Identifying good patterns for relation extraction

Janez Starc, Blaž Fortuna
Artificial Intelligence Laboratory
Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
{janez.starc, blaz.fortuna}@ijs.si

ABSTRACT

In pattern based relation extraction, patterns that with high precision and recall produce semantically useful relations are preferred. We present a technique similar to n-gram extraction that extracts patterns from large text corpora and calculates statistics, like frequency, minimal token frequency and normalized expectation, which guide to preferred patterns. Patterns have named-instances and/or one variable length gap as arguments. We extracted patterns from a large news corpus and translated them to Cyc relations. We focused on four patterns, which we evaluate by asserting their translated relations to Cyc knowledge base.

1 INTRODUCTION

In this paper, we present an approach that identifies good patterns, which can be later used for domain-independent relation extraction. For instance, pattern "*[PERSON] was born in [LOCATION]*" might be used to extract relation *(placeOfBirth TigerWoods CityOfCypressCalifornia)* from unstructured text. We distinguish two types of relations. The first ones represent a fact. They consist of a predicate and arguments. The second ones represents a concept, which can be used as an argument to other relations. For instance *(StreetIntersectionFn PortageAvenue MainStreet)* represents the concept of the famous street corner in Canada. These relations consist of a function and arguments.

Mapping rule translates one match of the pattern to one, or several relations if any argument represents more than one argument. If word sense disambiguation is applied, we can reduce the result of translation to one relation. Relations are constructed in such a way that they can populate the target ontology. All relation's arguments must comply with semantic restrictions of the ontology in order for relation to be asserted in the ontology. For instance, a concept must be an instance of a person to become a valid argument of particular relation. Mapping rules are usually handcrafted. Our system helps human designers find patterns that are good enough to design mapping rules for them.

But, what is a good pattern? Good patterns give *high recall*, *precision* and *semantic usefulness* of the extracted relations. In our work, we built a system that optimizes this three metrics. We define recall of the pattern as the total number of unique matches of a pattern in the examined corpus. Precision is defined as the fraction of matches that produce a valid relation. Relation is valid if it complies with all the semantic restrictions, and if the extracted relation really reflects the meaning of the mention in the text. Semantic usefulness depends on how much do extracted relations contribute to the application that uses them. If the application is ontology population, semantically richer relations are favoured. While recall is very easy to measure, human evaluation needs to be done to measure the precision. Finally, semantic usefulness is almost unmeasurable because of its subjective nature.

Pattern based approach to relation extraction emerged in the early nineties, with the use of lexico-synctactic (Hearst) patterns to extract hypernym (is-a) relations [1]. In [2] similar approach was used to extract meronymy (part-of) relations. Next, semi-supervised approaches become widely used. They use a very small number of seed patterns or instances of relations to do bootstrap learning [3] [4] [5]. Recently, unsupervised (open) relation extraction techniques become very popular [6]. In these systems, relations are learned automatically from very large corpora. In contrary, our system does not learn relations neither pattern-relation pairs automatically. However, it can be used to find good patterns to extract relations in a domain-specific environment or to prepare seed patterns for the bootstrapping approach.

In Section 2, we present our system, which helps separate good patterns from the rest. In Section 3, we evaluate several patterns. We end with discussion and future work in Section 4.

2 IDENTIFIYING GOOD PATTERNS

We developed a scalable system that extracts patterns from a specially prepared corpus and calculates a few statistics that help identify good patterns. In this corpus, each sentence is an independent unit, disallowing patterns to be split across sentences. We replaced named instances with their types using a named entity recognizer [7]. We used the following types: person, location, organization, date, and money. These are later used as the arguments of the patterns. Each sentence is split on tokens where each token is defined as a part of text (usually words) that is tagged by a single part-of-speech tag, or an entity category.

