Expressing Implicit Semantic Relations without Supervision

-- Peter D. Turney

Presented by Lu Zhang
Saarland University

Outline

- Motivation
- Introduction to Pertinence
- The Pertinence Algorithm
- Evaluation
- Conclusion

Motivation

- Input: a set of word pairs $W = \{X_1 : Y_1, ..., X_n : Y_n\}$
- Output: ranked lists of patterns $\langle P_1, ..., P_m \rangle$ for each input pair
- Example
  - X=ostrich , Y= bird
  - The two highest output patterns: “X is the largest Y” and “Y such as the X”
- Main challenge: How can we find a way of ranking the patterns , so that patterns like “Y such as the X” are highly ranked?

Outline

- Motivation
- Introduction to Pertinence
- The Pertinence Algorithm
- Evaluation
- Conclusion

Pertinence

- What is pertinence?
  - The pertinence of a pattern P_i for a word pair *X* :*Y* is the expected relational similarity between the given pair and typical pairs for P_i.

- How to calculate pertinence?

$$pertinence\left(X_j : Y_j, P_i\right) = \sum_{k=1}^{n} p\left(X_k : Y_k \middle| P_i\right) \cdot sim_r\left(X_j : Y_j, X_k : Y_k\right)$$

Pertinence

- To estimate $p(X_k : Y_k | P_i)$, we first estimate $p(P_i | X_k : Y_k)$:

$$p(P_i | X_k : Y_k) = f_{k,i} \Big/ \sum_{j=1}^{m} f_{k,j}$$

  - where $f_{k,i}$ is the number of occurrences in a corpus of the word pair $X_k : Y_k$ with the pattern P_i .

Pertinence

- Applying Bayes's Theorem:

$$p\left(X_k : Y_k \middle| P_i\right) = \frac{p\left(X_k : Y_k\right) \cdot p\left(P_i \middle| X_k : Y_k\right)}{\sum_{j=1}^{n} p\left(X_j : Y_j\right) \cdot p\left(P_i \middle| X_j : Y_j\right)}$$

- Assuming $p\left(X_n : Y_n\right) = 1/n$ for all pairs in W:

$$p\left(X_k : Y_k \middle| P_i\right) = p\left(P_i \middle| X_k : Y_k\right) \Big/ \sum_{j=1}^{n} p\left(P_i \middle| X_j : Y_j\right)$$

Outline

- Motivation
- Introduction to Pertinence
- The Pertinence Algorithm
- Evaluation
- Conclusion

The Pertinence Algorithm

- Input: a set of word pairs $W = \{X_1 : Y_1, ..., X_n : Y_n\}$
- Output: ranked lists of patterns $\langle P_1, ..., P_m \rangle$ for each input pair
- An algorithm similar to the LRA (Latent Relational Analysis) algorithm, with several changes to support the calculation of pertinence
- The first unsupervised learning algorithm that can find patterns for semantic relations

The Pertinence Algorithm

- Step1: Find phrases
- Step2: Generate patterns
- Step3: Count pair frequency
- Step4: Map pairs to rows
- Step5: Map patterns to columns
- Step6: Build a sparse matrix
- Step7: Calculate entropy
- Step8: Apply SVD
- Step9: Calculate cosines
- Step10: Calculate conditional probabilities
- Step11: Calculate pertinence

Find phrases

- For each pair $X_i : Y_i$, make a list of phrases in the corpus that contain the pair.
  - Make one list of phrase that begins with X_i and end with Y_i and a second list for the opposite order.
- Each phrase must have one to three intervening words between X_i and Y_i .
- The first and last words in the phrase do not need to exactly match X_i and Y_i .

Generate Patterns

- For each list of phrases ,generate a list of patterns.
  - Replace the first word in each phrase with the generic marker “X” and replace the last word with “Y” .
  - The intervening words in each phrase may be either left as they are or replaced with the wildcard “*” .
- Example
  - phrase: “carpenter nails the wood”
  - patterns: “X nails the Y” , “X nails * Y” , “X * the Y” , “X * * Y”

Build a frequency matrix

- Count pair frequency
  - The number of lists from the preceding step that contain the given pattern
- Map pairs to rows
  - To build a matrix X , for each pair $X_i : Y_i$, create a row for $X_i : Y_i$ and another row for $Y_i : X_i$.

