Practice and Exploration of Ontology Creation Algorithms

Tianyu Li
University of British Columbia, Vancouver, BC, Canada
lty419@cs.ubc.ca

Abstract

Ontology creation from unstrcutred text is a meaningful and challenging problem. In order to see what kind of ontology is produced when applying different algorithms on specialized domain corpus, we select two representative approaches, one is Guided Hierarchical Clustering and the other is lexico-syntactic pattern based, which corresponds to two main stream directions in this field. In particular, we do not set constraints for the length of extracted concepts in the hope of discovering richer ontology. Applying these two methods to a Civil Engineering dataset, we evaluate their result in terms of precision and taxonomic metrics. Output relationships are examined and analyzed, to support our idea while providing lessons for implementation of pattern based ontology extraction.

1 Introduction

An ontology in computer science is a specification of conceptualizations of a domain of interest that are shared by a group of people (Biemann, 2005). There is no universal standard with respect to the components of an ontology. However, a common ontology will at least include concepts and hierarchical relationships among them, which actually form the taxonomic backbone of an ontology.

Building ontology from un-structured or semi-structured text can bridge the gap between human-readable data and machine-readable knowledge in a specific area. It benefits the navigation and discovery of information, and can be used in many other tasks as a knowledge base. As we all know, a hand-crafted ontology of words like WordNet [1] has limited coverage, especially in specialized field, where jargons and terminologies can have very different meanings from their dominant senses in general domain. Also it takes a lot of human effort and expert interference to build an ontology from a corpus of any new domain manually, which makes the problem of automating the ontology extraction process very meaningful. This problem is essentially challenging as it requires good understanding of the text and smart application of state-of-art Natural Language Processing techniques .

Rather than just looking for ISA relationships between pairs of terms, taking some time on pondering how to extract concepts with their semantic relationships and represent them at appropriate level of abstractness is worthwhile. However, current methods of creating ontology from text, usually bypass or simplify this problem by focusing on single-word-term or very short phrases. They will fail to capture many meaningful and distinct concepts in this domain that we will see in the remainder of the paper. The resulting ontology is usually trivial or redundant from the general thesaurus, which will be reflected in our experiments. In this project, we take a first step towards this direction by trying out a simple pattern-based approach on a Civil Engineering data set, and observing the extracted pairs of phrases having ISA relationships between each other. A representative algorithm in previous work is applied and compared as well. In this sense, this paper is more like a case study of different ontology extraction algorithms on text of specific domain.

The following sections are structured like this: Section 2 classifies and describes related work. Section 3 introduces the two methods that we implemented in detail. We discuss the implementation details in Section 4 and show the experiment results and lessons learned in Section 5. Finally we conclude the work and propose future direction.

[1] http://wordnet.princeton.edu/

2 Related Work

In Biemann's (2005) survey of ontology learning algorithms, he classifies ontologies into three kinds:

- A **Formal Ontology** is a conceptualization whose categories are distinguished by axioms and definitions.They are stated in logic that can support complex inferences and computations.
- **Prototype-based Ontologies** are distinguished by typical instances or prototypes rather than axioms and definitions in logic.
- **Terminological Ontologies** are partially specified by subtype-supertype relations and describe concepts by concept labels or synonyms.

YAGO (Suchanek et al., 2007) is a system that automatically extracts facts from Wikipedia[2]and unifying them with WordNet. It defines an ontology language by extending an existing one called RDFS, and produces a formal ontology using Wikipedia individuals and categories and WordNet synsets as basic components of concepts. YAGO heavily depends on the hierarchical structure within WordNet and can be considered as enriching WordNet with Wikipedia entities. It is usually too hard to produce a formal ontology if the task is to extract it from the text and build it from scratch, especially when the automatic building process is actually used as a first step for further refining steps (we have to admit none of these current automatic algorithms can produce high-quality ontologies that can be used in practice right away). Most state-of-art approaches aim at constructing ISA-related concept hierarchies, that either fall into the Prototype-based Ontology or the Terminological Ontology field.