Our system extracts two kinds of patterns. The first ones are fixed size n-grams. Each pattern is a sequence of n tokens, which contains at least one token that translates to an argument. Table 1 shows a pattern of the second type. These patterns have one variable length gap, which

Pattern	Frequency
[PERSON] , [__] of [ORGANIZATION]	9392
Gap filler	**Frequency**
president	878
director	818
chairman	560
head	549
one	449
executive director	438
a member	247
…	

Table 1: *Variable length gap n-gram representation*

becomes one of the arguments. Gaps are not allowed at the beginning or the end of the pattern. In this case, the length of the string that fills the gap, gap filler, is automatically defined. If the gap fillers are of the same type, the pattern has better chances of becoming a good pattern. The type of most of the gap fillers on Table 1 is "position in an organization". Therefore, one could make the mapping rule to the following relation template *(positionOfPersonInOrganization ?Person ?Organization ?Position).*

We extracted patterns from a corpus containing about half a million English news articles, which is about 14 million sentences. The extraction produced two sets of n-grams: a set of n-grams of length five tokens or less, and a set of n-grams that occurred at least twice and were of length ten tokens or less. From these n-grams, we generated all possible n-grams with one gap that had maximally five non-gap tokens.

The output of our system is a table of equally long patterns and their statistics. Table 2 shows part of the table for 6-grams patterns. We will present an example of how to manipulate the table to obtain good patterns. Patterns with very low pattern frequency (*Fq*) were filtered out to achieve good recall. We were only interested in patterns with two or three arguments (*Args*). Other patterns were filtered out. One could also order or filter the table according to the number of stop words (*StopW*). In our case, stop words are tokens from the standard stop-word list and non-alphabetical tokens. The table is sorted according to minimal token frequency (*MinTokFq*), which is the frequency of the token that appears the least. Patterns with high minimal token frequency are usually semantically poor, because they are too general. Similarly, patterns with higher normalized expectation (*NExp*) are usually semantically richer. As defined in [8], normalized expectation between n words is the average expectation of one word occurring in a given position knowing presence of other n-1 words also constrained by their positions.

$$NExp([w_1 w_2 \dots w_n]) = \frac{p([w_1 w_2 \dots w_n])}{\frac{1}{n}\sum_{i=1}^{n} p([w_1 w_2 \dots \widehat{w_i} \dots w_n])},$$

where $p([w_1 w_2 \dots w_n])$ denotes the probability of n-gram $[w_1 w_2 \dots w_n]$ occurring in the corpus. Term $\widehat{w_i}$ signifies that word w_i omitted from the n-gram, which becomes (n-1)-gram, which potentially has a gap. Authors in [8] have shown that normalized expectation multiplied with the frequency gives mutual expectation, which can be applied to find multiword units.

3 EVALUATION

In this section we present an experiment, where we selected several patterns with the process explained in Section 2, and translated them to relations, written in Cyc's language, CycL [8].

3.1 Pattern matching and translation algorithm

Our algorithm processes the corpus one sentence at the time. First, it finds all matches of the provided patterns in the sentence. Then, it first translates the patterns, which relations represent concepts. In the next step, it tries if any of these concepts fit as an argument in a pattern, which translates into a fact. These type of patterns are translated in the last step.

The translation of the pattern is done in the following way. First, each argument of the pattern (string) needs to be translated into one or more Cyc concepts. We inquired Cyc to obtain concepts that denote the argument and are instances of the argument's type. For example, we inquired the entity-type pair *("Boulder", Location).* The system accepts the first answer to the query, *(CityNamedFn "Boulder" Colorado-State)*, because it is a location, and rejects the second concept, Boulder, which represents the collection of all boulders (stones). If the query does not return and reject any concepts, we create a new concept. There are as many relations created, as there are combinations of concept assigned to each argument. At the end of the procedure, we assert all relations into Cyc's ontology.

