Efficient Extraction of Ontologies from Domain Specific Text Corpora

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

Pirooz Chubak
University of British Columbia
Vancouver, BC, Canada
pchubak@cs.ubc.ca

Laks V.S. Lakshmanan
University of British Columbia
Vancouver, BC, Canada
laks@cs.ubc.ca

Rachel Pottinger
University of British Columbia
Vancouver, BC, Canada
rap@cs.ubc.ca

ABSTRACT

Extracting ontological relationships (e.g., ISA and HASA) from free-text repositories (e.g., engineering documents and instruction manuals) can improve users' queries, as well as benefit applications built for these domains.

Current methods to extract ontologies from text usually miss many meaningful relationships because they either concentrate on single-word terms and short phrases or neglect syntactic relationships between concepts in sentences.

We propose a novel pattern-based algorithm to find ontological relationships between complex concepts by exploiting parsing information to extract multi-word concepts and nested concepts. Our procedure is iterative: we tailor the constrained sequential pattern mining framework to discover new patterns. Our experiments on three real data sets show that our algorithm consistently and significantly outperforms previous representative ontology extraction algorithms.

1. INTRODUCTION

An ontology is a specification of conceptualizations in a specific domain [3]. An ontology typically includes, at a minimum, concepts and hierarchical relationships among them. The two fundamental hierarchical relationships are ISA, which asserts a class-subclass relationship or class-instance relationship, which forms an ontology's taxonomic backbone, and HASA, which asserts a whole-part relationship.

Building an ontology from unstructured text on the web can bridge the gap between human-readable data and machine-readable knowledge in a specific area by improving navigation and information discovery. However, it is not enough to simply reuse general-purpose ontologies (e.g., WordNet). Such ontologies have limited coverage, particularly in specialized fields, where jargon and terminology can have very different meanings from their senses in a more general domain. For example, "foreign materials" in Architecture usually refers to substances other than piping; this is quite different from the meaning of foreign in "foreign nationals" which may be found in a legal document; and "agent" means different things in the medical and law domains.

Previous works on ontology discovery from unstructured text either explore hierarchical relationships among concepts from their distribution in a corpus [5, 27], or use lexico-syntactic patterns [22] (e.g., Hearst patterns [14]). Both approaches typically suffer from one of these drawbacks: (1) they generally extract *single-word terms* or *very short phrases* as concepts, or (2) they ignore the syntactic and parsing information available over text. These become problems when dealing with sentences involving complex nested noun phrases. Thus these *previous approaches fail to capture many meaningful and complex concepts in a domain, with the result that the discovered ontology is usually trivial or redundant given a general thesaurus.* Additionally, hierarchical approachess [5, 27] usually perform badly in the presence of sparse data, which is common in small corpuses.

We illustrate these ideas with an example drawn from a real architecture web dataset, which is one of the web datasets we use in our experiments: "avoid contact with finishes which may radiate noise, such as the concrete structure, future framing and drywall." In this fragment, "finishes which may radiate noise" is a meaningful concept; intuitively there is an ISA relationship from each of "concrete structure", "future framing" and "drywall" to the concept. However, both approaches above fail to capture this because: (i) this phrase is not frequent enough to be considered a concept by statistics-based algorithms; (ii) pattern-based algorithms identify noun phrases by matching part-of-speech (POS) tags applied by a tagger; hence a phrase with a clause modifier cannot be extracted and identified as a concept; (iii) "finishes which may radiate noise" is a nested noun phrase, so a naive pattern-based algorithm only looking for patterns like "such as" may end up retrieving "noise" as the parent concept in this ISA relationship, which is obviously wrong!

Extracting rich and complex concepts along with ontological relationships among them as accurately as possible is the main goal of this work. To this end, we developed an iterative pattern-based algorithm called LASER (for LArge ScalE Relation extraction system). The central paradigm used by

LASER is an iterative framework of starting with seed patterns that signal an occurrence of (ISA or HASA) relations, which are used to extract instances of relations in the corpus. They are in turn used to induce more patterns from the corpus and so forth. At every stage, the extracted patterns and instances are scored, reflecting their degree of reliability. When the score of extracted patterns drops below a threshold, the process terminates. This iterative framework was first pioneered by Brin [4] for extracting general relations from the web and was later adapted by researchers to various other contexts. Indeed, the ESPRESSO algorithm proposed in Pantel and Pennacchiotti [22] adapted this framework for ontology extraction. While a more detailed comparison with the ESPRESSO approach appears in Section 2, some fundamental differences between LASER and ESPRESSO include the following. (1) ESPRESSO assumes a set of seed instances for its iteration while LASER starts instead with a set of seed patterns well known to be reliable in the ontology extraction literature [14, 2, 12]. (2) We extract noun phrases by an analysis of the parse tree as opposed to relying on POS tag matching, a method most previous algorithms including ESPRESSO rely on. (3) Nested noun phrases pose a serious challenge for the correct extraction of relations. Unlike previous algorithms, we solve this problem effectively. (4) We solve pattern discovery using a novel approach based on constrained closed sequential pattern mining.

