpus:

- SemCor: semantically annotated English corpus
- English corpus with semantically annotated texts. The semantic analysis was done manually with WordNet 1.6 senses (SemCor version 1.6) and later automatically mapped to WordNet 3.0 (SemCor version 3.0).
 - the corpus consists of 352 texts from Brown corpus.
- the corpus has also multi-word expressions (MWE) marked with underscore (_), e.g. manor_house.

Consegna

- Implementare l'algoritmo di Lesk (\neq usare implementazione esistente, e.g., in nltk...).
- 1. **Estrarre 50 frasi dal corpus SemCor** (corpus annotato con i synset di WN) e disambiguare (almeno) un sostantivo per frase. Calcolare l'accuratezza del sistema implementato sulla base dei sensi annotati in SemCor.
 - SemCor è disponibile all'URL
<http://web.eecs.umich.edu/~mihalcea/downloads.html>
- 2. **Randomizzare la selezione delle 50 frasi** e la selezione del termine da disambiguare, e restituire l'accuratezza media su (per esempio) 10 esecuzioni del programma.
- **[OPZIONALE]**: implementare corpus-Lesk algorithm utilizzando Sem-Cor.

Daniele Radicioni - TLN