3.2 Experiment

We used a test corpus of about 7500 news articles, published the same day, to test a few dozen rules. We will present evaluation of four different rules. One human evaluator examined all the match-assertions pairs. He was given the sentence containing the match and the translated relations to subjectively decide, whether the meaning of one of the assertions is also found in the sentence. Precision was calculated based on this number. Not having a system for word sense disambiguation integrated in our system, we considered the translation successful, even though the match had one valid assertion, but other assertions were not valid.

Pattern coachOfOrganization

This pattern connects sport coaches to sport organizations (Table 3). From pattern statistics table we expected a bigger recall for this pattern. However, many articles have talked about one NBA game. Most of the matches were connecting these two clubs to their coaches. Out of 70

Pattern	*Fq*	*Args*	*StopW*	*MinTokFq*	*NExp*
[PERSON] , executive director of *[ORGANIZATION]*	21	2	2	11699	0.484
's hospital in *[LOCATION]* , *[LOCATION]*	22	2	2	11917	0.571
to *[PERSON]* parents , *[PERSON]* was	40	2	3	12020	0.564
death by *[PERSON]* parents , *[PERSON]*	24	2	2	12020	0.282
[PERSON] parents , *[PERSON]* and *[PERSON]*	22	3	2	12020	0.506
[PERSON] have no idea what *[PERSON]*	20	2	3	12449	0.553
(*[ORGANIZATION]*) - *[PERSON]* scored	22	2	3	12514	0.735
victory over the *[ORGANIZATION]* on *[DATE]*	55	2	3	12626	0.653
, died *[DATE]* , at *[ORGANIZATION]*	61	2	3	12822	0.712
died *[DATE]* , at *[ORGANIZATION]* in	45	2	3	12822	0.732
[PERSON] was a member of *[ORGANIZATION]*	38	2	3	13399	0.623

Table 2 *Part of the table representing 6-gram patterns and their statistics*

matches, there were only 24 unique. There were less total assertions (19) than there were matches (24). Some matches were not transformed into assertions because of semantic constraints. While some matches had more than one assertion because one of its arguments had more than one denotation.

Pattern personPositionInOrganization

We present the second pattern on Table 3. The pattern expresses people's positions in organizations. The second argument is a variable length gap. Analysis from Table 1 has shown that most gap fillers are positions in organization. Out of 17 newly created "*POSITION*" concepts, eight were really representing position in organization. Majority of non-valid *"POSITION"* concepts were made out of very long gaps, which occupied almost the whole sentence. Seven "*POISTION*" concepts, like PresidentOfOrganization and CheifExecutiveOfficer, were already in the ontology.

Pattern personMadeAStatement

The analysis of the third pattern is also presented on Table 3. This pattern connects a person to the statement that he gave. The way pattern's CycL is structured, it connects them through the event of informing. This is one of the most frequent patterns in the news articles. It may be not be as semantically rich as the previous patterns. However, together with similar patterns, all the statements of the particular person can be quickly gathered. If the argument *STRING* had been further parsed, the pattern would have been semantically richer. The pattern has many more total assertions than matches. This is because a few matches have a big number of assertions. For instance, in one match LeBron James was mentioned as "James". There are 57 concepts denoting "James". Not even one of them is LeBron James.

Pattern personsFather

The result of the relation produced by this pattern is a concept denoting somebody's father. This pattern usually matches "his father" or "her father". Using the co-reference resolution, pronouns are connected to the name of the persons. This pattern had 147 matches. However, it was used in a fact relation only once. It was an argument in the personSaidAStatement relation.

4 DISCUSSION AND FUTURE WORK

Table 3 shows there were more cases when new term needed to be created than cases where the replacement for the arguments already existed in the ontology. Most of our arguments represented named-instances. It turns out that Cyc's ontology is not very well populated with named-entities. It would be reasonable to find named-instances in other ontologies like DBpedia or Freebase, and connect them to Cyc Ontology.

