Distributional similarity

Introduction and implementation

Tim Van de Cruys

University of Cambridge

INCAS³ WORKSHOP
databases and annotating

Wednesday 10 August, 2011

Distributional similarity

- Most work on semantic similarity relies on the DISTRIBUTIONAL HYPOTHESIS (Harris 1954)
  - Take a word and its contexts:
    - tasty *tnassiorc*
    - greasy *tnassiorc*
    - *tnassiorc* with butter
    - *tnassiorc* for breakfast
- By looking at a word's context, one can infer its meaning

Distributional similarity

- Most work on semantic similarity relies on the DISTRIBUTIONAL HYPOTHESIS (Harris 1954)

  - Take a word and its contexts:
    - tasty *tnassiorc*
    - greasy *tnassiorc*
    - *tnassiorc* with butter
    - *tnassiorc* for breakfast

  $\Rightarrow$ **FOOD**

- By looking at a word's context, one can infer its meaning

Distributional similarity

- Most work on semantic similarity relies on the DISTRIBUTIONAL HYPOTHESIS (Harris 1954)

- Take a word and its contexts:
  - tasty *tnassiorc*
  - greasy *tnassiorc*
  - *tnassiorc* with butter
  - *tnassiorc* for breakfast

- By looking at a word's context, one can infer its meaning

Matrix

- Capture co-occurrence frequencies of two entities

Matrix

- Capture co-occurrence frequencies of two entities

	rouge	délicieux	rapide	d'occasion
pomme	2	1	0	0
vin	2	2	0	0
voiture	1	0	1	2
camion	1	0	1	1

Matrix

- Capture co-occurrence frequencies of two entities

	rouge	délicieux	rapide	d'occasion
pomme	7	9	0	0
vin	12	6	0	0
voiture	7	0	8	4
camion	2	0	3	4

Matrix

- Capture co-occurrence frequencies of two entities

	rouge	délicieux	rapide	d'occasion
pomme	56	98	0	0
vin	44	34	0	0
voiture	23	0	31	39
camion	4	0	18	29

Matrix

- Capture co-occurrence frequencies of two entities

	rouge	délicieux	rapide	d'occasion
pomme	728	592	1	0
vin	1035	437	0	2
voiture	392	0	487	370
camion	104	0	393	293

Similarity calculation

Cosine

- $cos(\overrightarrow{x}, \overrightarrow{y}) = \frac{\overrightarrow{x} \cdot \overrightarrow{y}}{|\overrightarrow{x}||\overrightarrow{y}|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}}$
- Examples:
  - $cos(pomme, vin) = .96$
  - $cos(pomme, voiture) = .42$

- Other possibilities:
  - set-theoretic measures
    - Dice
    - Jaccard
  - probabilistic measures
    - Kullback-Leibler divergence
    - Jensen-Shannon divergence

Different kinds of context

- Three different word space models based on context:
  - document-based model (nouns $\times$ documents)
  - window-based model (nouns $\times$ context words)
  - syntax-based model (nouns $\times$ dependency relations)
- Each model with plethora of parameters!
  - document size, window size, type of dependency relations
  - weighting function
  - $\pm$ dimensionality reduction

Document-based model

- Plain word corpus
- Matrix contains the number of times a word appears in a particular document (web page, newspaper article, wikipedia entry, ...)
- Parameters:
  - **document size**: full document, paragraph, ...
  - **weighting**: TF/IDF, logarithmic, ...

	doc1	doc2	doc3	doc4
word1				
word2				
word3				
word4				

Window-based model

- Plain word corpus
- Matrix contains the number of times a word appears in a particular window around a (small window, sentence, paragraph, ...)
- Parameters:
  - **dependency relations**: which ones?
  - **weighting**: TF/IDF, pointwise mutual information, logarithmic, ...

