Filling a Knowledge Graph with a Crowd

GyuHyeon Choi, Sangha Nam, Dongho Choi, and **Key-Sun Choi**
Machine Reading Lab, School of Computing,
Korea Advanced Institute of Science and Technology (KAIST),
Daejeon, Republic of Korea
{wiany11, nam.sangha, zmal0103, kschoi}@kaist.ac.kr

Abstract

Building accurate knowledge graphs is essential for question answering system. We suggest a crowd-to-machine relation extraction system to eventually fill a knowledge graph. To train a relation extraction model, training data first have to be prepared either manually or automatically. A model trained by manually labeled data could show a better performance, however, it is not scalable because another set of training data should be prepared. If a model is trained by automatically collected data the performance could be rather low but the scalability is excellent since automatically collecting training data can be easily done. To expand a knowledge graph, not only do we need a relation extraction model with high accuracy, but also the model is better to be scalable. We suggest a crowd sourcing system with a scalable relation extraction model to fill a knowledge graph.

1 Introduction

Existence of good knowledge graphs is essential for question answering system. Due to inefficiency of manually adding triples to a knowledge graph, researches about extracting triples from raw text have been being conducted. Relation Extraction (RE) is a task to extract relational facts from unstructured text in triple format. For example, the triple (George W. Bush, parent, George H. W. Bush) can be extracted from the sentence "A president of USA George W. Bush is the son of George H. W. Bush." Various approaches have been researched for RE. We are focusing on two different paradigms here.

Fully supervised approaches require sufficiently many handcrafted training data. A triple is labeled to a sentence if the sentence expresses a certain relational fact between two entities. This kind of manually labeled data rarely has noise, however, it is very costly because annotation takes a lot of time and effort from experts. Manually annotated training data are good in quality but limited in quantity. Furthermore, a new training dataset may be needed for another corpus if the corpus has different characteristics (Zhou et al., 2005). Thus, this model is not scalable.

Another type of approaches makes use of seed patterns. Using seed patterns, triples are first extracted with sentences they are extracted from then extracted triples and sentences are used to generate new patterns. By repeating this process, more and more patterns are collected. This kind of approaches can be an alternative solution when manually annotated training data are not available, however, the noise in triples and sentences tends to propagate over iterations (Bunescu and Mooney, 2007; Pantel and Pennacchiotti, 2006). This model is scalable because it can generate data by itself for the next training but the problem is low accuracy of extraction.

We want a knowledge graph to contain only correct triples. Crowd sourcing is the first alternative to validate triples before they are uploaded into a knowledge graph. We want to take one more advantage in this triple validation process; not only for deciding whether to upload triples or not but also for training a relation extraction model. Needless to say, a better relation extraction model improves the efficiency of crowd sourcing; a larger portion of triples fed back by crowds are valid to be uploaded. We propose a crowd sourcing system that utilizes feedback of crowd's for both triple validation and model training.

Proceedings of the Open Knowledge Base and Question Answering (OKBQA) Workshop,
pages 67–71, Osaka, Japan, December 11 2016.

Figure 1: Description of pattern generation from a dependency tree

2 Model Explanation

A model trained by our system consists of patterns and these patterns are extracted from training data; pairs of sentence and triple. To filter out bad patterns, our system uses feedback from a crowd. We will explain how easily a crowd feeds back and our system updates a model later. To maximize efficiency of crowd sourcing, we prefer to get as much feedback as possible per pattern. In other words, we want to have as few patterns as possible. We will also explain the method we used to regulate the number of patterns later in this section. First, we will give a brief explanation about how a pattern is generated in the following subsections.

2.1 Pattern generation

For scalability, our systems uses distantly supervised data to train a relation extraction model (Mintz et al, 2009). Since Distant Supervision uses an existing knowledge graph to collect sentences for training, there is a possibility to gather more sentences as a knowledge graph grows. From a pair of sentence and triple, a pattern can be generated. If named-entities repeatedly appear in a sentence, all possible subject and object pairs are listed first.

Figure 1 describes how a pattern is generated from the sentence ”A president of USA George W. Bush is the son of George H. W. Bush.” labeled with the subject and object pair (George W. Bush, George H. W. Bush) by Distant Supervision. A pattern is generated using basic Natural Language Processing; dependency parsing and Part Of Speech tagging. In a dependency tree, our system finds the first common predicate which is the first common parent node for both subject and object nodes in a dependency tree. Our system generates a pattern using subject, object, the first predicate node, and their incoming and outgoing nodes. Attributes of pattern are lemmas of the nodes. For nonexistent incoming and outgoing nodes, empty attributes are added to a pattern. Intuition behind this pattern generation is that a key word in a proper position plays an important role to express a relation between subject and object entities. In Figure 1, two consecutive incoming and outgoing nodes are considered.

