Sweat2012: Pattern based English Slot Filling System for Knowledge Base Population at TAC 2012

Fang Liu
Institute of Automation, Chinese Academy of Sciences
Zhongguancun East Road 95,
HaiDian District,Bejing, China, 100190
fliu@nlpr.ia.ac.cn

Jun Zhao
Institute of Automation, Chinese Academy of Sciences
Zhongguancun East Road 95,
HaiDian District,Bejing, China, 100190
jzhao@nlpr.ia.ac.cn

Abstract

In this paper, we describe the english slot filling system of sweat2012 team for Knowledge Base Population (KBP) task at TAC2012. Our slot filling system is based on pattern. In specific, we first construct (target, value) pairs for every attribute of our interests from previous evaluation results, and divide these entity pairs into training set and assessment set; then the extraction patterns are learned from the training set and the confidence of every learned pattern is evaluated on the assessment set. Secondly, we use the learned pattern to extract candidate values for each required slot of target entities. Finally we post-process filler candidates extracted by patterns which includes estimating probabiliy of each candidate, validating these candidates according to tag and type constraints and removing duplications. Our system achieves F-measure 0.099 on the final test dataset, which turns out to be the median level in all 11 teams who summit their results to the English Slot-Filling task this year.

1 Introduction

The goal of Slot Filling task is to harvest new values for pre-defined attributes (slots) of entities from document collection. The input is queries that consist of the name of target entity and additional information like type, unique ID in KB (optional), a document ID that provide disambiguation context and the attributes (slots) which need to be filled. In KBP task, we only concern two kinds of target entities: person and organization. For person, we focus on slots like *birth place*, *age*, *schools*, etc. For organization, we focus on *top_members*, *founders*, *affiliations*, etc.

For the convenience of conveying information, there are some patterns on expressing the mind of people which can be learned by computer. For example, from sentences like this:

"Megawati, 49, told her supporters…"

We aim to implement a system that can learn a pattern from the above sentence as following:

"<TARGET>, <VALUE_NEnumber>"

In our system, we adopt pattern-based method to extract fillers for each slot. Since it is the fourth year of Knowledge Base Population task, an amount of evaluation results (almost 300 target entities) about the past three years are provided which we can make the full use of to train a slot filler extractor. It takes three steps to implement this method. First we need to learn pattern from training data-a set of (target, value) pairs. The training data is constructed from evaluation results of past years. Second, the learned patterns from the first step are used to extract fillers. Finally we filter out wrong and redundant candidates and then rank remained responses. Therefore, it consists of three main modules of our system: pattern learning

module, slot filler extraction module and post-processing module.

The key point in our method is the representation of patterns. Some researchers have done works on this subject. Previously DIPRE (Brin, 1998) used lexicon to generate a pattern tuple, SNOWBALL (Agichtein, 2000) introduced NE type feature into pattern tuples. Later DIRT (Lin, 2001) used dependency path to express a pattern. Recently, PATTY (Nakashole, 2012) gave a taxonomy of relation patterns. It combines syntactic features (S), ontological type signatures (O), and lexical features (L), which is called SOL pattern model. Since our goal is to use pattern to extract information rather than express semantics, we combine NE type, lexical features in our patterns.

The remainder of this paper is organized as follows. Section 2 describes the design and implementation of our slot filling system. Section 3 presents the performance results and some discussions. Finally, section 4 concludes the paper.

2 Our Method

Our system contains offline part and online part. The offline part is pattern learning module used to generate weighted pattern, and the online part consists of slot filler extraction and post processing used to generate fillers for each interested slot of target, as illustrated in Figure 1.

The pattern is learned by following steps:

1. Relocate (target, value) pairs of the training set into collection.
2. Collect the words between and around (before target and after value) target mentions and value, if the distance between them is not too long.
3. Make NE recognition for extracted context of relocated training pair.
4. Pattern consists of all the NE tags recognized and the most frequent words. The most frequent words extraction uses the method similar to PATTY (Nakashole, 2012). The other words are replaced by "*".