Biemann (2005) also concludes three main directions for the approaches to ontology learning from unstructured text.

1. Use Harris' distributional hypothesis, which states that similar words tend to occur in similar contexts. They employ different kinds of clustering algorithms to build synset-like concept, which is a group of similar words.
2. Use patterns such as Heart Patterns (Hearst, 1992) that explicitly grasp a certain relation between words. They (Pennacchiotti and Pantel, 2006) (Sanderson and Croft, 1999) usually follow an iterative process with bootstrapping, but can still suffer from low recall.
3. Using the World Wide Web either as additional resource or as the main source of information is often a possibility to avoid data sparseness problem.

[2]http://www.wikipedia.org/

Some algorithms, such as (Cimiano and Staab, 2005) (Caraballo, 1999), try to combine all or some of the above clues to achieve richer and more reasonable result.

Most of the clustering-based algorithms produce prototype-based ontology, which poses another problem: how to label each group of word so that it can be easily understood and used. Some other work, including (Zavitsanos et al., 2010), use topic modeling based method to extract concepts and represent concepts as latent topics which are distributions of words. However, for the practice use and evaluation consideration, we will focus on those methods producing terminological ontology, which consists of terms rather than latent topics or word clusters.

3 Practices of Clustering and Pattern Based Algorithms

As we mentioned in last section, clustering based and pattern based approaches are two mainstream methods of learning ontology from text. In order to see their performance on a domain-specific corpus, we implement two algorithms and apply them to an architecture data set, which consists all kinds of documents produced during the civil engineering process. The Guided Hierarchical Clustering algorithm is a clustering based method, while the other is just a simple implementation of lexico-syntactic patterns first proposed by Hearst (Hearst, 1992). In the remainder of these section, we will introduce the two algorithms in detail.

3.1 Guided Hierarchical Clustering

The Guided Hierarchical Clustering algorithm (Cimiano and Staab, 2005)is a representative state-of-art algorithm as it combines all three directions in ontology learning, as we mentioned in Section 1. It relies on the distributional similarity of terms with respect to an underlying corpus. For each term, they extract syntactic surface dependencies by matching regular expressions over part-of-speech tags. The similarity between terms can be calculated between the vectors of dependency features including adjective modifiers, prepositional phrase modifiers, noun phrases in subject or object position, and etc.

With the similarities between pairs of terms, an agglomerative clustering algorithm is driven by hypernyms acquired by other means. Particularly, the author exploits hypernyms extracted from WordNet, and from matching lexico-syntactic patterns indicating a hypernym-relationship in the corpus as well as search engine documents.Figure 1 shows the architecture of the system.

The guided clustering algorithm works like this:

1. For a list of terms to be placed in the ontology, calculate the similarity between each pair of terms and

sort them.

2. Pick a pair of terms with highest similarity in the remaining list of pairs to be clustered, if either of them is not placed in the ontology. Try to put them in the right position of the growing ontology based on the their common hypernym evidences found from WordNet, matched patterns in Corpus and matched patterns in WWW, following some predefined rules. Otherwise, the pair get clustered.

3. Place the clustered pairs and single terms that do not get into the ontology in the most appropriate position to make sure the resulting ontology is a connected hierarchy.

