# Semi-automatic rule construction for semantic linking of relation arguments

*Janez Starc, Dunja Mladenić*
Artificial Intelligence Laboratory
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
{janez.starc, dunja.mladenic}@ijs.si

**ABSTRACT**

**In this paper, we propose an iterative semi-automatic approach for linking textual arguments of relations to their semantic form using rules. Textual arguments are completely decomposed – every word is considered. They are composed back into semantic form using functions, which bring additional semantic information. The process starts with an initial set of seed rules, which can be obtained automatically. In each iteration, the user constructs new rules using the recommendations, which are calculated based on the frequency statistics of unlinked textual arguments. Our approach was tested on extraction of roles that people have in organizations. The results show that only 31 human crafted rules are needed to link more than 3400 additional arguments. We also show that combining rules have positive effects. The number of linked arguments grows super-linearly with respect to the number of patterns.**

## 1 INTRODUCTION

Lately, methods for relation extraction from text have had a considerable amount of success. Relatively high percision, recall and F1 scores, and very little human intervention are usually their properties. On the other hand, many of them leave the arguments in textual form [1], without automatically linking them to a semantic knowledge base, such as Freebase, DBpedia, OpenCyc. In this way, the extracted knowledge is less actionable and less automated reasoning can be done on it.

For example, suppose some relation extraction method extracted the following relation (isa "blue plastic spoon" KitchenItem) and this is the only relation about "blue plastic spoon". The relation is not wrong, however, more information could be extracted. For example, the observed argument is a spoon, made out of plastic.

(isa "blue plastic spoon" Spoon)
(mainMaterial "blue plastic spoon" Plastic)
(prevailingColor "blue plastic spoon" BlueColor)

The observed entity can also be expressed with one expression, containing the same information as in the three relations above.

(ColorFn (MaterialFn Spoon Plastic) BlueColor)

*ColorFn* and *MaterialFn* are functions. *MaterialFn* accepts a tangible object and a material as arguments. The result of the function is a new entity. In our case, this is a spoon made out of plastic. This kind of knowledge representation is very suitable for recursive information extraction, where arguments can be further split into sub-arguments. Many functions can be found in Cyc [1].

Noun phrases are usually arguments that are to be linked. However, this is not always the case. For example, in the sentence: *"Stocks will tumble" John told reporters…* it is possible to further extract John's statement.

### *Rule definition*

When extracting relations and establishing connection to a semantic knowledge base, semantic forms are assigned to textual arguments by applying *rules*. A semantic form is composed of functions, predicates and entities from a particular knowledge base. These terms are combined in such a way that they express the meaning of the textual argument. Rules are composed of a lexical part and a semantic part. The lexical part consists of fixed words and empty slots. When the lexical part is applied to the text, empty slots become filled with words and become lexical arguments. The semantic part consists of terms from the knowledge base and empty slots, which become filled with the semantic form of the corresponding lexical argument.

Rules that do not have arguments are *entity rules*. For example,

"Barack Obama" → id39813

In this rule, the name of the US president is assigned an id from a particular knowledge base.

*Pattern rules* have at least one argument. For instance,

"blue" [object] -> (ColorFn [object] BlueColor)

is the pattern rule that is used in the previous example.

### *Related work*

Our problem has similarities with *entity linking* [2]. The goal of entity linking is to link noun phrases to entities in a large database. However, many noun phrases do not have a corresponding entity in the knowledge base, and some are not even entities. The problem of determining the type of