1-4 steps are iteratively manipulated on every (target, entity) pair in the training set to collect all candidate patterns.

5. Apply each learned pattern to (target, value) pairs from the assessment set.
6. Make a statistics on the number of corrects and errors for each learned pattern, and assign weight to corresponding patterns.

5-6 steps are iteratively manipulated on every learned pattern. And it does generate new patterns here.

The procedure of online part of our system is like this:

1. Read the name of target entity
2. Expand alternate names of this target
3. Search these expanded names in document collection
4. Apply learned patterns to returned document by Lucene[1]
5. Compute the probability of each candidate
6. Check whether the tag and type constraints are satisfied by candidates
7. Remove duplicated fillers.
8. Rank the remained fillers

A more detailed discussion for each module will be presented in the following subsections.

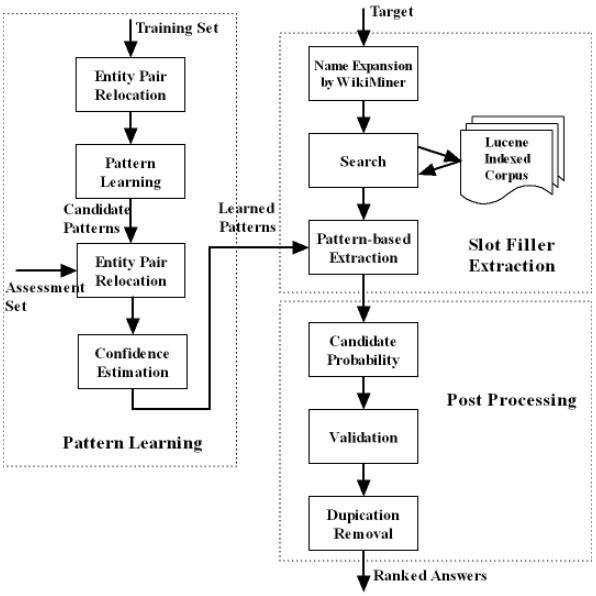

Figure 1: The diagram of our pattern-learning based system

2.1 Pattern Learning

This process is illustrated as pattern learning module (left) in Figure 1. From evaluation results of the past three years, we can easily construct (target, value) pairs and relocate them back to their

[1] http://lucene.apache.org

occurrences in source corpus which is provided for KBP task. In order to estimate the confidence of learned patterns, we split (target, value) pairs into two sets: training set and assessment set. We learn candidate patterns from the training set and compute the confidence according to the following formula for each pattern based on extraction results from the assessment set. The equation indicates extraction precision of pattern which has *c* correct extractions and *e* errors. And *+1* in the dominator is a variant of Laplacian smoothing.

$$confidence = \frac{c}{c+e+1}$$

This confidence value of pattern will later be used in post-processing to compute the probability of slot value candidates.

Our system generates a pattern whenever a pair of (target, value) appears within a window of a certain width (usually 30 words in our settings). The representation of patterns is a crucial part which affects the performance of the system. Named Entity Type information is necessary, and keywords usually co-relocate with certain relationship suggests lexicon is also a good feature. Therefore, a NER-lexical pattern representation is used here. We combine the NER type and the lexicon in a regular expression to find matched values. This pattern expression can basically achieve the goal of our pattern learning mentioned in introduction. We obtain this pattern "<TARGET>, <VALUE_NEnumber>" with 41 correct and 1 wrong extractions.

2.2 Slot Filler Extraction

This process is illustrated as slot filler extraction module (upright) in Figure 1. To obtain a reasonable recall, we need to collect as many as possible information of target. WikiMiner[2] (Milne, 2008) is a useful tool to mine the linkings of entities in Wikipedia. We employ it to find alternate names of target entity based on the resources of Wikipedia to improve recall. Besides wikiMiner aliases, name variants are also taken into consideration. Suname, lastname, and different completeness of renderings of a person name are alternate_name candidates for PERSON type target. Acronym, fullname or organization name without determiner (e.g., the, a, an, etc.) are potential alternate names for ORGANIZATION type target. Name Expansion makes an important role in slot-filler extraction component, whose quality determines the coverage of subsequent extraction.