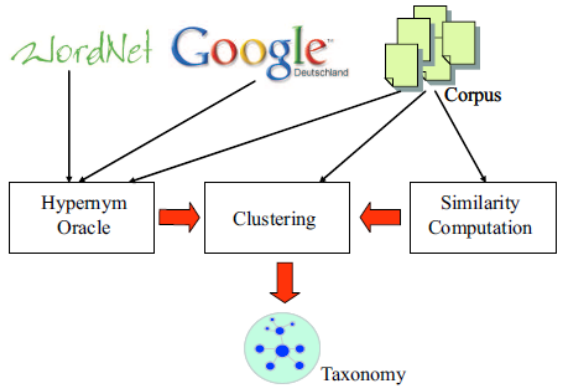

Figure 1: System Overview of Guided Hierarchical Clustering Algorithm (Cimiano and Staab, 2005)

Here we introduce the lexico-syntactic patterns used by this algorithm and in our algorithm. These patterns are first proposed by Hearst (1992) and augmented by many following papers. Seven of them are listed below.
(1) NP_0 such as $NP_1, NP_2, ..., NP_{n-1}$ (and|or) NP_n
(2) such NP_0 as $NP_1, NP_2, ..., NP_N$ (and|or) NP_n
(3) $NP_1, NP_2, ..., NP_n$ (and|or) other NP_0
(4) NP_0, (including|especially) $NP_1, NP_2, ..., NP_{n-1}$ (and|or) NP_n
These four are originally from the Hearst paper while the following three are extended ones.
(5)NP_1 is NP_0
(6)NP_1, another NP_0
(7)NP_0 like NP_1

In these patterns, NP stands for a noun phrase. We can derive that for all $NP_i, 1 \leq i \leq n, isa(NP_i, NP_0)$. However this paper and many other papers use the inferred rule that removes modifiers of any noun phrase, $NP_i, 1 \leq i \leq n, isa(head(NP_i), head(NP_0))$, in order to apply these patterns on single-word terms or very short phrases.

3.2 Pattern-based Extraction of Concepts and Semantic Relations

As we have seen in the last section, the way of removing all modifiers when applying the Hearst pattern are employed by many other papers such as (Caraballo, 1999). It does improve the recall when you are checking if a pair of terms appear in this kind of patterns, especially the low recall is always a shortcoming of pattern-based approaches. However, Hearst (1992) did mention in her paper that how much modification is desirable depends on the target domain.For building up a basic, general-domain thesaurus, single-word nouns and very common compounds are most appropriate. For a specialized domain, more modified terms are necessary. In our architecture dataset, we will lose a lot of concepts if we just focus on single-word or very common compounds as many concepts cannot be described in one or two words. For example, "finishes which may radiate noise" and "retardants with bromine or chlorine" are snippets from our document that represent very concrete concepts. It would be meaningful if any ontology learning algorithm can discover these knowledge, instead of just finding general word "finish" and "retardant". In our minds, a good ontology for a specialized domain should be rich enough to have actual use, while not being too specific that cannot be applied elsewhere. Even the standard of a good ontology is a philosophy problem, we move ahead to the direction of "rich ontology" by extracting instances of these lexico-syntactic patterns without setting the constrains on the length of terms. We use the same list of patterns but in different ways, which we will see in next section.

4 Implementation Details

In this section, we discuss the details and issues coming up during the process of implementation and evaluation.

4.1 Data Set Description and Preprocessing

The dataset used in this case study is called AEC(Architecture, Engineering and Construction) data. Belonging to the ARTIFACT Project [3] in Data Management Lab of UBC, it contains all kinds of data generated the process of civil engineering, such as scheduling data, 3D design data, meeting notes and reports. The files are in PDF format and text is mixed with graphs. Since we only want to work on the text part, we apply the following preprocessing step, which is shared by the two algorithms.

1. Convert the PDF file to txt format using the A-PDF

[3] http://www.cs.ubc.ca/labs/db/research.php/artifact

Text Extractor Command line (PTCMD). [4]. This step introduces a lot of noise because of the imperfect accuracy of the program and the fact that text is mixed with graphs.

2. Apply a set of regular expression based processing methods to remove noises from the text. This part is implemented in Java and the rules are customized to this data set after observing abnormal data in the corpus.

3. Detect the sentence boundaries within processed text and chunk the text into sentences using LingPipe[5] tool kit, which is a Java library for a set of NLP tasks.