2.2 Pattern expansion

As mentioned earlier, we want to keep as few patterns as possible for efficient crowd sourcing. It means that our model has to quickly filter out as many bad patterns as possible. To accomplish this goal, Our system uses the special model shown in Figure 2. This model has tree structure and each node has four elements; a pattern, remaining attributes, accuracy, and expandability. A new pattern can be added only as a child node of an expandable node. Thus, expandability is the key for regulating the number of patterns.

The root node is always expandable and has negative accuracy. Other than the root node, accuracy of each node is defined from a crowd’s feedback; sample evaluation. Our system can extract triples from sentences using a specific pattern and vice versa. So if a crowd feeds back by evaluating samples, our system can backtrack a pattern and update its accuracy. For example, our system extracts the triple (*A*, *parent*, *B*) from the sentence ”*A* is the son of *B*.” using a certain pattern *P*. A crowd may positively evaluate this sample. Then our system adds one more positive answer to feedback history of the pattern *P* then calculates a new accuracy. Expandability is also updated.

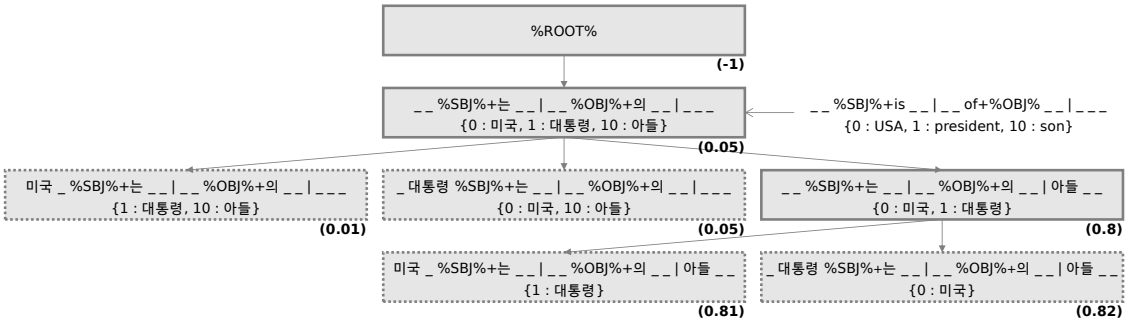

Figure 2: Example of pattern expansion

Now we can track back to the model in Figure 2. The model is grown up from the pattern in Figure 1. First, the pattern is generalized. A generalized pattern contains only two attribute; subject and object with preposition or postposition if any. Attributes other than those two are remaining attributes and separately stored with their positional information. Then RECrowd extracts sample triples with corresponding sentences using the generalized pattern then asks a crowd to evaluate. Whatever accuracy the pattern gets back, the node is expandable since the root node has negative accuracy. Then it can produce a child pattern using its remaining attributes. Next, the first child pattern is added. It has one more attribute than its parent pattern. Since its accuracy is not higher than its parent pattern, it becomes not expandable. The second child pattern is also not expandable for the same reason. We can guess that attributes 'USA' or 'president' do not give much hint to the relation *parent*. However, the third child pattern with additional attribute 'son' has much higher accuracy and remains expandable. Then it can produce next child patterns as you can see. But they are not expandable so the model does not grow anymore.

In pattern matching, an empty attribute means that any lemma can come to its position. This is why a parent pattern is more general than a child pattern. A child pattern unconditionally extracts less triples than its parent pattern. Then the only advantage we can expect when we add a new child pattern to a model is that the child pattern extracts triples with outstanding accuracy. If accuracy does not increase much from parent pattern to child pattern, disadvantage of computational cost cancels out the advantage of having one more pattern. So it is reasonable to stop if a child pattern does not have much higher accuracy. We can define the expandability like the following:

$$Expandability := \left(\frac{\Delta Crowd\ Accuracy}{\Delta \parallel Extracted\ Triples \parallel} \geq threshold \right) \tag{1}$$

3 System Workflow

Workflow of our system is shown in Figure 3. Since our system assigns a different set of patterns for each relation, the workflow starts with relation selection. After a specific relation is chosen, one random pattern is pulled out from patterns to learn or learned patterns. Patterns learned indicate patterns which have been fed back from a crowd so they already have accuracy and expandability. Patterns to learn mean patterns which have not yet received any feedback from a crowd. Once a pattern to learn got feedback from a crowd, then pattern will be added to patterns learned with accuracy and expandability. Using the selected pattern, our system can extract triples from sentences and sample a few triples with corresponding sentences for a crowd to evaluate. A crowd only needs to read a sentence then checks an associated triple is correctly expressed in the sentence. A crowd simply answers whether each triple extraction is correct or not. Since feedback procedure is very easy like this, an inexpert crowd can interact with the system. Extraction with positive answers are uploaded into a knowledge graph. Our system also keeps saving up a crowd's true-false feedback and updating accuracy and expandability. Ancestor and descendant patterns could be recursively updated when certain expandability is changed. Once again a relation extraction model is continuously trained while a knowledge graph is expanded.