We use Lucene to index source corpus and search the occurrence of expanded names of target entity. Finally the patterns learned offline is applied to the returned documents with target mentions to extract filler candidates in this module. The document containing the target is returned by Lucene. Here a candidate may be extracted by more than one pattern. These patterns are treated as a support set for the candidate they extracted.

2.3 Post Processing

The last module is used to filter out redundant, wrong and unreliable filler candidates and then rank the left, which is important to achieve a high precision. This process can be further divided into three parts: candidate probability computation, candidate validation and duplicated candidate removal, which are illustrated at downright in Figure 1.

C**andidate Probability Computation.** Suppose a candidate slot value is supported by a set of patterns *S* which are used to extract this candidate. We use the following formula to compute the probability of this candidate.

$$prob(candidate_j) = \sum_{i=0}^{|S|} confidence(pattern_i)$$

This value serves as a confidence value for the candidate, which is also the weight used for final ranking.

Candidate Validation. According to our knowledge, we cast some constraints on values for certain slots as shown in Table 1. Take slot "per:age" as an example, the part-of-the-speech (POS) tag of candidate should be 'Noun', and NE type should be 'NEduration'. For "org:website", pos tag is 'String', NE type is 'NEurl'.

Duplicated Candidate Removal. Slots like "per:age", "org:date_founded" are single-valued, others like "per:origin", "org:shareholders" are list-valued. The single filler is required by the single-valued slot, multiple answers are permitted by list-valued slots. But redundant answers should be eliminated. This part is responsible for removing duplicated fillers for a slot.

[2] http://wikipedia-miner.cms.waikato.ac.nz/

3 Results and Discussion

We submitted three runs that differ in filtering strategy. Here we only analyze results of our best run which has 1183 no-NIL responses with 135 correct fillers. In order to assess the performance of system, we compute Precision(P), Recall(R), F-measure(F) according to the following formula:

$$P = \frac{correct}{noNIL}$$

$$R = \frac{correct}{correct + mis\sin g}$$

$$F = \frac{2PR}{P + R}$$

Because of the existence of list-valued slots, the number of missing may not be exact and is lower than the real missing value. This causes the estimated recall being optimistic. But this consequence is equal to all list-valued slots. Hence evaluation results are still valuable.

To find the problem of our system, we compute P, R and F for every target and show the best and worst results for PER and ORG type in Table 1. Generally, our system has higher precision for PER and higher recall for ORG. From overall F value, we can see that the performance on ORG is better than that on PER.

Type	SF_ID	Precision	Recall	F
PER	10(best)	50	50	50
	28(worst)	11.11	6.25	8
	overall	16.43	9.35	11.92
ORG	80(best)	100	37.5	54.55
	52(worst)	1.79	25	3.33
	overall	9.83	42.58	15.98

Table 1: Experimental results for target entities on official evaluation

In TAC-KBP task, the ratio of single-valued/list-valued slots is 11/15 for PER and it is 7/9 for ORG. Meanwhile, list-valued slots have multiple answers. Hence, the quality of list-valued slots extraction dominates the performance of a system. To assess the performance of our system, we compute P, R and F for single-valued and list-valued slots shown in Table 2. We can tell our system has higher precision for single-valued slot than that for list-valued slots. But the recall for single-valued slot is quite low. From F value, we may infer that our system probably has a relatively better performance on list-valued slots than on single-valued slot. Although the recall of list-valued slots is higher than the real value, the overall recall is larger than that of single-valued slot. Hence, this conclusion is still true.