After this step, we get 59,497 sentences in total.

4.2 Applying GHC to AEC Data

We get the author's algorithm package for the Guided Hierarchical Clustering algorithm from his website [6]. In order to prepare the input to the algorithm as required, we tag the whole corpus with TreeTagger [7]. It is recommended by the user. We stick to it to keep the consistency of the tag set used in the algorithm, even though this tagger is not capable of handling larger input one time which means we have to do tagging in batch.

As for the input terms to be placed in the ontology, the author does not clarify on what criteria we could select the terms. So we follow a simple and reasonable strategy, that we pick all terms with one of these POS tags("NP","NPS","NN","NNS") and order these potential nouns by their frequency in the corpus. Then we select the top 100 terms excluding the stop words and use the GHC algorithm to organized them into an ontology.

As for the algorithm setting, the only parameters are the combinations of three hypernym evidences. We cannot run the part that employing a search engine to check if a pair of terms have ISA relationships with each other, as it requires the Google Search API key, which Google has stopped to provide. We just use the "WordNet + Hearst Pattern in Corpus" combination, since it is one of the best setting according to the author's experiments.

4.3 Ontology Extraction from AEC Data by Pure Patterns

The GHC algorithm finds the instances of these patterns in Section 2, by matching each Part-of-Speech tagged sentence with regular expressions. They use hand-crafted rules over combinations of tags to determine if a text segment is a NP. For example, $(DT \setminus t(\setminus w+))?(JJ \setminus t(\setminus w+))?((NN(S?) \setminus t([a-z]+) \setminus s?)+)$ is used to determine a non-recursive NP. This kind of strategy has these constraints:

1. The inaccuracy of POS tagging will influence the pattern matching, and the effect can be chain-reaction that fail the matching on a whole sentence.

2. The strict matching may fail to capture some pattens that hidden in the proposed patterns. For example, we find this pattern "including but not limited to" that matches the Pattern (4) with the common connector "including", but will get excluded if we strictly require a NP immediately following the connector "including".

3. Simple rules over POS tags cannot identify some NPs that having complex modification structures, which can be an obstacle for discovering rich ontology as we suggested in Section 1.

In order to overcome these limitations, we relax the matching rule by first only matching lexical connectors, then extract the corresponding NPs from the text segments surrounding those connectors or in between, by feeding those text in a parser. We believe that a probabilistic parser like Stanford Parser [8] can do a better job on determining a NP, than hand-crafted rules over POS tags. For example, for Pattern (1), we extract three text segments, which are on the left of "such as", between "such as" and "and | or", and on the right of "and | or" respectively. Then we extract a NP from the first segment, a list of NPs from the second segment and another NP from the third one by analyzing the parse tree of each segment.

As the extracted text segment is always accompanies with unwanted text from other parts(other clauses or predicates) of the sentence, we use some heuristic rules to extract corresponding NPs from the parse tree. We conclude four cases depending on the different positions these NPs lie in and list them as below:

1. *Leading NP:* corresponds to NP_0 in Pattern (1)(4)(7) and NP_1 in Pattern (5)(6). Extracting them by in the parse tree identifying the minimum NP containing the word immediately preceding the connectors.

2. *Leading NP List:* corresponds to the list $NP_1, NP_2, ..., NP_n$ in Pattern (3). First get the largest NP containing the word immediately preceding the connectors, then split the clauses separated by commas in the corresponding text.

3. *Trailing NP:* corresponds to NP_n in Pattern(1)(2)(4), NP_0 in Pattern (3)(5)(6) and

[4]http://www.a-pdf.com/text/cmd.htm

[5]http://alias-i.com/lingpipe/

[6]http://www.cimiano.de/doku.php?id=olp

[7]http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger

[8]http://nlp.stanford.edu/software/lex-parser.shtml

NP_1 in pattern (7). We extract them by identifying from the parse tree the maximal NP containing the word immediately following the connectors.