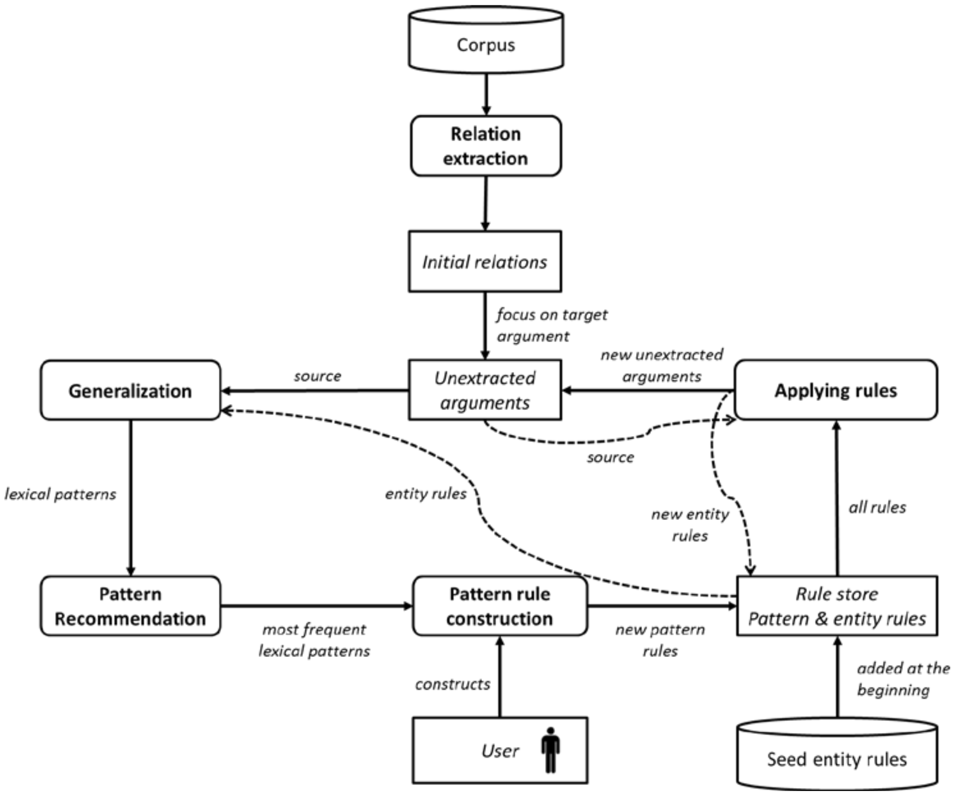

Figure 1: *Architecture of the system*

such entities is studied in [3]. In our case, the type of the argument is known from the beginning, similar like in targeted disambiguation problem, which was examined by [4]. Our goal is not only to discover if the argument belongs to the target type, but also providing additional information by decomposing it.

In the following sections, we first present our approach to link arguments (Section 2). We evaluated our approach on a relation that expresses roles of people in organizations. The experimental setting and results are presented in Section 3. The discussion follows in the final section (Section 4).

## 2 APPROACH

This section describes the proposed approach. The architecture of the system is presented in Figure 1. The input to the system consists of (1) a corpus containing several text documents relevant for the rules we would like to construct and (2) seed entity rules providing the starting point for extraction. Seed entity rules should be taken from the target knowledge base. Notice that a human user is also needed to finalize the construction of pattern rules based on the recommendations provided by the system. The result of our approach are rules, which provide links for the selected textual arguments. Selected arguments represent one position in the relation. For instance, we would to link *role* arguments from *roleInOrganization* relations.

*(roleInOrganization person organization **role**)*

### *Relation extraction*

A relation extraction method must be chosen to extract the relations of the selected type from the corpus. This method can be very simple, e.g., a set of hand-crafted patterns or more sophisticated like semi-supervised method from [5]. The selected arguments are placed on the unlinked argument list before the iterative procedure, which is presented in the following section.

### *Iterative procedure*

Each iteration starts with *generalization* of unlinked arguments into lexical patterns. Some lexical patterns are presented to the user as *recommendations*. The user is encouraged to use them as lexical parts in the *pattern rule construction*. Newly created pattern rules are added to the rule store. In the next step, both pattern and entity rules are *applied* on unlinked arguments, which results in new unlinked arguments and entity rules. This is the last step of the iteration and it is explained in details below.

In the first iteration, the user skips his turn and the seed entity rules are added to the empty rule store. Therefore, the first application of rules is done without pattern rules.

For the rest of this section, we will present each step shown in Figure 1 in more details.

### *Generalization*

Generalization is used to group similar arguments into *lexical patterns*. In process of generalization, parts (substrings) of arguments are replaced by their types. For instance:

“vice president” is replaced by [Role],
“London” is replaced by [Location],
“October 2012” is replaced by [Date].