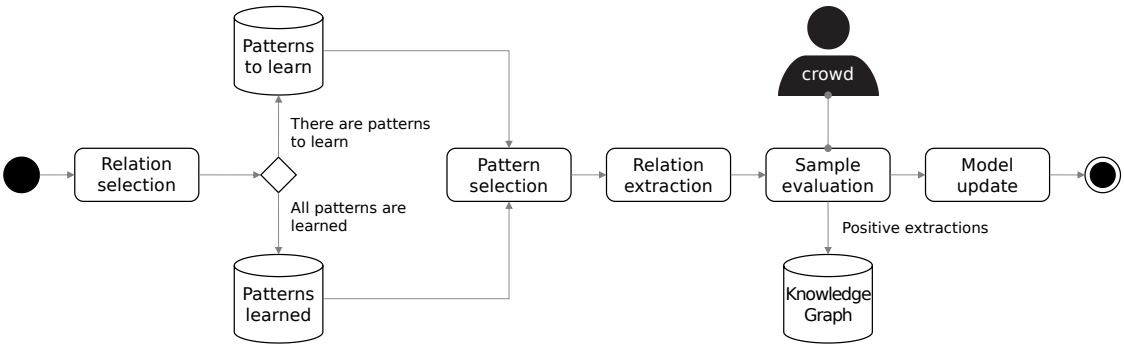

Figure 3: RECrowd workflow

4 Experiment

To see if a relation extraction model in our system is really trained, a small-scale experiment was conducted. Our main focus is to manually trace if a model finds out valid patterns as it grows. We want to guess if a model is capable to achieve this goal when it grows big by inspecting a mini model.

We trained a model for the relation *parent* using 10 randomly chosen sentences from 2400 Korean sentences collected from Korean Wikipedia and Korean DBpedia using distantly supervised approach (Mintz et al, 2009). In training phase, total 112 patterns are generated from the 10 sentences. After 320 sample evaluations were fed back, however, 48 patterns were filtered out as child patterns of not expandable patterns. The trained model contains 64 patterns and 21 patterns were still expandable. In result, all meaningful patterns are still contained in the trained model. Since 320 evaluations are quite small in crowd sourcing, it is a promising result. As Figure 2 shows, a model in our system has a tree structure without much depth. This means that this model quickly reaches to meaningful patterns then efficiently filters out meaningless patterns using stopping condition, the expandability. The rate of filtered out patterns will be larger in a bigger model. In addition, 75 extractions with positive feedback are simply uploaded to a knowledge graph.

5 Conclusion

Existence of good knowledge graphs is essential in question answering system. To upload only correct triples, crowd sourcing may be necessary to validate triples. Our system makes one more use of crowd sourcing; not only to find correct triples to add into a knowledge graph but also to train a relation extraction model. A well trained relation extraction model maximizes efficiency of crowd sourcing; more extractions get positive feedback from a crowd then more triples can be uploaded. Our system is developed using Korean NLP (Natural Language Processing) tool and tested on Korean Wikipedia dump. You can see the demonstration in our demo web page[1].

Acknowledgement

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. R0101-16-0054, WiseKB: Big data based self-evolving knowledge base and reasoning platform). This work was supported by the Bio & Medical Technology Development Program of the NRF funded by the Korean government, MSIP(2015M3A9A7029735).

References

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. *Exploring various knowledge in relation extraction*, In ACL 05, pages 427-434. Ann Arbor, MI.

[1] http://143.248.135.210/recrowd

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. *Distant supervision for relation extraction without labeled data*, In ACL 2009, pages 1003-1011. Stanford University, Stanford, CA 94305.

Patrick Pantel and Marco Pennacchiotti. 2006. *Espresso: leveraging generic patterns from automatically harvesting semantic relations.*, In COLING/ACL 2006, pages 113-120. Sydney, Australia.

Razvan Bunescu and Raymond Mooney. 2007. *Learning to extract relations from the web using minimal supervision*, In ACL 2007, pages 576-583. Prague, Czech Republic, June.