Semantic Types of Some Generic Relation Arguments: Detection and Evaluation

Sophia Katrenko
Institute of Informatics
University of Amsterdam
the Netherlands
katrenko@science.uva.nl

Pieter Adriaans
Institute of Informatics
University of Amsterdam
the Netherlands
pietera@science.uva.nl

Abstract

This paper presents an approach to detection of the semantic types of relation arguments employing the WordNet hierarchy. Using the SemEval-2007 data, we show that the method allows to generalize relation arguments with high precision for such generic relations as *Origin-Entity*, *Content-Container*, *Instrument-Agency* and some other.

1 Introduction and Motivation

A common approach to learning relations is composed from two steps, identification of arguments and relation validation. This methodology is widely used in different domains, such as biomedical. For instance, in order to extract instances of a relation of protein interactions, one has to first identify all protein names in text and, second, verify if a relation between them holds.

Clearly, if arguments are already given, accuracy of relation validation is higher compared to the situation when the arguments have to be identified automatically. In either case, this methodology is effective for the domain-dependent relations but is not considered for more generic relation types. If a relation is more generic, such as *Part-Whole*, it is more difficult to identify its arguments because they can be of many different semantic types. An example below contains a causality relation (**virus** causes **flu**). Note that syntactic information is not sufficient to be able to detect such relation mention and the background knowledge is needed.

> A person infected with a particular **flu virus** strain develops antibody against that virus.

In this paper we propose a method to detect semantic types of the generic relation arguments. For the *Part-Whole* relation, it is known that it embraces such subtypes as *Member-Collection* or *Place-Area* while there is not much information on the other relation types. We do not claim semantic typing to be sufficient to recognize relation mentions in text, however, it would be interesting to examine the accuracy of relation extraction when the background knowledge *only* is used. Our aim is therefore to discover precise generalizations per relation type rather than to cover all possible relation mentions.

2 A Method: Making Semantic Types of Arguments Explicit

We propose a method for generalizing relation argument types based on the positive and negative examples of a given relation type. It is also necessary that the arguments of a relation are annotated using some semantic taxonomy, such as WordNet (Fellbaum, 1998). Our hypothesis is as follows: because of the positive and negative examples, it should be possible to restrict semantic types of arguments using negative examples. If negative examples are nearly positive, the results of such generalization should be precise. Or, in machine learning terms, such negative examples are close to the decision boundary and if used during generalization, precision will be boosted. If negative examples are far from the decision boundary, their use will most likely not help to identify semantic types and will result in overgeneralization.

To test this hypothesis, we use an idea borrowed from induction of the deterministic finite automata.

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 185–188,
Columbus, Ohio, USA, June 2008.

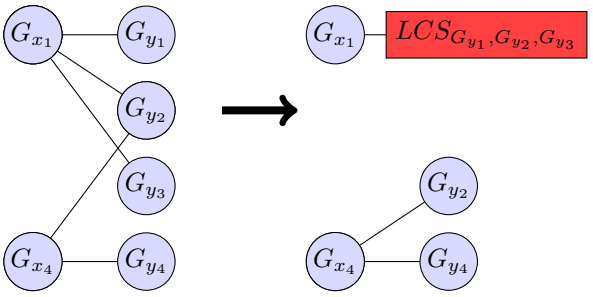

Figure 1: Generalization process.

More precisely, to infer deterministic finite automata (DFA) from positive and negative examples, one first builds the maximal canonical automaton (MCA) (Pernot et al., 2005) with one starting state and a separate sequence of states for each positive example and then uses a merging strategy such that no negative examples are accepted.