We make the following contributions:

- We extract the complex concepts having ISA/HASA relationships from a constituent parse tree of the text and effectively tackle the challenges involved in dealing with parsing information. We also propose a novel algorithm to resolve the challenge of determining the correct noun phrases linked by these relations when nested noun phrases are present. Compared to POS tag matching (used by most previous approaches), an analysis of parse trees more accurately identifies complex and nested phrases and relationships among concepts (Section 4.1).
- We start with a list of seed patterns from the the ontology extraction literature [14, 2, 12] and find ISA/HASA relationships and new patterns that signal occurrences of such relationships. Compared to seed instances used by ESPRESSO [22], seed patterns are more reliable and less variable across domains, while seed instances are often domain-specific, require validation, and can bias the quality of extractions.
- We tailor a sequential pattern mining framework to find frequent patterns that signal ISA/HASA relationships, and generalize the framework to allow the introduction of new patterns.
- We carry out a detailed experimental comparison between LASER and major representative algorithms from previous work on three real datasets. Our experiments show that LASER has a significantly better recall than previous algorithms while enjoying a comparable or better precision. In particular, our F_β-measure is significantly better than previous algorithms, for $\beta = 0.5, 1, 2$, demonstrating our approach's superiority. We show that LASER works very well and has stable performance on both small and large corpuses. We also show that LASER is much more scalable than previous approaches.

The rest of paper is structured as follows: Section 2 classifies and describes related work. Section 3 introduces our system architecture and components in detail. We discuss the LASER approach and the algorithms in Section 4 and discuss our experiments and lessons learned in Section 5. Finally, Section 6 concludes and discusses future work.

2. RELATED WORK

Ontologies fall into one of three types [3]: (i) a *formal ontology* is a set of logical expressions relating concepts by axioms and definitions; as such it can support inference and computation; (ii) a *prototype-based ontology* typically lacks labels for concepts, but relies on term clusters, which are regarded as prototypical instances of the underlying concept or category; (iii) a *terminological ontology* describes concepts by concept labels or synonyms, instead of using prototypical instances, and expresses hierarchical (e.g., subtype-supertype and whole-part) relationships between concepts.

This paper focuses on automatically building terminological ontologies from domain-specific text corpora. For example, given a set of engineering documents containing "furnace", "air conditioning unit", and "HVAC", we would try to find that the air conditioning unit ISA HVAC and the furnace ISA HVAC. In contrast, a prototype-based ontology might cluster "furnace" and "air conditioning unit" together, but it would not say that either ISA "HVAC", and a formal ontology would include additional semantic relationships. We survey previous related work below.

There are three main approaches to learning ontologies from unstructured text [3]: (i) *Data mining:* Clustering approaches cluster similar words based on the hypothesis that similar words tend to occur together in similar context [13]. Some approaches assign labels to the clusters, treating the labels as concepts and the terms in the cluster as its instances. More recently, association rule mining has been used for ontology learning; (ii) *Lexico-syntactic patterns:* Lexico-syntactic patterns such as Hearst Patterns [14] are used to extract relationships between terms. For example, the pattern "X such as Y" frequently implies Y ISA X; (iii) *Web as a data source*: to overcome data sparsity, some algorithms use the web as an additional data source, possibly in conjunction with other data sources like WordNet[1].

A large body of approaches build ontologies consisting of single words [5, 23] or short common compounds [27, 6]. Indeed, very few works allow for longer and complex terms to be concepts, generally because they have a very low frequency of occurrence compared to short ones. Drymonas et al. [10] make an attempt in this direction and allow more complicated noun phrases. They use a statistical measure to select meaningful concepts and an agglomerative clustering algorithm to build a prototype ontology. Recently, Navigli and Velardi [20] proposed Word-Class Lattices (WCLs) learned over a training set of definition sentences. These lattices are used for matching definitions and hypernyms of a given set of target terms and phrases. Using a terminology extractor (TermExtractor), WCLs can be utilized to iteratively extract a set of class-subclass relationships, resulting in a Hypernym Graph (WCL-HG). Pruning this hypernym graph using graph-based algorithms, Navigli et al. induce a domain taxonomy [21]. TermExtractor extracts long and short phrases as seed concepts and uses WCLs to extract complex hypernyms. One major drawbacks of WCLs is that their construction requires manual tagging of definition sentences by expert taggers. Moreover, they use POS tags and

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

therefore do not perform well on identifying boundaries of complex and nested phrases. In contrast, LASER relies on deep parsing information to get richer concepts that cannot be identified by WCLs. A detail experimental comparison of LASER with WCL-HG is presented in Section 5.

Data mining: Many clustering-based algorithms first produce prototype-based ontologies (See Caraballo [5] for an example). Others additionally assign labels to the clusters. For leaf clusters, he assigns a label using syntactic patterns, while for internal clusters he assigns the most dominant hypernym of the largest number of the node's descendants. Pantel and Ravichandran [23] first cluster words from a text corpus; each cluster forms a concept. They extract concept names by searching for syntactic patterns such as "concept apposition-of instance" and "concept such as instance". The word that co-occurs with the most dominant instances in a cluster most frequently is picked as the concept name. They create ISA relationships between the concept and all instances in the cluster. This helps with data sparsity since not all instances need to co-occur with the concept name for us to derive the ISA relationship between them. However, this kind of ISA relationship is necessarily confined to one level hierarchy, while our work produces a complex multi-level concept hierarchy.

