

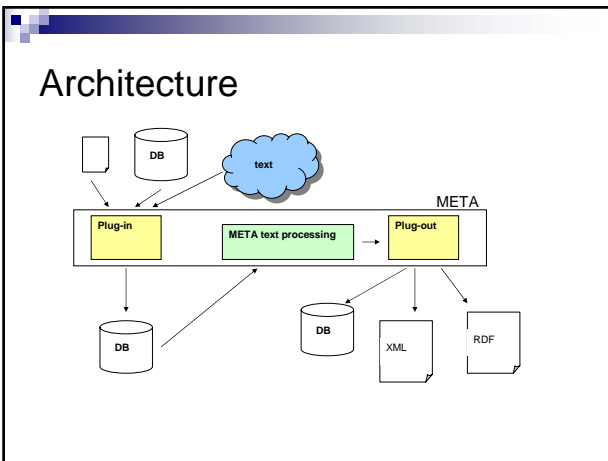
META
MultilanguagE
Text
Analyzer

Pierpaolo Basile
Marco Degemmis
Giovanni Semeraro
Pasquale Lops

LACAM - Knowledge Acquisition and Machine Learning Lab
Department of Computer Science - University of Bari

Overview

- META is a tool for text analysis which implements some NLP functionalities
- META provides the tools for semantic indexing and exploits WordNet as knowledge source in Word Sense Disambiguation processing



Plug-In

- The Plug-In is a component that is able to import data from different sources and inserts them in a database. Then, data are processed by META.
- In order to integrate this component into META it is necessary to:
 - develop a class that implements the interface `plugs.plugin.Plugin` and inserts it into package `plugs.plugin.pluginImpl.<new_package>`
- The developed Plug-In is able to import text from: PDF, DOC, HTML, OpenOffice, XLS, RTF, TXT, XML

META Text-processing

- Automatic language recognition
- Text normalization
- Tokenization
- Stop word elimination
- Stemming
- Text summarization (TF/IDF based)
- POS-tagging
 - Based on ACOPOST T3 (HMM - Hidden Markov Model)
- Entity recognize
- WSD (Word Sense Disambiguation)
 - The WSD algorithm was evaluated by SENSEVAL framework
- TF/IDF parametric on features

Plug-Out

- The Plug-Out is a component that is able to export META output in a different format.
- In order to integrate this component into META it is necessary to:
 - develop a class that implements the interface `plugs.plugin.Plugin` and inserts it into package `plugs.plugin.pluginImpl.<new_package>`
- The developed Plug-Out is able to export META output in:
 - Database, RDF, XML, TXT
 - different indexing models: Vector Space Model, LSI (Latent Semantic Indexing), ...

Future work

- Improve entity recognition through Machine Learning techniques
 - Currently, META includes an annotation tool for the development of tagged-corpus
- Exploit domain ontologies in the indexing process, in order to map entities to instances available in the ontologies

JIGSAW – WSD algorithm

- JIGSAW is a Word Sense Disambiguation algorithm that uses three different kinds of techniques to disambiguate nouns, verbs, adjectives and adverbs.
- Input:
 - $W = \{w_1, w_2, \dots, w_n\}$ is the set of all words in text which will be disambiguated
 - synVerb is the algorithm for disambiguating verbs
 - synNoun is the algorithm for disambiguating nouns
 - synAdjAdv is a Lesk-based algorithm for disambiguating adjectives and adverbs

JIGSAW – WSD algorithm

```
for each  $w_i \in W$ 
  if ( $w_i$  is a verb) execute synVerb on  $w_i$ 
for each  $w_i \in W$ 
  if ( $w_i$  is a noun) execute synNoun on  $w_i$ 
  else if ( $w_i$  is a adjective or  $w_i$  is a adverb)
    execute synAdjAdv su  $w_i$ 
```

- Input:
 - $W = \{w_1, w_2, \dots, w_n\}$ is the set of all words in text which will be disambiguated
 - synVerb is the algorithm for disambiguating verbs
 - synNoun is the algorithm for disambiguating nouns
 - synAdjAdv is a Lesk-based algorithm for disambiguating adjectives and adverbs

LB (Lesk-based) algorithm

- LB algorithm uses WordNet glosses to create a relation between grammatical categories and the context.
- This allows to create semantic relations with words in the context of the word to be disambiguated (*target word*)
- The algorithm measures the similarity between glosses of target words and glosses of context words. This operation is performed in 4 steps:
 - keep *target word* context
 - create the string (*glossContext*) which contains the glosses of words in the context
 - for each meaning of the *target word*, create a string (*glossTarget*) which contains the gloss of the meaning and all words related with *target word* and compute the similarity between *glossContext* and *glossTarget*
 - select the meaning for *target word* which has the max similarity wrt the *glossContext*

LB (Lesk-based) algorithm

I play basketball.