4. *Center NP List:* corresponds to the list $NP_1, NP_2, ..., NP_n$ $-$ 1 in Pattern (1)(2)(4). For the clauses immediately following or preceding the connectors, we extract largest NP from each of them, for other clauses in between, we return as they are.

5 Evaluation Results and Lessons Learned

In this section, we show our experiment results by presenting and analyzing ontologies produced by two algorithms respectively. We also do comparisons in terms of precision and taxonomic metrics including depth and number of children. Then we conclude some lessons learned from implementing pure pattern based method and analyzing the negative examples.

For each output pair of terms/concepts, we determine if the ISA relationship is valid manually, given the context containing this pair as a reference. Since many terms and phrases are very domain-specific, we use some external resources such as search engine to identify their senses, ensuring we make as few mistakes as possible.

5.1 Resulting Ontology Produced by the GHC

As the output size of GHC is controllable, (determined by the terms to be clustered), we only generate the ontology consisting of the top 100 terms with highest frequency in the corpus, in order to ease the manual validation. Figure 2 shows the ontology produced by the GHC, and only validated relationships are included.

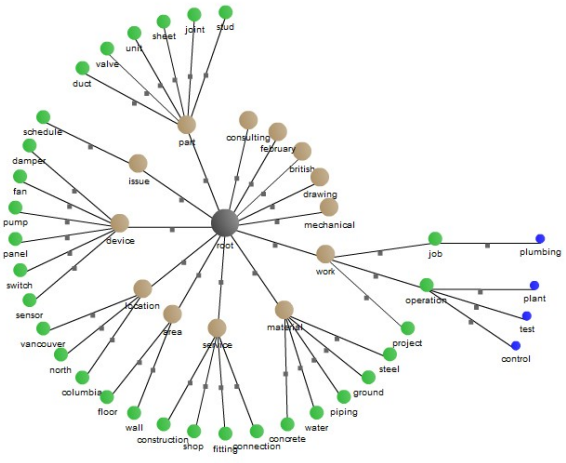

Figure 2: Resulting Ontology Produced by the GHC

What surprises us is the precision performance, only 39 pairs out of 100 are valid even including the trivial pairs from the "root" to any terms, is way lower than the author's report average value in this setting, which is above 0.6. The possible reasons are: the data set is smaller than those data sets author experimented on, so that both syntactic features and pattern evidences will suffer from the sparsity problem; the selection of terms is not fine tuned that high-frequency terms do not necessarily clustered together.

Also the ontology learnt this way can be criticized for being too general. Relationships like "floor **ISA** area", "fan **ISA** device" and "concrete **ISA** material" are valid yet too trivial in a professional point of view, as they can easily found in general thesaurus like WordNet. The whole algorithm is just doing word sense disambiguation in this sense.

According to our experience of applying the GHC algorithm may not able to discover rich concepts and interesting relationships between them, especially when the data set is relative small. Clustering algorithms based on distributional similarity tend to suffer from sparse features.

5.2 Concept Pairs Extracted by Patterns

The seven patterns generate 652 pairs of concepts with 279 valid pairs, which is better than expectation as we do not apply any post-processing on the data. It is not hard to imagine even a simple strategy that extends a pair of concepts using the ISA transitivity like this: for NP_1 **ISA** NP_0, add $head(NP_1)$ **ISA** $head(NP_0)$ and NP_1 **ISA** $head(NP_0)$, can boost the recall largely. Not to mention the trivial ISA relationships between a NP and its head word. The current recall is also good enough, as it is generated by a small data set, while ontologies used in many domain-specific tasks are not very large.