Build a frequency matrix

- Map patterns to columns
  - For each unique pattern of the form “X … Y” from step2 , create a column for the original pattern “X … Y” and another column for the pattern “Y … X”.
  - To keep the matrix manageable , drop all patterns with a frequency less than ten.
- Build a sparse matrix
  - The value for the cell in row i and column j is the pattern frequency of the j-th pattern for the i-th word pair.

The Pertinence Algorithm

- Calculate entropy
  - Apply log and entropy transformation to the matrix X , each cell is replaced with its logarithm and multiplied by a weight based on the negative entropy of the corresponding column vector in the matrix.
- Apply SVD (Singular Value Decomposition)
  - Decompose X into $U\Sigma V^T$, where U and V are in column orthogonal form and Σ is a diagonal matrix of singular value.
  - SVD is used to reduce noise and compensate for sparseness.

The Pertinence Algorithm

- Calculate cosines
  - $sim_r\left(X_j : Y_j, X_k : Y_k\right)$ is given by the cosine of the angle between their corresponding row vectors in the matrix $U_k \Sigma_k V_k^T$.
  - All of the cosines can be derived from the matrix $U_k \Sigma_k \left(U_k \Sigma_k\right)^T$.

- Calculate conditional probability
  - $p\left(X_i : Y_i \middle| P_j\right)$ should be calculated with the raw frequency data in the matrix from step6 .

The Pertinence Algorithm

- Calculate pertinence
  - Calculate $pertinence\left(X_i : Y_i, P_j\right)$ for every row $X_i : Y_i$ and every column P_j for which $p\left(X_i : Y_i \middle| P_j\right) > 0$.
  - For each pair $X_i : Y_i$, output two ranked lists , one for patterns of the form “X … Y” and another for the form “Y … X”.
  - The patterns in both lists are sorted in order of decreasing pertinence to $X_i : Y_i$.

Outline

- Motivation
- Introduction to Pertinence
- The Pertinence Algorithm
- Evaluation
- Conclusion

Evaluation

- Experiment with Word Analogies
  - Use 374 college-level multiple-choice word analogies , taken from the SAT test
  - Task: Find the choice that most analogous to the stem
  - Evaluate each choice by taking the intersection of its patterns with the stem's pattern.

Experiment with Word Analogies

	Word pair	Best shared pattern	Score
1.	ostrich:bird		
(a)	**lion:cat**	***"Y such as the X"***	**1.0**
(b)	goose:flock	*"X * * breeding Y"*	43.5
(c)	ewe:sheep	*"X are the only Y"*	13.5
(d)	cub:bear	*"Y are called X"*	29.0
(e)	primate:monkey	*"Y is the * X"*	80.0
2.	traffic:street		
(a)	ship:gangplank	*"X * down the Y"*	53.0
(b)	crop:harvest	*"X * adjacent * Y"*	248.0
(c)	car:garage	*"X * a residential Y"*	63.0
(d)	pedestrians:feet	*"Y * accommodate X"*	23.0
(e)	**water:riverbed**	***"Y that carry X"***	**17.0**
3.	locomotive:train		
(a)	horse:saddle	*"X carrying * Y"*	82.0
(b)	**tractor:plow**	***"X pulled * Y"***	**7.0**
(c)	rudder:rowboat	*"Y * X"*	319.0
(d)	camel:desert	*"Y with two X"*	43.0
(e)	gasoline:automobile	*"Y powered * * X"*	5.0

Table 1. Three examples of SAT questions.

