Clustering Word Pairs to Answer Analogy Questions

Ergun Biçici and Deniz Yuret

http://www.ku.edu.tr

June 22, 2006

TAINN 2006 Presentation

Today's Talk

- Analogy Questions and Introduction
- Approach
- Word pairs data from text, dataset generation
- Clustering Scoring Algorithm
- Experiments with SAT Questions
- Summary

Analogy Questions

- Analogy identification is important in question answering.
- In Aristotelian format: *hand:palm::foot:sole* (same semantic relations between word pairs)
- Analogy derivation, analogical reasoning, and similarity judgments take central role in reasoning.
- Many proglems of NLP are related: question answering, information extraction, word sense disambiguation

Our Work

- Observation: Semantically related word pairs might be clustered closely in the vector space model.
- We cast the problem of solving word analogy questions as an instance of learning clusterings of data
- We devise a heuristic approach to combine the results of different clusterings for the purpose of distinctly separating word pair semantics
- We answer SAT-type word similarity questions using our technique.

Approach

Assumption: Semantic relations are determined by both the word pairs and the syntactic patterns that they are observed with.

- Observe: $[ship\ \texttt{made of}\ gold]$ vs. $[ship\ \texttt{carrying}\ gold]$.
- We used the dataset from [3] derived by using Waterloo MultiText System [1] as the search engine.
- Word pairs are represented by the syntactic patterns that are observable between the words in the pair given a large corpus.
- The dataset is smoothed by mapping its feature vectors into a lower dimensional space using SVD.

Word pairs data from text, dataset generation

Input: Set of word pairs, WP, that we are interested. Each word pair, $wp \in WP$, is represented as $w_1 : w_2$ where w_1 and w_2 are the two words in the pair.

Steps involved:

1. **Identify alternates (wp):**
   $\forall wp \in WP$,
   $$\textbf{wp} = \bigcup w_1' : w_2',$$
   where w_1' is any one of the top 10 similar senses of the word w_1.

Steps Involved

1. *Identify alternates*
2. **Select alternates:** $\forall wp' \in$ **wp**,
   - Query the search engine for patterns of the form $[w'_1 * * * w'_2]$, where $wp' = w'_1 : w'_2$.
   - **wp**$'$ = The top 3 most frequent alternate word pairs.
   - **wp**$' =$ **wp**$' \bigcup wp$, where wp is the original word pair.

Steps Involved

1. *Identify alternates*
2. *Select alternates*
3. **Find patterns:** $\forall wp \in$ **WP**, where **WP** $= \bigcup$ **wp**$'$,
   - Query the search engine to find patterns of the form: $[w_1 * * * w_2]$, $[w_1 * *w_2]$, or $[w_1 * w_2]$.
   - Each $*$ in the pattern can match a word from the corpus.
   - Select the top 4000 patterns.

Steps Involved

1. *Identify alternates*
2. *Select alternates*
3. *Find patterns*
4. **Generate a matrix:** $\forall wp \in$ **WP**,
   - Create a row and for each pattern $w_1\ P\ w_2$,.
   - Create a column for $w_1\ P\ w_2$ and another one for $w_2\ P\ w_1$ (8000 columns).
   - The final matrix, X, $X(i,j) =$ *frequency of the* j*th pattern that contain* i*th word pair.*

Steps Involved

1. *Identify alternates*
2. *Select alternates*
3. *Find patterns*
4. *Generate a matrix*
5. **Apply SVD:** X is smoothed by mapping its feature set to a lower dimensional space using SVD [2] by choosing the largest 300 singular values. This reduces the number of columns to 300 instead of 8000. Let this new matrix be X_{300}.

Steps Involved

1. *Identify alternates*
2. *Select alternates*
3. *Find patterns*
4. *Generate a matrix*
5. *Apply SVD*
6. **Apply clustering:** Apply k-means, and spectral clustering on X_{300}. This provides us with different clusterings (i.e. allocation of word pairs into disjoint clusters).

Steps Involved

1. *Identify alternates*
2. *Select alternates*
3. *Find patterns*
4. *Generate a matrix*
5. *Apply SVD*
6. *Apply clustering*
7. **Apply scoring function:** The resulting clusterings are scored and combined to answer analogy questions and to pick the correct answer from a given set of choices.

Clustering Scoring Algorithm

Experimented with k values $2^1, 2^2, \ldots, 2^8$. Each *clustering* is an allocation of points to different clusters based on k.

```
cq = clusters(qwp);
for cluster = 0; cluster < numClusterings; cluster++ do
        ns = 0;                 /* Number of distinct answers in the same cluster as the question */
        foreach awp ∈ AWP do
                ca = clusters(awp);
                if cq[cluster] == ca[cluster] then
                        ns++;
                end
        end
        foreach awp ∈ AWP do
                ca = clusters(awp);
                if cq[cluster] == ca[cluster] then
                        score[awp] = score[awp] + numofClusters[cluster]/ns;
                end
        end
end
choice = max(score);
```

Algorithm 1: Clustering Scoring Algorithm.

Experiments with SAT Questions

- Tested with 374 college-level multiple-choice SAT analogy questions.
- 8128 word pairs: $(374 \times 6 \times 4)-$ no alternate cases $-$ word pairs not present in any pattern.
- The average human score is 57%.
- Turney [3] reports a performance of 56% using latent relational analysis.

Results with SAT Questions

Alternates	k-means	Spectral with local scaling
none	41.23%	35.72%
question	44.01%	34.87%
answer	39.95%	35.45%
both	38.50%	32.89%

Performance of clustering methods in answering word analogy questions. The first column shows whether thesaurus based alternate word pairs have been used for the question and answer pairs.

Summary

- We present an analysis of clustering algorithms' performance on answering word similarity questions.
- The dataset we have is based on word pairs and their occurrence frequencies in some common sytactic patterns.
- We cast the problem of solving word analogy questions as an instance of learning clusterings of data.
- We devise a heuristic approach to combine the results of different clusterings.
- We answer SAT-type word similarity questions.
- We observe that semantic relations between word pairs may be distinguished by using clustering techniques.

References

[1] Charles L. A. Clarke, G. V. Cormack, and F. J. Burkowski. An algebra for structured text search and a framework for its implementation. *The Computer Journal*, 38(1):43–56, 1995.

[2] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics, 1997.

[3] Peter Turney. Measuring semantic similarity by latent relational analysis. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1136–1141, Aug 2005.

Thank you!

Acknowledgments: We acknowledge the generous allowance of Peter Turney from NRCC, Canada, for access to the dataset and Waterloo MultiText System.