Similarly, for a positive example $< x_i, y_i >$ we collect all f hyperonyms $H_{x_i} = h^1_{x_i}, h^2_{x_i}, \ldots, h^f_{x_i}$ for x_i where $h^1_{x_i}$ is an immediate hyperonym and $h^f_{x_i}$ is the most general hyperonym. The same is done for y_i. Next, we use all negative examples to find G_{x_i} and G_{y_i} which are generalization types of the arguments of a given positive example $< x_i, y_i >$. In other words, we perform generalization per relation argument in a form of *one positive example vs. all negative examples*. Because of the multi-inheritance present in WordNet, it is possible to find more hyperonymy paths than one. To take it into account, the most general hyperonym $h^f_{x_i}$ equals to a splitting point/node.

It is reasonable to assume that the presence of a general semantic category of one argument will require a more specific semantic category for the other. Generalization per argument is, on the one hand, useful because none of the arguments share a semantic category with the corresponding arguments of all negative examples. On the other hand, it is too restrictive if one aims at identification of the relation type. To avoid this, we propose to generalize semantic category of one argument by taking into account a semantic category of the other. In particular, one can represent a binary relation as a bipartite graph where the corresponding nodes (relation arguments) are connected. A natural way of generalizing would be to combine the nodes which differ on the basis of their similarity. In case of WordNet, we can use a least common subsumer (LCS) of the nodes. Given the bipartite graph in Figure 1, it can be done as follows. For every vertex G_{x_i} in one part which is connected to several vertices $G_{y_1}, \ldots, G_{y_k}$ in the other, we compute LCS of $G_{y_1}, \ldots, G_{y_k}$. Note that we require the semantic contrains on both arguments to be satisfied in order to validate a given relation. Generalization via LCS is carried out in both directions. This step is described in more detail in Algorithm 1.

Algorithm 1 Generalization via LCS

1: Memory $\mathcal{M} = \varnothing$
2: Direction: $\rightarrow$
3: **for all** $< G_{x_i}, G_{y_i} > \in \mathcal{G}$ **do**
4: Collect all $< G_{x_j}, G_{y_j} >, j = 0, \ldots, l$ s. t. $G_{x_i} = G_{x_j}$
5: **if** exists $< G_{x_k}, G_{y_j} >$ s. t. $G_{x_i} \neq G_{x_k}$ **then**
6: $\mathcal{G} = \mathcal{G} \cup \{< G_{x_j}, G_{y_j} >\}$
7: **end if**
8: Compute $\mathcal{L} = LCS_{G_{y_0},\ldots,G_{y_l}}$
9: Replace $< G_{x_j}, G_{y_j} >, j = 0, \ldots, l$ with $< G_{x_j}, \mathcal{L} >$ in $\mathcal{G}$
10: $\mathcal{M} = \mathcal{M} \cup \{< G_{x_j}, \mathcal{L} >\}$
11: **end for**
12: Direction: $\leftarrow$
13: **for all** $< G_{x_i}, G_{y_i} > \in \mathcal{G}$ **do**
14: Collect all $< G_{x_j}, G_{y_j} >, j = 0, \ldots, l$ s. t. $G_{y_i} = G_{y_j}$ and $< G_{x_j}, G_{y_j} > \notin \mathcal{M}$
15: Compute $\mathcal{L} = LCS_{G_{x_0},\ldots,G_{x_l}}$
16: Replace $< G_{x_j}, G_{y_j} >, j = 0, \ldots, l$ with $< \mathcal{L}, G_{y_j} >$ in $\mathcal{G}$
17: **end for**
18: **return** $\mathcal{G}$

Example Consider, for instance, two sentences from the SemEval data (Instrument-Agency relation).