The performances of the seven patterns do varies, which reflects their different levels of effectiveness. Pattern (1) achieves best performance with a precision of 0.56 while Pattern (2) and (6) fail to find any valid instances. This confirms our intuitive guess that Pattern (1) is a strong indication of ISA relationship while some other patterns will suffer from their inherent rarer frequency (or we can say lower prior frequency). As a matter of fact, the precision of pattern-based approach can be even higher as most negative examples are resulting from wrong selection of NPs, we will see in a following section.

Here are some example relationships found by pattern based algorithms, apparently they are contain richer knowledge and novel in the sense you cannot find them easily in a general domain ontology or determine them by our common sense.

- TCPP **ISA** retardants with bromine or chlorine
- vinyl flooring **ISA** a flooring surface

- backflow prevention stations **ISA** equipment intended to change the pressure of the fluid
- Vulkem 245 **ISA** pre-approved sealant

The first and fourth relationship actually contains named entities, so that we can easily distinguish them as InstanceOf relation from ISA if we use an entity recognizer. The third one is also worth noticing that the parent concept here is accompanied with long modifiers which makes the concept rich and complex. This leaves us a problem, which Hearst (1992) also proposed without giving a clear answer, that how to represent these concepts with enough modifiers that can identify them yet concise enough as we don't want to have an ontology too specific that we cannot use elsewhere. Hearst mentioned that we can safely remove those modifiers that only express subjectivity of the speaker but not property of the concept, such as "beautiful", "important" and etc.

5.3 Comparison

We compare the performance of these two algorithms in terms of precision and taxonomic metrics including maximum depth, average depth and average number of children. We only consider the precision of GHC without counting the relationship involving "root" because it is trivial and the pattern based algorithm does not produce "root". We do not have metric recall as we are not supposed to know all ISA relationships embedded in this corpus, and we can not really have fair comparison for recall when we control the output size of GHC algorithm. The reasons we use these taxonomic metrics are:

1. Maximum depth and average depth of the taxonomy: the deeper the average depth of the taxonomy, the more complex the relationships.
2. Average number of children: a high average number of children represents a rich taxonomy.

We build the validated ontology by creating an edge from each parent concept to the child concept, and using a common "root" to connect all concepts which no other concepts pointing to them. In this way we can get a Directed Acyclic Graph, so that the depth and children metrics can be calculated.

Figure 3 4 5 show the results for each metric respectively. The actual numbers are presented in Table 1.

We can see that pattern-based algorithm outperforms GHC in terms of precision and get competitive depth and number of children. It is understandable the pattern-based algorithm does not do better on taxonomic metrics, as the ontology is made up of very preliminary output as we discussed in last section. The GHC has stable performance as for taxonomic metrics, since its goal is to build a connected ontology. We can actually consider

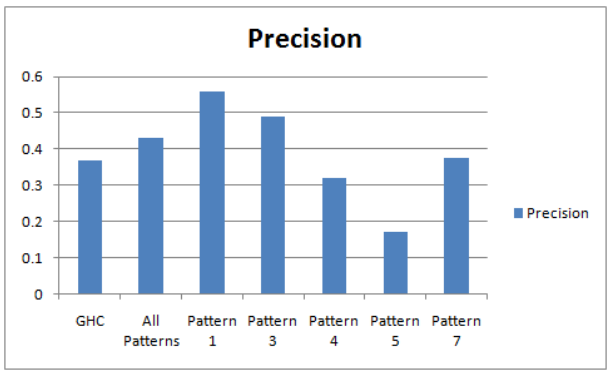

Figure 3: Precision of GHC and Pattern Based Algorithm Result

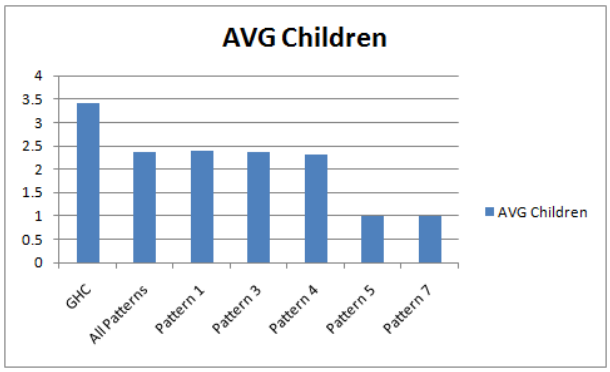

Figure 4: Maximum Depth and Average Depth of GHC and Pattern Based Algorithm Result

the well-structured ontology produced by GHC as a basis, and enrich this trivial ontology with longer concepts found by pattern-based algorithm. This kind of combination makes use of strong points from both approaches, and looks promising.

