Automatic analysis of semantic similarity in comparable text through syntactic tree matching

Erwin Marsi, Emiel Krahmer

TiCC, Tilburg University

Monday August 23, 2010

Introduction

- Natural languages allow us to express essentially the same underlying meaning as many alternative surface forms
- In other words, there are often many similar ways to say the same thing
- This poses a problem for many NLP applications

Example: automatic summarization

- automatic summarizers typically rank sentences according to their informativity and then extract the top n sentences
- sentences are treated as independent, but typically they are not: may have substantial semantic overlap
- results in unintended redundancy in the summaries
- this is particularly problematic in the case of multi-document summarization, where sentences extracted from related documents are expected to be similar (Radev and McKeown, 1998).
- Therefore, if *semantic similarity* between sentences could be detected automatically, this would help to avoid redundancy in summaries.
- Similar arguments can be made for many other NLP applications

Analysis of semantic similarity

- In addition to *detecting* semantic similarity, we can ask *to what extent* two expressions share meaning, e.g.,
  - the meaning of one sentence can be fully contained in that of another, or
  - the meaning of one sentence can overlap only partly with that, of another
  - ...
- Can play an important role in NLP applications

Example: multi-document summarization

- Idea: avoid redundancy by merging similar sentences
- *Sentence fusion*: a new sentence is generated that conveys all common information from both sentences without introducing redundancy (Barzilay and McKeown, 2005)
- Requires analysis of semantic similaritys

Approaches to analysis of semantic similarity

- From shallow approaches relying to string matching
  - cannot capture less obvious cases such as synonyms or syntactic paraphrasing
- To deep semantic analysis (representation & inference)
  - suffers from coverage and robustness problems
- Approach proposed here:
  - aligning syntax trees, where each node is matched to the most similar node in the other tree (if any)
  - label alignments according to the type of similarity relation that holds between the aligned phrases

Outline of talk

- Introduction
- Problem statement
- Corpus data
- Graph matcher
- Experiments
- Conclusion

Tree alignment

- Alignment:
  - Given a pair of comparable sentences
  - and their corresponding source and target parse trees
  - align each node from the source tree to the most similar node in the target tree
- Nodes can be aligned to at most one other node
- Nodes may remain unaligned
- See paper for a formal description

Relation labeling

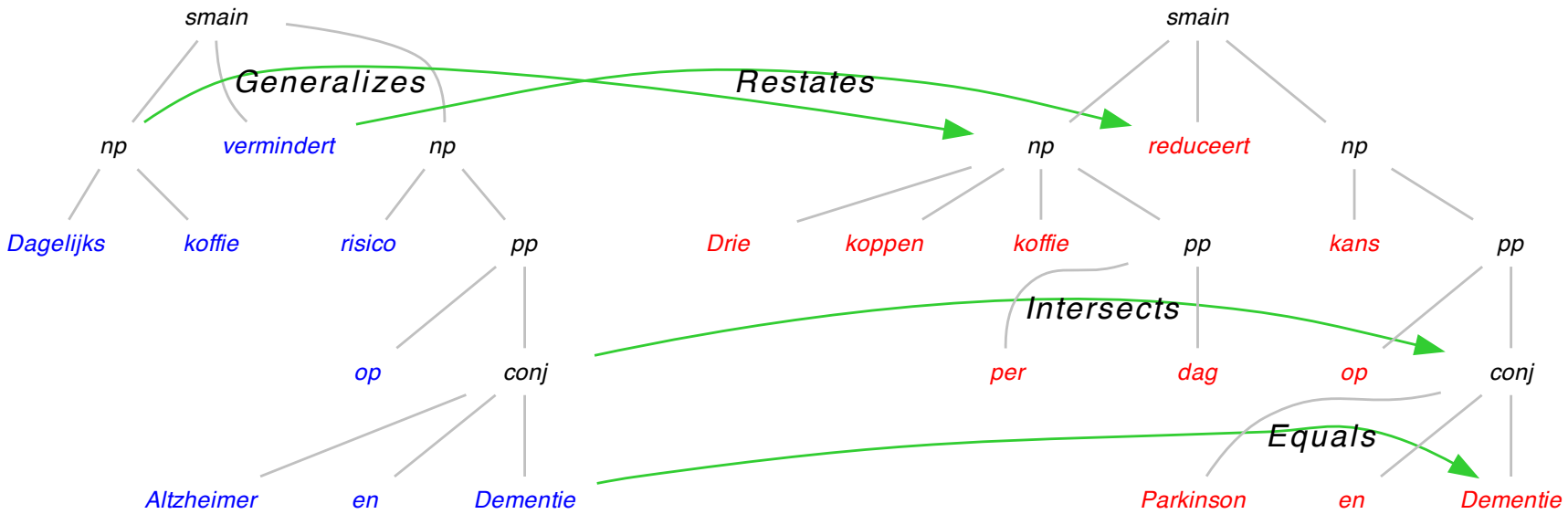
- Each alignment is labeled according to one of five mutually exclusive similarity relations:
  1. **equals**: lower-cased source text and lower-cased target text are identical – example: *Dementia* equals *Dementia*;
  2. **restates**: source text is a proper paraphrase of target text – example: *diminishes* restates *reduces*;
  3. **generalizes**: source text is more general than than target text – example: *daily coffee* generalizes *three cups of coffee a day*;
  4. **specifies**: source text is more specific than target text – example: *three cups of coffee a day* specifies *dailly coffee*;
  5. **intersects**: source and target texts share meaning, but each also contains unique information not expressed in the other – example: *Alzheimer and Dementia* intersects *Parkinson and Dementia*.
- See paper for details