> 013 ”The test is made by inserting the end of a <e1>jimmy</e1> or other <e2>burglar</e2>’s tool and endeavouring to produce impressions similar to those which have been found on doors or windows.” WordNet(e1) = ”jimmy%1:06:00::”, WordNet(e2) = ”burglar%1:18:00::”, Instrument-Agency(e1, e2) = ”true”
> 040 ”<e1>Thieves</e1> used a <e2>blowtorch</e2> and bolt cutters to force their way through a fenced area

topped with razor wire." WordNet(e1) = "thief%1:18:00::", WordNet(e2) = "blowtorch%1:06:00::", Instrument-Agency(e2, e1) = "true"

First, we find the sense keys corresponding to the relation arguments, ("jimmy%1:06:00::", "burglar%1:18:00::") = (jimmy#1, burglar#1) and ("blowtorch%1:06:00::", "thief%1:18:00::") = (blowtorch#1, thief#1).By using negative examples, we obtain the following pairs: (apparatus#1, bad_person#1) and (bar#3, bad_person#1). These pairs share the second argument and it makes it possible to apply generalization in the direction $\leftarrow$. LCS of apparatus#1 and bar#3 is instrumentality#3 and hence the generalized pair becomes (instrumentality#3, bad_person#1).

Note that an order in which the directions are chosen in Algorithm 1 does not affect the resulting generalizations. Keeping all generalized pairs in the memory $\mathcal{M}$ ensures that whatever direction ($\rightarrow$ or $\leftarrow$) a user chooses first, the output of the algorithm will be the same.

Until now, we have considered generalization in one step only. It would be natural to extend this approach to the iterative generalization such that it is performed until no further generalization steps can be made (it corresponds either to the two specific argument types or to the situation when the top of the hierarchy is reached). However, such method would most likely result in overgeneralization by boosting recall but drastically decreasing precision. As an alternative we propose to use memory $\mathcal{M}^{\mathcal{I}}$ defined over the iterations. After each iteration step every generalized pair $< G_{x_i}, G_{y_i} >$ is applied to the training set and if it accepts at least one negative example, it is either removed from the set $\mathcal{G}$ (first iteration) or this generalization pair is decomposed back into the pairs it was formed from (all other iterations). By employing backtracking we guarantee that empirical error on the training set $E_{emp} = 0$.

3 Evaluation

Data For semantic type detection, we use 7 binary relations from the training set of the SemEval-2007 competition, all definitions of which share the requirement of the syntactic closeness of the arguments. Further, their definitions have various restrictions on the nature of the arguments. Short description of the relation types we study is given below.

Cause-Effect(X,Y) This relation takes place if, given a sentence S, it is possible to entail that X is the cause of Y. Y is usually not an entity but a nominal denoting occurrence (activity or event).

Instrument-Agency(X,Y) This relation is true if S entails the fact that X is the instrument of Y (Y uses X). Further, X is an entity and Y is an actor or an activity.

Product-Producer(X,Y) X is a product of Y, or Y produces X, where X is any abstract or concrete object.

Origin-Entity(X,Y) X is the origin of Y where X can be spatial or material and Y is the entity derived from the origin.

Theme-Tool(X,Y) The tool Y is intended for X is either its result or something that is acted upon.

Part-Whole(X,Y) X is part of Y and this relation can be one of the following five types: Place-Area, Stuff-Object, Portion-Mass, Member-Collection and Component-Integral object.

Content-Container(X,Y) A sentence S entails the fact that X is stored inside Y. Moreover, X is not a component of Y and can be removed from it.

We hypothesize that *Cause-Effect* and *Part-Whole* are the relation types which may require sentential information to be detected. These two relations allow a greater variety of arguments and the semantic information alone might be not sufficient. Such relation types as *Product-Producer* or *Instrument-Agency* are likely to benefit more from the external knowledge. Our method depends on the positive and negative examples in the training set and on the semantic hierarchy we use. If some parts of the hierarchy are more flat, the resulting patterns may be too general.

As not all examples have been annotated with the information from WordNet, we removed them form the test data while conducting this experiment. *Content-Container* turned out to be the only relation type whose examples are fully annotated. In contrast, *Product-Producer* is a relation type with the most information missing (9 examples removed). There is no reason to treat relation mentions as mutually exclusive, therefore, only negative example provided for a particular relation type are used to determine semantic types of its arguments.