When constructing a system for relation extraction, the question is whether to create new concepts liberally or to allow only assertions that consist of arguments that are already in the ontology. Our system is in the middle of both extremes: high recall and high precision. New concepts are not created if there are recognized concepts, even if they all fail to meet the semantic requirements. If the system is self-supervised, it might be better to make it more biased towards precision.

One expansion of our system would be allowing variable length gaps at the beginning or the end of the pattern. When the gap is in the inner part, the length is defined by the pattern. However, if the gap is on either end of the pattern, it is hard to decide how many words to put in the argument. There are two approaches to deal this problem: for each of the first few lengths of n-grams check if there are any denoting concepts, or use a natural language parser to detect phrases.

In our case, arguments that represent named-entities have their type assigned from the beginning. On the other hand, types of arguments that represent variable length gaps must

Pattern	*[ORGANIZATION]* coach *[PERSON]*	
CycL template	*(#$hasCoach ?ORGANIZATION ?PERSON)*	
Recall		24
New terms		29
Recognized arguments		9
Total assertions		19
Matches with ambiguous assertions		0
Matches with a valid assertion		16
Precision		0.67

Pattern	*[PERSON]* , *[POSITION]* of *[ORGANIZATION]*	
CycL template	*(#$positionOfPersonInOrganization ?PERSON ?ORGANIZATION ?POSITION)*	
Recall		38
New terms		70
Recognized arguments		14
Total assertions		30
Matches with ambiguous assertions		2
Matches with a valid assertion		20
Precision		0.53

Pattern	*" [STRING] , " [PERSON] said*	
CycL template	*(#$thereExists ?INFORMING (#$and (#$isa ?INFORMING #$Informing) (#$senderOfInfo ?INFORMING ?PERSON) (#$infoTransferred-NLString ?INFORMING ?STRING)))*	
Recall		379
New terms		227
Recognized arguments		35
Total assertions		830
Matches with ambiguous assertions		37
Matches with a valid assertion[2]		337
Precision		0.89

Pattern	*[PERSON] father*	
CycL template	*($#FatherFn ?PERSON)*	
Matches		147
New terms		78
Recognized arguments		24
Valid assertions		1

Table 3 *Evaluation statistics of the selected patterns.* *[2]Expected number of valid assertions after evaluating 100 unique matches*

be manually assigned. We propose a method for finding the most common type of the gap filler. Each gap filler is assigned its type (node) in the hypernym (is-a) relation tree. The algorithm then searches for the lowest node that is the parent of the majority of nodes. Resulting nodes that appear very low in the hierarchy tree are more desirable.

Acknowledgements

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

References

[1] M. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th conference on Computational linguistics-Volume 2*, 1992.

[2] Berland, M. and Charniak, E., "Finding parts in very large corpora," in *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, 1999.

[3] S. Brin, "Extracting patterns and relations from the world wide web," *The World Wide Web and Databases,* pp. 172--183, 1999.

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

[5] Etzioni, O. and Cafarella, M. and Downey, D. and Kok, S. and Popescu, A.M. and Shaked, T. and Soderland, S. and Weld, D.S. and Yates, A., "Web-scale information extraction in knowitall:(preliminary results)," in *Proceedings of the 13th international conference on World Wide Web*, 2004.

[6] Yates, A. and Cafarella, M. and Banko, M. and Etzioni, O. and Broadhead, M. and Soderland, S., "TextRunner: open information extraction on the web," in *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 2007.

[7] Štajner, T. and Rusu, D. and Dali, L. and Fortuna, B. and Mladenić, D. and Grobelnik, M., "Enrycher: service oriented text enrichment," in *Proceedings of SiKDD*, 2009.

[8] Dias, G. and Guilloré, S. and Lopes, J.G.P., "Mutual expectation: a measure for multiword lexical unit extraction," in *Proceedings of VExTAL Venezia per il Trattamento Automatico delle Lingue*, 1999.

[9] D. Lenat, " CYC: A large-scale investment in knowledge infrastructure," *Communications of the ACM,* vol. 38, pp. 33--38, 1995.