Example: comparable sentences

(1) a. Dagelijks koffie vermindert risico op Alzheimer en
Daily coffee diminishes risk on Alzheimer and
Dementie.
Dementia.

b. Drie koppen koffie per dag reduceert kans op
Three cups coffee a day reduces chance on
Parkinson en Dementie.
Parkinson and Dementia.

Example: alignment and labeling



Goal

- A method for automatic analysis of semantic similarity through
- alignment of the syntax trees for comparable sentences
- and labeling of the alignment relation

Daeso corpus

- a monolingual treebank of parallel/comparable Dutch text
- over 1M words, about half of it manually aligned
- alignments at the level of sentences and syntactic nodes
- node alignments also labeled according to five similarity relations

Text material

- Corpus segments ranging from parallel to comparable text
  1. book translation
  2. autocue-subtitle from TV news
  3. headlines from news sites
  4. answers from QA system
  5. news from the two major news agencies in The Netherlands
- In this work we only use the manually annotated part of the news segment
  - 8,248 pairs of sentences
  - 162,361 tokens (ignoring punctuation)

Corpus construction

1. tokenization and sentence splitting
2. alignment of similar articles (partly manual)
3. sentence alignment (partly manual)
4. parsing with Alpino dependency parser for Dutch (Bouma et al., 2001)
5. tree alignment and labeling (mostly manual)

Memory-based graph matcher

- input: pair of syntactic trees T_s and T_t
- output: labeled matching of nodes T_s to nodes of T_t
- alignment and labeling are performed simultanously
- as a combination of
  - exhaustive pairwise classification using a supervised machine learning algorithm
  - followed by global optimization of the alignments using a combinatorial optimization algorithm

Step 1: Feature Extraction

- For each possible pairing of a source node n_s and a target node n_t, create an instance consisting of feature values extracted from the input trees.
- Features can represent
  - properties of individual nodes, e.g., source node is NP
  - relations between nodes, e.g., nodes have same POS

Step 2: Classification

- for each instance, a suprvised classifier predicts a class label
- class is similarity relation or *none* (=no alignment).
- we use a memory-based learner TiMBL (Daelemans et al., 2009)
- trained on instances derived from a parallel treebank of aligned and labeled syntactic trees

Step 3: Weighting

- associate a *cost* with each alignment
  - low cost indicates classifier is confident about the alignment
  - high cost indicates classifier is not sure about the alignment
- we use as cost the *normalized entropy* of the class labels in the set of nearest neighbours
  - if all nearest neighbours are of the same class, cost is zero
  - if nearest neighbours are equally distributed over all possible classes, cost goes to 1

Step 4: Matching

- we search for a node matching which *minimizes the sum of costs* over all alignments
- known as the *Assignment Problem*
- can be solved in polynomial time ($O(n3)$) using the Hungarian algorithm (Kuhn, 1955).