Experiment with Word Analogies

	Algorithm	Prec.	Rec.	F
1	pertinence (Step 11)	55.7	53.5	54.6
2	log and entropy matrix (Step 7)	43.5	41.7	42.6
3	TF = f, IDF = log((N-n)/n)	43.2	41.4	42.3
4	TF = log(f+1), IDF = log(N/n)	42.9	41.2	42.0
5	TF = f, IDF = log(N/n)	42.9	41.2	42.0
6	TF = log(f+1), IDF = log((N-n)/n)	42.3	40.6	41.4
7	TF = 1.0, IDF = 1/n	41.5	39.8	40.6
8	TF = f, IDF = 1/n	41.5	39.8	40.6
9	TF = 0.5 + 0.5 * (f/F), IDF = log(N/n)	41.5	39.8	40.6
10	TF = log(f+1), IDF = 1/n	41.2	39.6	40.4
11	p(X:Y|P) (Step 10)	39.8	38.2	39.0
12	SVD matrix (Step 8)	35.9	34.5	35.2
13	random	27.0	25.9	26.4
14	TF = 1/f, IDF = 1.0	26.7	25.7	26.2
15	TF = f, IDF = 1.0 (Step 6)	18.1	17.4	17.7
16	Turney (2005)	56.8	56.1	56.4
17	Turney and Littman (2005)	47.7	47.1	47.4
18	Veale (2004)	42.8	42.8	42.8

Table 4. Performance of various algorithms on SAT.

Evaluation

- Experiment with Noun-Modifiers
  - Task: To classify noun-modifier pairs , such as “flu virus”, according to the semantic relation between the head noun (virus) and the modifier (flu)
  - Input: A set of 600 manually labeled noun-modifier pairs
  - There are five classes of labels

Class name	Prec.	Rec.	F	Class size
causality	37.3	36.0	36.7	86
participant	61.1	64.4	62.7	260
quality	49.3	50.7	50.0	146
spatial	43.9	32.7	37.5	56
temporality	64.7	63.5	64.1	52
all	51.3	49.5	50.2	600

Table 5. Performance on noun-modifiers.

Experiment with Noun-Modifiers

	Algorithm	Prec.	Rec.	F
1	pertinence (Step 11)	51.3	49.5	50.2
2	TF = log(f+1), IDF = 1/n	37.4	36.5	36.9
3	TF = log(f+1), IDF = log(N/n)	36.5	36.0	36.2
4	TF = log(f+1), IDF = log((N-n)/n)	36.0	35.4	35.7
5	TF = f, IDF = log((N-n)/n)	36.0	35.3	35.6
6	SVD matrix (Step 8)	43.9	33.4	34.8
7	TF = f, IDF = 1/n	35.4	33.6	34.3
8	log and entropy matrix (Step 7)	35.6	33.3	34.1
9	TF = f, IDF = log(N/n)	34.1	31.4	32.2
10	TF = 0.5 + 0.5 * (f/F), IDF = log(N/n)	31.9	31.7	31.6
11	p(X:Y|P) (Step 10)	31.8	30.8	31.2
12	TF = 1.0, IDF = 1/n	29.2	28.8	28.7
13	random	19.4	19.3	19.2
14	TF = 1/f, IDF = 1.0	20.3	20.7	19.2
15	TF = f, IDF = 1.0 (Step 6)	12.8	19.7	8.0
16	Turney (2005)	55.9	53.6	54.6
17	Turney and Littman (2005)	43.4	43.1	43.2

Table 7. Performance on noun-modifiers.

Outline

- Motivation
- Introduction to Pertinence
- The Pertinence Algorithm
- Evaluation
- Conclusion

Conclusion

- Problem: Find a way to make ranked lists of patterns for each input pair.
- The idea of pertinence allows us to take an opaque measure of relational similarity and use it to find patterns that express the implicit semantic relations between two words.
- There’s still room for improvement of the algorithm , such as adding part-of-speech tagging.

References

- Turney, Peter D. Expressing Implicit Semantic Relations without Supervision. In *Proceedings 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL-06)*, pages pp. 313-320, Sydney, Australia. 2006

- Peter D. Turney. Similarity of Semantic Relations. In *Computational Linguistics Journal*. Volume 32, Issue 3. pp. 379-416. NRC 48775. September 2006.

- G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd Edit., The Johns Hopkins University Press, Baltimore, MD, 1996.

Thank you !