The first example shows the generalization of substrings that are generalized to the type of the observed argument. If a part of an argument matches the lexical part of any entity rule, then it is generalized. The bottom two examples show generalization of other entities. Using named-entity resolution, locations, organizations, people and dates are generalized. Numbers are generalized using part-of-speech tagger. In case, that two overlapping substrings are candidates for generalization, longer substring is generalized. If two overlapping candidates are equal length, the one that starts first is generalized.

***Pattern recommendation***
The user, who constructs pattern rules, can use the recommended lexical patterns to construct rules that will provide many correct links. Lexical patterns obtained in the generalization step, which do not contain any target type generalizations, are discarded. The remaining lexical patterns are sorted according to their frequencies and only the most frequent lexical patterns are presented to the user. The frequency of the lexical pattern is the number of unlinked arguments that matches it. Different lexical patterns will appear on the top of the list in different iterations. Lexical patterns that have been used in pattern rules will automatically disappear from the list in the succeeding iteration. Other patterns will have at least the frequency they had in the preceding iteration.

***Rule construction***
For the selected lexical patterns, the user constructs their semantic parts using the terms from the knowledge base or creates new ones. The newly created pattern rules are added to the rule store. Different orders of rules can produce different results. Entity rules are placed before pattern rules. The order of pattern rules is defined by the user.

***Applying rules***
In this step, the rules from the rule store are applied on each unlinked argument. The algorithm for applying rules on an argument, *ApplyRules*, is presented on Figure 2. The result of the algorithm is the semantic form of the argument. Completely linked arguments have no empty slots in their semantic form. They are removed from the list of unlinked arguments and moved to the rule store in a form of an entity rule. In this rule, the argument presents the lexical part, and the result of the algorithm presents the semantic part. If the argument is partially linked, which means that some rules did apply, but not all parts are linked, then the unlinked parts are added to the unlinked arguments. In the latter case, where none of the rules apply, the argument is left in the unlinked argument list.

```
Algorithm 1 ApplyRules
Input: argument $Arg$, rule store $RS$, new argument list $NewArgs$
Output: semantic form $SF$

$SF \leftarrow empty$
while $RS$ has more rules $and$ $SF = empty$ do
    rule $R$ = next rule from $RS$
    if $R$.semanticPattern matches $Arg$ then
        $SF \leftarrow R$.semanticPattern
        $SubArgs$ = sub arguments from matching
        for each argument $Arg' \in SubArgs$ do
            $SubSF \leftarrow$ ApplyRules($Arg'$, $RS$, $NewArgs$)
            if $SubSF = empty$ then
                add $Arg'$ to $NewArgs$
            else
                insert $SubSF$ into $SF$
            end if
        end for
    end if
end while
```

Figure 2 Algorithm ApplyRules

## 3 EVALUATION
We evaluated our approach on the *roleInOrganization* relation, which states the role of a person in an organization. For example,

*(roleInOrganization ‘Peter Murphy’ ’The Walt Disney Co.’ ’former strategic officer’ )*

We tried to link the last argument of the relation, the role. To extract the initial relations from the corpus, the following pattern was used

*[person],[role] of [organization]*

The words between the comma and ‘of’ are taken as the role argument. Although we used only one pattern, there were more than 110.000 matches in a corpus of 1.3 million English news articles.
We prepared the seed entity rules using Freebase data. We have chosen a list of types, including job title, leadership role, academic post title, whose instances are roles. Each instance has one property ‘*name*’ and some of them also have property ‘*also known as*’. For each instance, at least one entity rule was constructed. The properties mentioned above represent the lexical part of the rule and the id of the instance presents the semantic part.
The user was presented with 30 most frequent lexical patterns on each iteration. The user repeated the process until no useful lexical patterns with frequency above ten were present on the recommendation list.

***Results***
There were five iterations, in which the user created 31 pattern rules. A selection of constructed rules is presented in Table 1. After the rule construction procedure, rules were applied on initial arguments. Without pattern rules there

| *Lexical part* | *Semantic part* | *Applied on # arguments* |
|---|---|---|
| the [role] | [role] | 2798 |
| [role] and [role] | (and [role] [role]) | 2085 |
| former [role] | (FormerFn [role]) | 229 |
| who was [role] | (FormerFn [role]) | 148 |
| assistant [role] | (AssistantFn [role]) | 26 |

Table 1: *A selection of pattern rules from the experiment. Empty slots are denoted with [pos].*

were 10.819 completely linked arguments. If pattern rules are added, than additional 3.428 arguments were completely linked, and 5.123 are partially linked.