	word1	word2	word3	word4
word1				
word2				
word3				
word4				

Syntax-based model

- Syntactically annotated (automatically parsed) corpus
- Matrix contains the number of times a word appears with a particular syntactic (dependency) feature (apple: direct object of *eat*, bomb: subject of *explode*, ...)
- Parameters:
  - **window size**: *n* words (left/right), sentence, paragraph ...
  - **weighting**: TF/IDF, pointwise mutual information, logarithmic, ...

	dep1	dep2	dep3	dep4
word1				
word2				
word3				
word4				

Different kinds of semantic similarity

- **'tight', synonym-like similarity**: (near-)synonymous or (co-)hyponymous
- **loosely related, topical similarity**: more loose relationships, such as association and meronymy

Different kinds of semantic similarity

- **'tight', synonym-like similarity**: (near-)synonymous or (co-)hyponymous
- **loosely related, topical similarity**: more loose relationships, such as association and meronymy

Example

- **médecin** 'doctor': *docteur* 'doctor', *médecin de famille* 'family doctor', *chirurgien* 'surgeon', *spécialiste* 'specialist', *dermatologue* 'dermatologist', *gynécologue* 'gynaecologist'
- **médecin** 'doctor': *malade* 'patient', *maladie* 'disease', *diagnostic* 'diagnosis, *traitement* 'treatment, *hôpital* 'hospital', *stéthoscope* 'stethoscope'

Relation context – similarity

- Different context leads to different kind of similarity
- Syntax, small window $\leftrightarrow$ large window, documents
- The former models induce **tight, synonymous similarity**
- The latter models induce **topical relatedness**

Relation context – similarity

- Different context leads to different kind of similarity
- Syntax, small window $\leftrightarrow$ large window, documents
- The former models induce **tight, synonymous similarity**
- The latter models induce **topical relatedness**

Evaluation

- Syntax-based model scores best when evaluated according to Wordnet similarity measures (CORNETTO)
- Large window and document-based do not score well on Wordnet similarity, but do score on Wordnet domain evaluation

Weighting

- How salient is a word within a document/window/dependency relation?
- *de voetballer* **is** $\leftrightarrow$ *de voetballer* **scoort**
- Local vs. global weighting:
  - local: only based on information in matrix cell (e.g. logarithmic weighting)
  - global: based on global instances/feature frequencies (probabilities)

local weighting: logarithmic weighting

- $f_{i,j} = 1 + log(f_{i,j})$
- smooths high frequency data

global weighting: pointwise mutual information

- $pmi(i,j) = log(\frac{p(i,j)}{p(i)p(j)})$
- Compare joint probability $p(i,j)$ with marginal probabilities $p(i)$ and $p(j)$
- higher value if i and j occur together more often than one would expect given their independence

Dimensionality reduction

- reduce large number of features to limited number of 'semantic dimensions'
- useful for topical similarity (dimensions represent topics)
- latent semantic analysis, latent dirichlet allocation, non-negative matrix factorization

Python framework

1. Preprocessing
2. Determination of instances and features
3. Matrix construction
4. Similarity computations

Preprocessing

- Convert corpus to proper format/usable form
  - convert to raw text
  - syntactic parsing
  - extraction of dependency triples
  - storage: plain text or MySQL database

Read in corpus

- Parent **Corpusreader** class
- Child classes for specific corpus formats

Determination of instances and features

- instances (words) and features (window-based words, dependency features) need to be determined beforehand for proper matrix construction
  - ‘dry run’ on corpus
  - *or* initial sort for most frequent instances/features
  - *or* complete construction with pruning step

Matrix construction

- Matrices tend to be very sparse (dependency-based: $<1\%$ zeros)
- sparse matrix implementation: lists-of-hashes (lists-of-dicts)
- word strings mapped to integers (with translation dict)
- alternative: scientific libraries (Numpy/Scipy)

Matrix construction

- Parent **Matrix** class with
  - initial determination
  - fill options
  - weighting functions
  - normalization
- Child classes for specific models (document-based, window-based, dependency-based)

Similarity computations

- different similarity functions operating on vectors of matrix (cosine, KL-divergence, . . .)
- functions of Matrix itself
- Numpy/Scipy