Slot Type	Precision	Recall	F
Single-valued	13.04	5.33	7.57
List-valued	11.25	25.84	15.68
overall	11.3924	8.74919	9.89736

Table 2: Experimental results for slot attributes on official evaluation

In order to improve the system, it is also necessary to estimate the performance of our system on every slot. The P, R and F of slots with correct responses are shown in Table 3. From these 11 slots with corrects, we find that only org:date_founded (bold) is single-valued slot. The precision of org:founded_by is extremely low. Slots do not appear in Table 3 can be divided into two sets. One set contains slots with responses but none of them is correct, like per:member_of，per:spouse，per:origin，per:religion，per:age，er:country_of_birth. The other set contains slots without any responses.

SlotName	P	R	F
per:statesorprovinces_of_residence	42.86	10	16.22
per:cities_of_residence	15.09	40	21.92
per:title	32.95	78.38	46.4
per:employee_of	5.04	54.55	9.23
org:top_members_employees	13.91	90.62	24.12
org:subsidiaries	9.63	94.74	17.48
org:founded_by	0.47	33.33	0.92
org:date_founded	8.7	66.67	15.38
org:country_of_headquarters	13.64	42.86	20.69
org:stateorprovince_of_headquarters	13.64	50	21.43
org:city_of_headquarters	20	57.14	29.63

Table 3: Experimental results for slots with correct answers of our system on official evaluation

Based on the above experimental results, we find the following problems affect the performance of our system:

1. Pattern representation. Through checking the pattern learned by our system, we find that some patterns are too specific to be used in

filler extraction. For “per:title”, a pattern like this is extracted:

“<TARGET>
[^<]*?(?:<NEacademicTitle>)?[^<]*?<NEorganization> <VALUE> <NEperson>”,
(5 corrects and 50 wrongs)

This pattern has a low confidence and it is rare to find matcher of this pattern in documents. Conversely, some patterns are too short to confine the semantics. For “org:founded_by”, our system learned a pattern like this:

” <TARGET>, <VALUE_NEperson>”,
(1 correct and 1 wrong)

This pattern is supported by 1 correct and 1 wrong extraction. It may generate wrong fillers, as from the pattern itself, we cannot infer much information about “org:founded_by”.

2. Data sparseness. For some slots, there are too little training data to learn a pattern. Slots facing this problem are “per:date_of_death”, “per:cause_of_death” and “org:dissolved”.
3. Name expansion. We cannot predict the correct variants of names, which need to be further validated in the collection.
4. Some wrong answers. We cannot enumerate all cases of wrong answers which make filtering is not adequate. We use type checking, but NER tool that usually introduce noises like wrong border of filler, limited types and wrong types.

4 Conclusion

We describe our sweat2012 system that extract English slot filler for the given targets from Source Corpus which is obtained from the web, as defined in Knowledge Base Population task. We use pattern based method to extract slot fillers which involving learning patterns offline, extracting and validating candidate online. It consists of three modules: pattern learning, slot filler extraction and post processing. The pattern learning module is the core of our methods. The slot filler extraction module is the online part to handle every target entity. The post processing module checks whether the type constraints of attribute are satisfied by extracted values and ranks the outputs. The official evaluation shows that our system is the median system with P=11.3924, R=8.74919 and F=9.89736.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61070106), the National Basic Research Program of China (No. 2012CB316300), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA06030300) and the Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation.

References

Sergey Brin. 1998. Extracting patterns and relations from the World- Wide Web. In Proceedings of the 1998 International Workshop on the Web and Databases (WebDB’98)

Eugene Agichtein and Luis Gravano. 2000. Snowball: extracting relations from large plain - text collections. DL '00: Proceedings of the fifth ACM Conference on Digital Libraries, 85-94

Dekang Lin, Patrick Pantel. 2001. DIRT: discovery of inference rules from text. KDD 2001

Ndapandula Nakashole, Gerhard Weikum, Fabian Suchanek. 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. EMNLP 2012.

David Milne, Lan H. Witten. 2008. Learning to link with Wikipedia. CIKM’08