Sanderson and Croft [27] use association rules to find ISA relationships (called subsumption in the paper) between terms. They use the intuition that for two terms, x and y, x ISA y holds if $P(x|y)$ is sufficiently large and $p(y|x) < 1$, where $P(x|y)$ is the probability that a document contains x given that it contains y. This approach is purely based on a statistical heuristic, which cannot produce a high-quality ontology alone. We use statistical heuristics as well as syntactic and semantic (parsing) information.

Lexico-syntactic patterns: Pure pattern-based methods (e.g., Pantel and Pennacchiotti [22]) usually iteratively interleave pattern discovery and instantiation until the reliability drops below a threshold. These methods tend to suffer from low recall.

ESPRESSO [22] is a system that given a small set of seed instances for a particular relation, learns lexical patterns, applies them to extract new instances, and then uses the web to filter and expand the instances. This procedure continues iteratively until it meets some stopping criteria such as reliability dropping below a threshold. We adapt their scoring mechanism in the iterative process, but instead of starting with seed instances, we bootstrap from seed patterns, which as discussed earlier are more reliable.

We extract instances from patterns substantially differently from ESPRESSO as well: we rely on deep parsing information to get richer concepts that cannot be identified by regular expression matching over data obtained from shallow parsing. Our pattern finding is also generalized to be less restrictive and more expressive than ESPRESSO. Additionally, we provide a novel formulation of pattern discovery as a constrained sequential pattern mining problem.

Hybrid approaches: As the name suggests, these approaches borrow ideas from one or more of the previous approaches. Guided Hierarchical Clustering (GHC) [6] is a representative algorithm in this class. It first calculates the similarity between a set of given input terms based on syntactic dependency features in the corpus including adjective modifiers, prepositional phrase modifiers and noun phrases in subject or object position. Then, using an agglomerative clustering algorithm, it picks the most similar pair of terms in the remaining list of pairs to be clustered, and uses WordNet, Hearst patterns in the corpus, and the WWW to position them in the growing ontology. If no relationship is found, the pair is clustered. Finally, they make sure the resulting ontology is a connected hierarchy. Unlike these, our approach finds relationships before concepts. We start from patterns that indicate relations, and then get concepts from there, thus not requiring terms as input.

Zavitsanos et al. [33] use topic modeling to extract concepts which are represented as distributions. TextToOnto [17] and Text2Onto [7] focus on conceptual relationships rather than hierarchical relationships. The concept hierarchy is used as a knowledge base to find more complex relations. For practical applications, algorithm that produce concrete ISA and HASA relations with labeled concepts are more useful than ones that produce latent topics and word clusters without labels.

Formal Ontologies: YAGO [29] automatically creates a formal ontology by extending WordNet using Wikipedia's info boxes. More recently, SOFIE [30] extends YAGO by extracting relationships from free text. Thus, they can process Wikipedia articles (not just info boxes) and indeed arbitrary web pages. Kylin/KOG [31, 32], DBPedia [1], and DBLife [9] are other examples of systems that have extracted very large ontologies containing millions of entities and relations.

Poon and Domingos [25] induce and populate a probabilistic ontology, using dependency-parsed text as input. The output ontology of this system mainly consists of verb classes in hierarchy with nouns as their argument class.

One of the distinguishing features of our work is that our algorithm can produce high quality ISA and HASA relations from domain specific text corpora. Our experiments show that our algorithm significantly outperforms previous ones both in quality and in running time. Indeed, as mentioned in [30], even for systems generating formal ontologies, algorithms that produce hierarchical relations on a large scale and with a high quality are essential in order to give the resulting ontology a clean structure.

3. ARCHITECTURE AND BACKGROUND

This section outlines the architecture of our system and develops the key notions and ideas used in our algorithms.

3.1 System Architecture

LASER (Figure 1) uses an iterative process. The dotted circle in Figure 1 highlights the main components.

LASER takes as input the preprocessed corpus consisting of a set of text documents, with each word tokenized. Additionally, it takes in a set of *seed patterns*, i.e., lexico-syntactic templates such as Hearst patterns [14] that imply ISA/HASA relationships. We define a *Subsumption Candidate Instance Pair* (SCIP), as a pair of noun phrases x, y such that they are involved in a class-subclass or class-instance (ISA) or a whole-part (HASA) relationship, and denote it SCIP(x, y). It states that either HASA(x, y) or ISA(x, y) holds.