To measure the precision and recall, one evaluator evaluated 300 random arguments together with their semantic form. If the argument is completely linked, then it counts as retrieved. If the semantic form represents the right meaning of the retrieved argument, then it is a true positive. The experiment achieved 100% precision and 84% recall.

In many cases, more than one pattern rule must be applied to completely extract an argument. If one rule is missing, then the argument cannot be completely linked. Therefore, rules do not perform well by themselves. Having lots of rules should be beneficial, because they complement each other.

We made an experiment, where we took the arguments and rules constructed in the experiment. We split the experiment into two cases. In the first case, the pattern rules are combined and in the second case they are applied alone. For each number *k* from one to the number of pattern rules, we randomly selected *k* pattern rules and count how many arguments they completely link. In the first case, the rules are applied together. In the second case, each rule is applied separately and the counts are added up. In both cases, arguments that are completely linked without any pattern rule, only with seed entity rules, are not counted. This procedure is repeated several times and the average for each number *k* is calculated. The results are presented on Figure 3. The trend for the uncombined case is linear. For the combined case the trend is super-linear. If the *k* is maximum (31) then the combined method extracts 24 percent more arguments than uncombined method. The motivation for selecting k rules at random out of all rules was to show that rules have added value if they are combined, and not to measure the growth of linked arguments with every new rule user constructs.

## 4 DISCUSSION

Our approach does not focus on extraction of relations, but exhaustively extracts information from their arguments. The approach is semi-automatic, thus it needs human intervention. However, our experiment shows that with only 31 human made rules, the number of completely linked arguments increases by 32%. Furthermore, many of these have semantic forms of better quality, because they are composed of functions.

The experiment was done only on one relation - *roleInOrganization*. However, this approach could be used

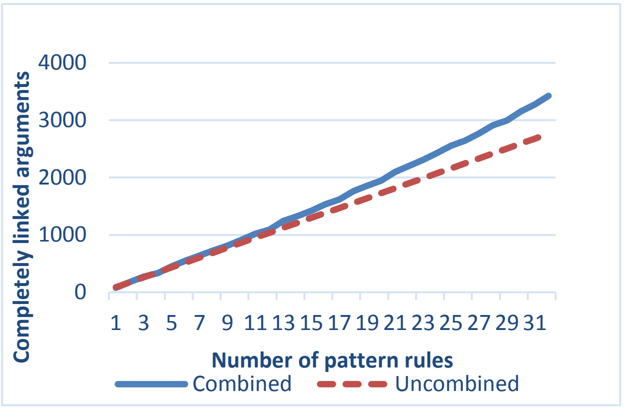

Figure 3 *Comparison of combined and uncombined approach*

on any relation. It works better if target arguments have many common words, if they can be decomposed with functions, and if enough seed entity rules are provided.

***Acknowledgements***

This work was supported by Slovenian Research Agency and the ICT Programme of the EC under XLike (ICT-STREP-288342).

## References

[1] Matuszek, Cynthia and Cabral, John and Witbrock, Michael J and DeOliveira, John, "An Introduction to the Syntax and Content of Cyc.," in *AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, 2006.

[2] R. C. a. P. M. Bunescu, "Using Encyclopedic Knowledge for Named entity Disambiguation.," in *In Proceedings of the 11th Conference of the European Chapter of the Association of Computational Liguistics (EACL)*, 2006.

[3] Lin, Thomas and Mausam, Oren Etzioni, "No Noun Phrase Left Behind: Detecting and Typing Unlinkable Entities," in *oceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.

[4] Chi Wang, Kaushik Chakrabarti, Tao Cheng, and Surajit, "Targeted disambiguation of ad-hoc, homogeneous sets of named entities," in *In Proceedings of the 21st International World Wide Web Conference (WWW)*, 2012.

[5] Agichtein, E. and Gravano, L., "Snowball: Extracting relations from large plain-text collections," in *Proceedings of the fifth ACM conference on Digital libraries*, 2000.

[6] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr. and T.M. Mitchell., "Toward an Architecture for Never-Ending Language Learning," in *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010.