Discussion The entire generalization process results in a zero-error on the training set. It does not, however, guarantee to hold given a new data set. The loss in precision on the unseen exam-

Relation type	P, %	R, %	A, %	B-A, %
Origin-Entity	100	26.5	67.5	55.6
Content-Container	81.8	47.4	67.6	51.4
Cause-Effect	100	2.8	52.7	51.2
Instrument-Agency	78.3	48.7	67.6	51.3
Product-Producer	77.8	38.2	52.4	66.7
Theme-Tool	66.7	8.3	65.2	59.2
Part-Whole	66.7	15.4	66.2	63.9
avg.	**81.6**	26.8	62.7	57.0

Table 1: Performance on the test data

ples can be caused by the generalization pairs where both arguments are generalized to the higher level in the hierarchy than it ought to be. To check how the algorithm behaves, we first evaluate the specialization step on the test data from the SemEval challenge. Among all the relation types, only *Instrument-Agency*, *Part-Whole* and *Content-Container* fail to obtain 100% precision after the specialization step. It means that, already at this stage, there are some false positives and the contextual classification is required to achieve better performance.

The results of the method introduced here are presented in Table 1. Systems which participated in SemEval were categorized depending on the input information they have used. The category *WordNet* implies that WordNet was employed but it does not exclude a possibility of using other resources. Therefore, to estimate how well our method performs, we calculated accuracy and compared it against a baseline that always returns the most frequent class label (B-A). Given the results of the teams participating in the challenge, the organizers mention *Product-Producer* as one of the easiest relations, while *Origin-Entity* and *Theme-Tool* are considered to be ones of the hardest to detect (Girju et al., 2007). Interestingly, *Origin-Entity* obtains the highest precision compared to the other relation types while using our approach.

Table 2 contains some examples of the semantic types we found for each relation. Some of them are quite specific (e.g., *Origin-Entity*), while the other arguments may be very general (e.g., *Cause-Effect*). The examples of the patterns for *Part-Whole* can be divided in several subtypes, such as Member-Collection (person#1, social_group#1), Place-Area (top_side#1, whole#2) or Stuff-Object (germanium#1, mineral#1).

Relation	(G_X, G_Y)
Content-Container	(physical_entity#1, vessel#3)
Instrument-Agency	(instrumentality#3, bad_person#1) (printing_machine#1, employee#1)
Cause-Effect	(cognitive_operation#1, joy#1) (entity#1, harm#2) (cognitive_content#1, communication#2)
Product-Producer	(knowledge#1, social_unit#1) (content#2, individual#1) (instrumentality#3, business_organisation#1)
Origin-Entity	(article#1, section#1) (vegetation#1, plant_part#1) (physical_entity#1, fat#1)
Theme-Tool	(abstract_entity#1, implementation#2) (animal#1, water#6) (nonaccomplishment#1, human_action#1)
Part-Whole	(top_side#1, whole#2) (germanium#1, mineral#1) (person#1, social_group#1)

Table 2: Some examples per relation type.

4 Conclusions

As expected, the semantic types derived for such relations as *Origin-Entity*, *Content-Container* and *Instrument-Agency* provide high precision on the test data. In contrast, precision for *Theme-Tool* is the lowest which has been noted by the participants of the SemEval-2007. In terms of accuracy, *Cause-Effect* seems to obtain 100% precision but low recall and accuracy. An explanation for that might be a fact that causation can be characterized by a great variety of argument types many of which have been absent in the training data. *Origin-Entity* obtains the maximal precision with accuracy much higher than baseline.

References

Christiane Fellbaum. 1998. WordNet: An Electronic Lexical Database. *MIT Press*.

Nicholas Pernot, Antoine Cornuéjols, and Michele Sebag. 2005. Phase transition within grammatical inference. *In Proceedings of IJCAI 2005*.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney and Deniz Yuret. 2007. SemEval-2007 Task 04: Classification of Semantic Relations between Nominals. *In ACL 2007*.