The LASER system has the following modules:

0. Corpus Text Parsing and Indexing: We generate the parse tree of the corpus text with a parser such as the Stanford Parser[2], and build an inverted index on the corpus text as well as on the parse tree for efficient lookup. This

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

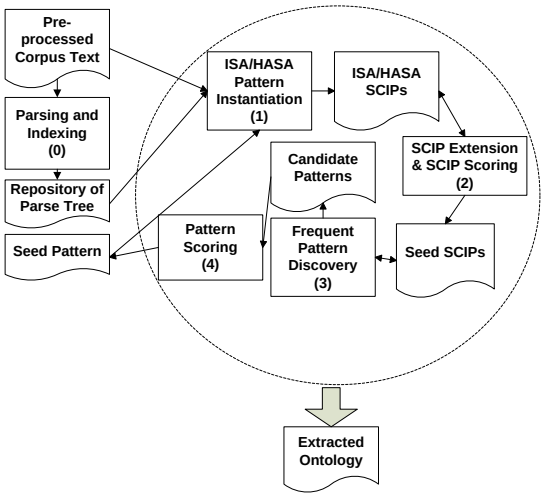

Figure 1: Ontology Extraction System Architecture

module is run once for a given corpus.

1. ISA/HASA **Pattern Instantiation**: This module finds the sentences containing seed pattern instances and extracts noun phrases involved in the ISA/HASA relationships.

2. SCIP Extension & SCIP Scoring: This module extends and ranks the instance pairs so that the highest ranking pairs are considered as seed instance pairs to aid new pattern discovery.

3. Frequent Pattern Discovery: This module takes the seed instance pairs from module 2 and uses them to find new patterns that imply ISA or HASA relationships.

4. Pattern Scoring: The candidate patterns from the previous module are scored to select seed patterns for the extraction of new instances in the next round.

This iterative process of finding instances and patterns continues until LASER cannot find new instances or the new patterns' score discovered drops below a threshold, which is tuned empirically. In our experiments we used the threshold of having the average score of patterns discovered in the current iteration being above 50% of the average score of patterns found in the previous iteration.

3.2 Background

We make use of the following notions in the LASER system.

DEFINITION 1. A *head word* is the word that determines the syntactic type of the noun phrase of which it is a member, while other words modify the head.[3] □

DEFINITION 2. A *noun phrase* is a syntactic unit of the sentence where information about the embedded noun is gathered [18]. Therefore the noun is the *head word* of the noun phrase, the central constituent which determines the syntactic characteristics of the phrase. A noun phrase usually consists of an optional determiner, zero or more adjective phrases, a noun head and other clause modifiers. □

E.g., in the example in the introduction, "finishes which may radiate noise" is a noun phrase with head word "finishes". A noun phrase is clearly a concept in the corpus and is a candidate concept in the domain-specific ontology that we seek to build.

DEFINITION 3. A *pattern* is a sentence fragment of the form: $NPList_1$ *Connector* $NPList_2$ where each $NPList_i$ is a list of one or more noun phrases and *Connector* is a sequence of words in a short phrase and/or the corresponding POS tags, which signals a relationship between the concepts represented by $NPList_1$ and $NPList_2$. □

For ISA and HASA relations, either $NPList_1$ or $NPList_2$ is the parent (i.e., the more general concept in an ISA relationship and the "whole" concept in a HASA relationship) while the other NPList is a child or a list of children (the more specific concept in an ISA relationship and the "part" concept in a HASA relationship), as determined by the specific connector. After identifying $NPList_1$ and $NPList_2$ of a pattern in the text corpus, we generate candidate instance pairs for this pattern consisting of one noun phrase from $NPList_1$ and another from $NPList_2$.

As an example, in the clause "plumbing equipment such as steel storage tanks, pressure reducing stations and ductile iron pipe", the string "such as" acts as a connector; "plumbing equipment", "steel storage tanks", "pressure reducing stations" and "ductile iron pipe" are noun phrases. In this example, $NPList_1$ is a single noun phrase and $NPList_2$ is a list of noun phrases. The connector "such as" indicates each noun phrase in $NPList_2$ is an instance or subclass of the noun phrase $NPList_1$, that is, one can infer the relations ISA(plumbing equipment, steel storage tanks), ISA(plumbing equipment, pressure reducing stations) and ISA(plumbing equipment, ductile iron pipe).[4] Recall, as defined in Section 3.1, we use the term SCIP pair, denoted $SCIPs(x, y)$, to indicate a pair of noun phrases x, y that are related by an ISA or a HASA relationship.

This kind of pattern was first proposed by Hearst [14] and extended by many subsequent papers.

We use the seven patterns below, called *seed patterns*, as the initial ISA patterns in the first round of our iterative taxonomic relation extraction procedure:

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

The above four are from Hearst [14] while the following three are from Cimiano and Staab [6], which extends [14]:

5. NP_1 is NP_0
6. NP_1, another NP_0
7. NP_0 like NP_1

As a follow up to Hearst's work [14], [2, 12] proposed similar lexico-syntactic patterns implying part-whole relationships (i.e., HASA relationships). We used five patterns from their work as our initial list of HASA patterns:

1. NP_0 (consists|consist|made) of NP_1
2. NP_1 (members|a member|a part) of NP_0
3. NP_0 have|has NP_1
4. NP_1 inside NP_0
5. parts of NP_0 include NP_1

[3] http://en.wikipedia.org/wiki/Head_linguistics

[4] ISA(x, y) indicates y is an instance (or subclass of) x.