Properties of Develop and Test Set

- Data consists of manually annotated part of the news segment of the DAESO corpus

Data	Graph pairs	Node pairs	Aligned nodes (%)
develop	2 664	22 741	47.20
test	547	4 894	47.05

- majority of nodes is unaligned

Distribution of semantic similarity relations

Data	Eq	Re	Spec	Gen	Int
develop	56.61	6.57	7.52	6.38	22.91
test	58.40	7.11	7.40	6.38	20.72

- distribution is skewed towards Equals

Baseline: Greedy alignment

- Greedy alignment:
  - identical words are aligned as Equals
  - identical roots as Restates
- Same procedure is extended to greedy alignment of phrases
  - phrases with identical words are aligned as Equals
  - phrases with identical roots as Restates
- Note: alignment of Equals is not trivial nor deterministic!
  - same word may occur multiple times in the source or target sentences, which introduces ambiguity
  - frequently occurs with function words such as determiners and prepositions

Features

- Word-level features
  - rely on pure string processing
  - example: words share a prefix/infix/suffix
- Morphological features
  - rely on stemming and decompounding
  - example: source root equals target root
- POS features
  - example: source POS equals target POS
- Lexical-semantic features
  - using Lin similarity measure (Lin, 1998)
  - require Cornetto, an extended Dutch WordNet
  - require background corpus for word counts
  - example: Lin similarity score if source word is a hyponym of target word, zero otherwise

Features (cont'd)

- Syntactic features
  - rely on parse trees
  - example: source node category equals target node category
- Phrasal features
  - express similarity between the terminal yields of source and target nodes
  - example: length of source/target phrase in words

Experimental setting

- Results presented here concern
  - Memory-based Graph Matcher (MBGM)
  - full tree alignment and labeling; paper also contains results restricted to word alignment and labeling
  - F-scores; paper also reports precision and recall
- Scores on develop set are averages over 10-fold CV
- Coarse-grained optimization of two parameters
  - downsampling of the *none* class (range 0.1 to 0.5)
  - parameter k of memory-based classifier (range 1 to 15)
- Human upperbound scores are inter-annotator agreement scores on different samples, so formally comparing them with MBGM scores is not sound

Results on alignment, regardless of relation

	F-score:
Dev baseline:	65.67
Dev MBGM:	86.27
Test baseline:	67.43
Test MBGM:	86.65
Human:	88.31

- MBGM scores about 20% above baseline
- significantly better than the baseline system ($t(18) = 17.72, p < .0001$)
- but still below human upperbound
- scores on dev and test set are very similar

Results on relation labeling

	Eq:	Re:	Spec:	Gen:	Int:
Dev baseline:	88.43	28.02	0.00	0.00	0.00
Dev MBGM:	95.08	36.08	31.03	38.54	70.30
Test baseline:	89.85	21.12	0.00	0.00	0.00
Test MBGM:	95.60	38.11	27.60	40.07	69.80
Human:	95.83	71.38	60.21	66.71	62.67

- MBGM significantly outperforms the baseline, for each semantic relation ($t(18) > 12.6636, p < .0001$)
- scores on Equals and Intersects are comparable to human upperbound
- but scores on other relations are much lower
- scores on dev and test set are very similar

Conclusion

- We analysed semantic similarity between comparable sentences by aligning their syntax trees
  - matching each source node to the most similar target node node (if any)
  - and labeling according to one of five semantic similarity relations
- We presented a Memory-based Graph Matcher that performs both tasks simultaneously by
  - exhaustive pairwise classification using a memory-based learner
  - global optimization of alignments using a combinatorial optimization algorithm

Conclusion (cont'd)

- Results on aligning comparable news texts from a monolingual parallel treebank for Dutch show that MBGM consistently and significantly outperforms the baseline,
- both for alignment and labeling
- However, performance on labeling Restates, Specifies and Generalizes is still weak

Future work

- test on parallel/comparable data from other domains
- try additional features, e.g., on the basis of word space models
- extrinsic evaluation in the context of multi-document summarization
- public release of MBGM implemenation