We also compare the performance of each pattern, with Pattern(2) and (6) omitted as they produce non valid relationships. Among all these patterns, Pattern (1) and (3) are the best ones. However we cannot assume these two patterns are better ones, because it might not be the case for other corpus. The prior confidence and frequency of any pattern can only estimated from corpus that are large enough.

5.4 Lessons learned

According to our analysis of negative examples in pattern-based algorithm output, we can conclude the factors that influence the extraction quality other than the inherent confidence of each rule.

- Wrong Sentence Boundary: the data after a series

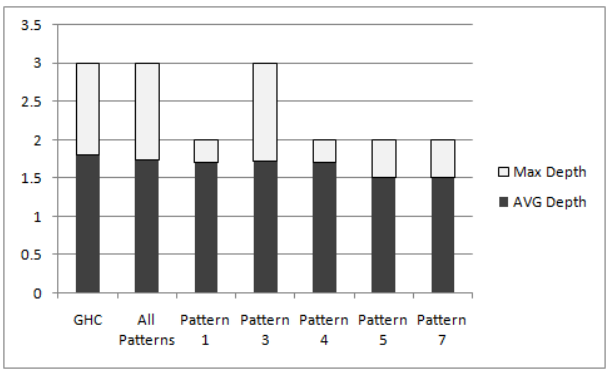

Figure 5: Average Number of Children of GHC and Pattern Based Algorithm Result

of pre-processing is still not noisy-free. The failure of sentence boundary detection will concatenate an irrelevant text segment that should have belonged to another sentence to the one we are extracting NP from. This kind of noise will confuse the parser by giving wrong result.

- Partial Parsing: As we discussed in Section 4, we feed the text segment into the parser instead of the whole sentence. We do this due to the consideration of length limitation the parser has.There is no document about its actual limitation but it does give error when parsing some whole sentences. However, a partial parsing sometimes fails to give correct parsing result without the grasp of the whole sentence structure. Here is an example, "X-ray all concrete walls, partitions, shafts, slabs and other concrete or concrete block assemblies prior to coring." When the text segment "concrete block assemblies prior to coring" will be determined as a NP, while "prior to coring" is actually a prepositional phrase that modifies the verb "X-ray" in the whole sentence. So we consider parsing the whole sentence in future work, as it is the best way to match lexico-syntactic patterns and extract desired NPs. We'll try to overcome the length limitation either by doing better sentence chunking or switching to a better parser.

- Greedy or Not: when extracting the *leading NP*, in the parse tree we identify the minimum NP containing the word immediately preceding the connectors. This kind of non-greedy NP matching works for this sentence, "Provisions of shading devices, such as overhangs or vertical fins, to let in quality natural light but exclude undesired direct sun light should be considered .", where "shading devices" is the parent concept that get specified, not "provision of shading devices". But it won't work for the sentence, "The work shall be carried out in accordance with the authorities having jurisdiction, including Ministry of Environment and the Workers Compensation Board of British Columbia and by contractors experienced in this specialty .", where "the authorities having jurisdiction" is desired rather than "jurisdiction". This paradox seems to imply that we cannot simply use uniform strategy (greedy or non-greedy) for all patterns. Through our observation, the leading NP in Pattern (4) favors greedy matching while the one in Pattern (1) prefers non-greedy matching.