4. ONTOLOGY EXTRACTION

In this section, we provide a detailed description of modules 1–4 in the architecture schematic (Figure 1) and provide the key algorithms used in our LASER system. Specifically, we describe the algorithms for correct handling of nested noun phrases and for pattern discovery using constrained sequential item-set mining.

4.1 ISA/HASA Pattern Instantiation

This module takes a list of known patterns suggesting ISA or HASA relationships and applies the patterns to the input corpus to find sentences matching the patterns.

4.1.1 Extracting Noun Phrases

Previous works [6, 22] usually find pattern instances by matching each POS tagged sentence with regular expressions. For example, the regular expression $(DT\backslash t(\backslash w+))?(JJ\backslash t(\backslash w+))?((NN(S?) \setminus t([a-z]+) \setminus s?)+)$ determines a non-recursive noun phrase, in which zero or one determiners (DT) followed by zero or one adjectives (JJ) plus one or more singular or plural nouns ($NN(S?)$) is a noun phrase. Thus, "a stringent requirement", which is tagged as "NN a JJ stringent NN requirement", can be recognized as a noun phrase because it matches the regular expression. Such a strategy has the following limitations:

1. The simple POS tag rules may identify the wrong noun phrase because the context is not considered. For example, in the sentence "Adding flooring finishes such as carpet can significantly change the Apparent-IIC", "flooring finishes" is the correct parent noun phrase of "carpet". However, according to the POS tags "Adding/VBG flooring/NN finishes/NNS" and the POS based rules, "adding flooring finishes" (a verb phrase) is incorrectly identified as a noun phrase, leading to the incorrect inference ISA(carpet, adding flooring finishes). The inaccuracy of POS tagging will lead to incorrect identification of noun phrases, which causes false positive and false negative pattern instances.
2. Strict application of pattern matching may fail to capture some patterns that contain the proposed patterns. For example, suppose we try to extract SCIPs with the seed pattern "... including ...". Then even though the connector "... including but not limited to ..." contains the seed pattern and is meaningful, we cannot extract SCIPs from it because the pattern matching requires that a noun phrase immediately follow the connector "including".
3. Simple POS tag rules cannot identify some noun phrases that have complex structures using modifiers. For example, "coatings that may be detrimental" is a noun phrase occurring in one of our real data sets. It has an attributive clause as a modifier, which cannot be correctly identified by simple regular expression rules.

To overcome these limitations, we perform pattern matching by first matching sentences containing lexical connectors, and then extracting the corresponding noun phrases from the text segments either surrounding those connectors or in between them, by analyzing the constituent parse tree structure for the sentences. The idea is that a well-trained parser like the Stanford Parser can be more effective at determining noun phrases than simply matching regular expressions over POS tags. For example, for the pattern $NPList_1$ *such as* $NPList_2$, we extract noun phrases from the left of and the right of "such as" respectively. We achieve this by identifying appropriate noun phrases in the parse tree of the matched sentence, knowing the position of the connectors in that tree. This novel approach is in contrast to previous approaches which use hand-crafted rules to match a whole sentence.

However, LASER adopts a slightly different strategy for the first iteration of pattern instantiation and the later iterations: it extracts noun phrases with the matched words in the pattern as clear boundary in first run but allows a sliding window around the words for patterns in later run. For example, this pattern
"NP_0 such as $NP_1, NP_2, ..., NP_{n-1}$ (and|or) NP_n"
is used in the first iteration. LASER will look for the noun phrase exactly preceding "such as" in the parse tree as the parent noun phrase. Similarly, the child noun phrases are extracted with the matched position of "such as" and "and|or" as clear boundaries.

On the other hand, "$, /, including/VBG$" is a pattern LASER found in later iterations, so LASER uses a sliding window of word size w that its left end starts with "$, /,$" and moves towards left (one word at a time), until the right end of the window reaches "$including/VBG$". in this way we try to find a noun phrase with the last word that is on the left of the pattern and resides in the window. Similarly we moves the window towards right until the left end of the window reaches "$, /,$", starting with the right end of the window placed at "$including/VBG$", and finds the corresponding child noun phrases. Simply saying, we allow the noun phrases to occur around the pattern with a small gap, which is $w - wordsizeofpattern$. The sliding window just simulates the procedure we try to match from close to further.

The reason we adopt different strategy here is: the seed patterns used in first iteration are defined by linguistic experts, that the words in a pattern are meaningful, so that we use exact match to get more accurate result. However the generic patterns in later iterations are found by frequent substrings, which can be a part of a meaning connector that is not frequent enough. Take the example of ", including" and ", including but not limited to", the relaxed match gives some longer patterns a second chance.

4.1.2 Nested Noun Phrase Challenge

One challenge in ontology extraction is that noun phrases may be nested in another noun phrase. In this case it is difficult to identify the appropriate noun phrases in the extracted relationship. Below, we give two examples that illustrate this challenge: in Example 1 the shorter, nested noun phrase is the correct one. In Example 2, the longer, outer noun phrase is the correct one. The examples are taken from one of the real data sets described in Section 5.1.