contextGloss: participate in games or sports We played hockey all afternoon Play cards Pele plays for Brazilian teams in many important matches act or have an effect in specified way...

targetGloss #1: basketball game a game played on a court by two opposing teams of five players points are scored by throwing the basketball through an elevated horizontal hoop *court game athletic game sports athletic diversion recreation activity game basketball shot rebound fastbreak dribbling tip-off tap-off basketball play half*

targetGloss #2: an inflated ball used in playing basketball ball game *equipment instrumentality object whole*

Algorithm for disambiguating nouns

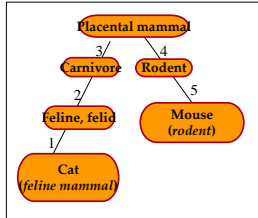
- The algorithm is based on observation that when 2 words have more than one meaning, the common minimal meaning (*Most Specific Subsumer - MSS* - in the WordNet hierarchy) supplies information on the most appropriate meaning for each word.
- Let W be the set of nouns in a sentence and S the set of all possible meanings, the goal is to define a function φ (between 0 and 1) that computes the probability of the meaning $s_j \in S$ for the word $w_i \in W$.
- For each word w_i in W , the algorithm selects the most appropriate meaning $s_{i \max}$

$$s_{i \max} = \arg \max_j \varphi(s_{ij})$$

Algorithm for disambiguating nouns

- In order to compute ϕ , the MSS, the distance between words and the similarity are exploited.

$$sim_{ab} = \max \left(-\log \left(\frac{N_p}{2D} \right) \right) \quad \text{Leacock-Chodorow}$$



$N_p = 5$ and
 $D = 16$ (in WordNet 1.7.1)

$$sim(cat, mouse) = -\log(5/32) = 0.806$$

Most Specific Subsumer
PLACENTAL MAMMAL

Algorithm for disambiguating nouns

Let d be the noun to be disambiguated:

```

for j=1,...|CONTEXT|
  v[j] ← sim(d, wj) * G(position(d) - position(wj))
  c[j] ← most_specific_subsumer(d, wj)
  for k=1,...|S|
    if (c[j] is hypernym of k-th meaning of d)
      support[k] ← support[k] + v[j]
  normalization ← normalization + v[j]
end for
posmax ← 1
maxφ ← -MAX_DOUBLE
for k=1,...|S|
  if (normalization!=0)
    φ[k] ← α * support[k] / normalization + β * R(k)
  else
    φ[k] ← α / |S| + β * R(k)
  if (φ[k] > maxφ)
    maxφ ← φ[k]
    posmax ← k
  end if
end for
best meaning of d is equal to sposmax ∈ S
  
```

Algorithm for disambiguating nouns

Let d be the noun to be disambiguated:

```

for j=1,...|CONTEXT|
  v[j] ← sim(d, wj) * G(position(d) - position(wj))
  c[j] ← most_specific_subsumer(d, wj)
  for k=1,...|S|
    if (normalization!=0)
      φ[k] ← α * support[k] / normalization + β * R(k)
    else
      φ[k] ← α / |S| + β * R(k)
    if (φ[k] > maxφ)
      maxφ ← φ[k]
      posmax ← k
    end if
  end for
best meaning of d is equal to sposmax ∈ S
  
```

$v[j] \leftarrow \text{sim}(d, w_j) * G(\text{position}(d) - \text{position}(w_j))$
 • It computes the similarity between j -th noun in W and d .
 • G is the Gaussian function which takes into account the difference between the noun's positions in the text

Verbs' algorithm

- Each noun w_i in N is compared with each noun w_j in $nouns(w,k)$ by computing a similarity value between w_i and w_j

$$sim(w_i, w_j) = \frac{-\ln\left(\frac{N_p}{2D}\right)}{-\ln\left(\frac{1}{2D}\right)}$$

Verbs' algorithm

- The algorithm computes the similarity for each pair of nouns and then, for the i -th noun w_i in N , computes the following value: $max_i = \max\{sim(w_i, w_j) \mid j = 1, 2, \dots, m\}$, where m is $|nouns(w,k)|$.
- The function φ is computed by the following equation:

$$\varphi(v, k) = R(k) * \sum_i \frac{G(d_i) * max_i}{\sum_i G(d_i)}$$

- $R(k)$ is the same function in the nouns' algorithm but has a different coefficient (0.9)

Evaluation

- All algorithms has been evaluated with SENSEVAL framework

| Algorithm | Precision |
|-----------------|-----------|
| LB (nouns) | 0,246 |
| LB (verbs) | 0,295 |
| LB (adjectives) | 0,403 |
| Verbs | 0,405 |
| Nouns | 0,319 |

Evaluation

- All algorithms has been evaluated with SENSEVAL

| Algorithm | Precision |
|------------|-----------|
| LB (nouns) | 0,246 |

JIGSAW precision
sample task = 0,376
all word task = 0.52