- When There Are Too Many "and"s: "and" is an important connectors in all four Hearst patterns as it links NP with other NPs. But this sentence show what the real word looks like: "District energy controls, including demand **and** supply monitoring and flow sensing **and** metering." The bolded "and"connects noun modifiers within a NP, rather than playing a role of connectors in the pattern. No concrete ideas are proposed to solve this problem yet.

- More strict or relax constraints: As we mentioned in Section 4, patterns like "including but not limited to" will be lost if we follow strict pattern matching. But when we use somewhat relaxed matching, we get pattern like "such NP as to ...", which has nothing to do with ISA relationship. This is also the reason the Pattern 2 fails to produce any valid result when it claims to find 25 pairs. We might need to reconsider the constraints.

6 Conclusion and Future Work

Many state-of-art algorithms for ontology learning from text focus on organize single-word terms or common compounds into hierarchy, which usually result in general yet trivial ontology; the practice value is in question especially for a specialized domain. We propose that it is necessary to discover richer concepts in order to achieve a richer ontology. We follow a very classical pattern-based approach, without setting constrains on the length of extracted concept. Applying the pattern instance extraction on a Civil Engineering data set, we get very meaningful output concepts. Meanwhile, a representative guided clustering algorithm is practiced on the same data set. According to our comparison, our pattern based approach outperforms the guided hierarchical clustering algorithm in terms of precision, and achieves competitive taxonomic metrics. We also point out the output pattern instances are full of potential, either being post-processed or integrated with a general ontology produced by the hierarchical clustering algorithm, which are possible directions for future work. We re-ask the question that

how to represent concepts with long or complex modification structures at appropriate level of abstractness without losing the rich information they contain. This is an interesting and challenging problem, especially in today when building web of concepts are appealing. Finally we analyze the output data and conclude many valuable lessons from the practice of pattern instance extraction, which can never be learnt if no case study like this is done.

References

Biemann, C. 2005. *Ontology learning from text: A survey of methods*. *LDV Forum*, 20(2):75–93.

Caraballo, S.A. 1999. *Automatic construction of a hypernym-labeled noun hierarchy from text*. *In Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, 120–126.

Cimiano, P. and Staab, S. 2005. *Learning concept hierarchies from text with a guided hierarchical clustering algorithm*. *In ICML workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*.

Hearst, M.A. 1992. *Automatic acquisition of hyponyms from large text corpora*. *In Proceedings of the 14th conference on Computational linguistics*, 2(1992), 539–545.

Pennacchiotti, M. and Pantel, P. 2006. *A bootstrapping algorithm for automatically harvesting semantic relations*. *In Proceedings of Inference in Computational Semantics*.

Sanderson, M. and Croft, B. 1999. *Deriving concept hierarchies from text*. *In Proceedings of the 22nd annual international ACM SIGIR* , 206–213.

Suchanek, F.M. and Kasneci, G. and Weikum, G. 2007. *Yago: a core of semantic knowledge*. *Proceedings of the 16th international conference on World Wide Web*, 697–706.

Zavitsanos, E. and Paliouras, G. and Vouros, G.A. and Petridis, S. 2010. *Learning subsumption hierarchies of ontology concepts from texts*. *In Web Intelligence and Agent Systems*, 8, 1(2010), 37–51.

Table 1: Metrics and Taxonomic Metrics of GHC and Pattern Based Algorithm Result

	GHC	All Patterns	Pattern 1	Pattern 3	Pattern 4	Pattern 5	Pattern 7
Precision	0.37	0.43	0.56	0.49	0.32	0.17	0.375
Validated Pairs	34	279	99	117	55	5	3
All Pairs	93	652	177	240	172	30	8
Max Depth	3	3	2	3	2	2	2
AVG Depth	1.809	1.734	1.696	1.717	1.699	1.5	1.5
AVG Children	3.4	2.374	2.39	2.367	2.318	1	1