EXAMPLE 1. Consider the 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 ." Here, "Shading devices" inside "provision of shading devices" is a nested noun phrase.

"Shading devices" is the correct parent concept of the ISA relationship not "provision of shading devices." That is, in making inferences about ISA relationship we should be using "shading devices", not "provision of shading devices." □

EXAMPLE 2. In "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 ," the noun phrase "the authorities having jurisdiction" contains the nested noun phrase "jurisdiction".

In this case the longer phrase is the right choice for use in the relationship inference. Using more complex noun phrase adds an extra dimension to the problem: not only do we aim to avoid outright incorrect choices, but we strive to pick the best among the correct ones. □

These two examples are in sharp contrast and clearly illustrate the challenge in determining the appropriate noun phrase for relationship inference; it is not always better to use the longer noun phrase nor always the shorter one. To solve this challenge, we employ a linguistically based heuristic approach that uses hints from an external source, e.g., a general thesaurus like WordNet. A useful cue about the type of a noun phrase can be obtained from its head word.

For example, the head word for "shading devices" is "devices" and "provision of shading devices" has the head word "provision". When the sentence contains an ISA or HASA pattern but the potential parent noun phrase is nested, such as in Example 1 and in Example 2, we can identify the head words of child noun phrases and the potential parent noun phrases (generated by extracting all noun phrases from the nested noun phrase recursively), and try to find relationships among these head words in WordNet.

In order to measure the relatedness between words, we use the semantic similarity defined in [24], which makes use of corpus statistics and the hierarchical structure in WordNet.

The WordNet::Similarity module[5] implements different variations of semantic similarity.

In our work, we use three of them and take the average:

- "Path" is the inverse of the shortest path length between two concepts in WordNet.
- The other two measures are based on *information content*, a corpus-based measure in information theory that is proposed by Resnik [26] to represent the specificity of a concept (more specific the concept is, larger this value will be). One way to estimate this value is by corpus statistics and the WordNet::Similarity module has pre-computed it for concepts in WordNet using standard corpus.
- The final measure is "JCN" is the semantic similarity described by Jiang and Conrath [16], which subtracts the information content of the LCA(Lowest Common Ancestor) of the two concepts, from the sum of the information content of these two, then takes the inverse of the substraction result (convert the distance to similarity).

Algorithm 1 extracts the best possible choice for a parent concept given a nested noun phrase (for parent) and a list of noun phrases (for child).

Lines 4 to 12 in Algorithm 1 calculate the sum of the similarity between each candidate parent's head word and head words of all children. We remember the candidate parent that has the maximum similarity sum, $MaxSimSum$.

For example, "Provisions of shading devices, such as overhangs or vertical fins", has two candidate head words: "provision" and "devices" for the candidate parents ("provision of shading devices" and "shading devices", respectively). Hence, we sum the semantic similarity between "provision" and "overhangs" and between "provision" and "fins". Similarly we sum the semantic similarity between "devices" and "overhangs", "devices" and "fins". In this case, the similarity sum is larger for "devices" than for "provision", which suggests that "shading devices" is a better choice of parent concept.

If two candidates have the same sum, we will choose the shortest one (Line 9–11), because the head word of a parent phrase tends to be closer to the child phrases that specify this parent. When the maximum similarity sum is zero, meaning head words are not found in WordNet (which is possible when we are dealing with a domain-specific corpus), we will try to find the shortest noun phrase, with head word in plural form if it exists, as a default behavior (Line 13–15).

[5]A module that implements a variety of semantic similarity and relatedness measures based on information found in the lexical database WordNet.

Algorithm 1 Parent_NP_Resolution_in_Nested_NP

Input: A nested noun phrase ($NestedNP$) containing the potential parent noun phrase; A set of child noun phrases ($ChildList$).

Output: The appropriate parent noun phrase ($ParentNP$), which is a hypernym of the noun phrases in $ChildList$.

1: $ParentList \leftarrow$ Recursively extract a list of noun phrases containing the last word in $NestedNP$ from the parse tree
2: $MaxSimSum = -1$
3: $CurrentCandidate = null$
4: **for all** $Candidate \in ParentList$ **do**
5: $SimSum = \sum_{ChildNP \in ChildList} Similarity(headof(Candidate), headof(ChildNP))$
6: **if** $SimSum > MaxSimSum$ **then**
7: $CurrentCandidate = Candidate$
8: $MaxSimSum = SimSum$
9: **else if** $SimSum == MaxSimSum$ and $length(Candidate) < length(CurrentCandidate)$ **then**
10: $CurrentCandidate = Candidate$
11: **end if**
12: **end for**
13: **if** $MaxSimSum == 0$ **then**
14: Return $ParentNP \leftarrow$ shortest $Candidate$ in $ParentList$, breaking ties in favor of a candidate with a plural head word if any, and then arbitrarily.
15: **end if**
16: Return $ParentNP \leftarrow CurrentCandidate$

4.2 SCIP Extension

We can extend the set of SCIP pairs derived by generating several more SCIPs that exploit the inherent ISA relationship between a complex phrase and its head word and the transitivity of ISA relationship. E.g., consider the SCIP ISA(plumbing equipment, ductile iron pipe). We can extend this by generating the SCIP ISA(equipment, ductile iron pipe). Many existing algorithms make the assumption, that if ISA(NP_1, NP_2), then necessarily ISA($head(NP_1)$, $head(NP_2)$). This is an assumption, not necessarily a valid inference. Notice that ISA($head(N_1), head(NP_2)$) does *not* follow from transitivity. In the above example, it turns out ISA(equipment, pipe) *happens* to be valid.

Our observations on real data sets indicate that this assumption results in many erroneous relationships or trivial relationships that can be found in a general ontology. This is because in many cases, the sense of the head word cannot be disambiguated without modifiers. According to our preliminary results, 49% of head word pairs derived do not form valid ISA pairs. For example, following this assumption on ISA(points of penetration of the vapor barrier jacket, raw

edges) yields ISA(points, edges), which is meaningless!

In summary, we extend every extracted pair ISA(NP_1, NP_2) from a SCIP by generating the additional pair ISA(*head* (NP_1), NP_2). Then we calculate reliability scores for all extracted and extended pairs based on the scoring mechanism described in Section 4.4. Finally, we filter those pairs with scores smaller than average and pick the top ones as seed SCIPs for discovering new patterns in the next iteration.

4.3 Frequent Pattern Discovery

Using the seed SCIPs (ISA/HASA relationships) produced by module (2) from Figure 1, we want to find new patterns that imply these relationships.

We adopt a *Frequent-Substring-based Pattern Extraction* approach to achieve this new pattern discovery. The idea is to find substrings that frequently occur in between the parent concept and the child concept of a SCIP in the corpus. Using seed instances in the form of SCIP(NP_1, NP_2) as input, and we find co-occurrences of NP_1 and NP_2 in the corpus where the text in between NP_1 and NP_2 is shorter than a pre-defined limit. After collecting text sequences for each SCIP, we find frequent substrings from them.

ESPRESSO [22] finds frequent substrings that contain both concepts of a SCIP by building a suffix tree for all tagged sentences containing both concepts of instances. This suffix tree keeps a record of all substrings of these sentences. The frequent substrings are considered to be candidates for new patterns. For example, given a tagged sentence: "Sensory/JJ aspect/NN such/JJ as/IN air/NN quality/NN can/MD easily/RB be/VB compromised/VBN ./.", and it is given ISA(sensory aspect, air quality). Espresso replaces the actual parent and child concept by "X" and "Y", which gives the generalized tagged sentence : "X such/JJ as/IN Y can/MD easily/RB be/VB compromised/VBN ./." A frequent substring of tagged sentences like above will look like "X such/JJ as/IN Y", in which ISA(X, Y).

However, this kind of pattern is not general enough because ESPRESSO requires *exact* matches of both words and the corresponding POS tags. Due to data sparsity, a problem that is especially severe for a small-scale and domain-specific corpus, frequent patterns are hard to find and the resulting patterns will have limited power in picking out instances in later iterations of instance extraction. ESPRESSO tries to generalize this kind of pattern by replacing terms (their counterpart of our "noun phrase"), that are identified by regular expression matching over tagged sentences, with a uniform symbol. As we show in Section 5, this kind of pattern still suffers from low recall.

In contrast, instead of just finding frequent substrings in all sentences (which can be really long) containing a SCIP, we find frequent substrings from the text in between the two concepts of a SCIP. Similarly, we do not require exact matches of both words and their corresponding POS tags.

Consider the input candidate connector "and others;" "or others" may also be a valid connector but does not occur frequently enough. Instead, we may look for a pattern like "* others", where we require that both terms have the same part of speech tags as the original pattern. Similarly, we could also allow for generalizing patterns based on POS tags.

To solve the above problem, we tailor the Generalized Sequential Patterns (GSP) algorithm in [28]. In GSP, given a database of lists (sequences) of transactions (item-sets) ordered by transaction time, the problem is to determine frequent sequential patterns that have a minimum user-specified support, i.e., number of sequences containing the pattern.

In our setting, each word in the corpus corresponds to an item-set consisting of two items — the word and the corresponding POS tag (e.g., flooring/NN). When looking for frequent subsequences, we sometimes generalize a POS tag (e.g., flooring/*) or the corresponding word (e.g., */NN) but we always require the item-set to contain at least one item (other than "*"). We say a subsequence of a sequence is *continuous* provided there are no gaps. E.g., in the candidate connector "including/VBG but/CC not/RB limited/VBN to/TO," "including/VBG */CC" is a continuous subsequence, but "including/VBG not/RB" is not. Thus, given a SCIP pattern (NP_1, NP_2), we find all candidate connectors for new patterns by finding all occurrences of NP_1 ⟨candidate connector⟩ NP_2 in the corpus where ⟨candidate connector⟩ may be different across occurrences. We then mine all frequent subsequences from the set of ⟨candidate connector⟩'s satisfying the constraints: (i) the mined subsequences must be continuous and (ii) each item-set in it must contain at least one item.

The frequent subsequences that are output are treated as candidate lexico-syntactic patterns. We make use of a scoring mechanism, described in the next subsection, for choosing the top patterns as the seed patterns for the next iteration of extraction of instance pairs. The algorithm outlined above is given in Algorithm 2.

Algorithm 2 Frequent_Pattern_Discovery

Input: A set of instances (*SeedInstances*) of the form ($ParentNP, ChildNP$) where ISA or HASA relationship may hold between $ParentNP$ and $ChildNP$
Input: A limit ($WindowSize$) on the number of words in the text window considered in each instance's occurrence
Output: A set of generic patterns (*CandidatePatterns*).

$Seqs = \varnothing$
$ReversedSeqs = \varnothing$
for all $instance \in SeedInstances$ **do**
  if $instance = (ParentNP, ChildNP)$ **then**
    add to $Seqs$ tagged substrings w/ length $< WindowSize$ that occur between $ParentNP$ and $ChildNP$
  end if
  if $instance = (ChildNP, ParentNP)$ **then**
    add to $ReversedSeqs$ tagged substrings with length $< WindowSize$ that occur between $ChildNP$ and $ParentNP$
  end if
end for
$Patterns = \varnothing$
$ReversedPatterns = \varnothing$
$Patterns \leftarrow constrainedBIDEplus(Seqs)$
$ReversedPatterns \leftarrow constrainedBIDEplus(ReversedSeqs)$
$CandidatePatterns \leftarrow Patterns \cup ReversedPatterns$

4.4 Scoring of Patterns and SCIPs

We need a scoring mechanism to select seed SCIPs and seed patterns to identify new patterns and new concept pairs respectively, and decide the stopping criteria for the iterative process. It is prohibitively expensive to evaluate the actual precision of patterns and SCIPs at run-time.In order to estimate the confidence of a pattern or a SCIP, we need to capture the association between a SCIP and any pattern contributing to its extraction in the pattern instantiation step. Similarly, we need to capture the association between a pattern and any SCIP contributing to its dis-

covery in the frequent pattern discovery step. We follow the Point-wise Mutual Information[8] (PMI) framework for scoring patterns and instances. PMI measures the association strength between two events x and y, and is defined as: $pmi(x,y) = \log \frac{P(x,y)}{P(x)P(y)}$

Using the PMI framework, we use the following formulation:

$$pmi(i,p) = \log \frac{\frac{|NP_1,p,NP_2|}{\sum_{\hat{p}\in P',\hat{i}\in I'} |NP_{1_{\hat{i}}},\hat{p},NP_{2_{\hat{i}}}|}}{\frac{|NP_1,*,NP_2|}{\sum_{\hat{i}\in I'} |NP_{1_{\hat{i}}},*,NP_{2_{\hat{i}}}|} \frac{|*,p,*|}{\sum_{\hat{p} in P'} |*,\hat{p},*|}} \quad (1)$$

in which P' is the set of patterns in the current iteration and I' is the set of instances used to find new patterns. In the above equation, we divide the frequency value in the numerator and the denominator with corresponding sum values, namely the sum of co-occurrence frequency for all pairs of instance and pattern, the sum of frequencies of all instances, and the sum of frequencies of all patterns, respectively. Here, $\hat{i}$ ranges over instances, i.e., $\hat{i} = (NP_{1_{\hat{i}}}, NP_{2_{\hat{i}}})$.

In the first iteration of pattern instantiation, we estimate the precision of the initial set of pre-defined patterns by manual validation on a sampled output, and use those estimates as initial scores. The algorithm runs until no more new SCIPs can be found or the average score of patterns produced in the current iteration is smaller than 50% of the average score of patterns from the previous iteration.

5. EXPERIMENTS AND EVALUATION

5.1 Dataset Description and Preprocessing

We evaluated our results on the following three datasets:

AEC: The Architecture, Engineering, and Construction dataset consists of the text data used by the construction firm in the process of constructing the Centre for Interactive Research on Sustainability (CIRS) building at the University of British Columbia. It is a web archive containing scheduling data, 3D design data, meeting notes, and reports used in the construction of a medium-sized building. We extracted the text and applied basic cleaning. The resulting small corpus contains 18,805 sentences and 312,936 words. This fairly small dataset shows challenges for ontology extraction when data is sparse.

LP: LP[6] consists of text from *http://www.lonelyplanet.com.* This tourism-domain small-scale dataset has 18,950 sentences and 453,299 words; LP is also used by [6].

MED: OHSUMED (or "MED" for short) consists of 348,566 medical references from MEDLINE [7] [15]. We use a large subset of this collection consisting of 1,221,462 sentences and 32,524,017 words. MED is a standard corpus in information retrieval.

We did the following preprocessing steps on all datasets:

- We cleaned the data (e.g., we removed running footers).
- We broke the text into sentences with the LingPipe toolkit[8].
- We used the Stanford NLP tools to tokenize, tag, and parse the data.
- We built an inverted index with Lucene[9].

[6]http://olc.ijs.si/lpReadme.html
[7]http://www.ncbi.nlm.nih.gov/pubmed/
[8]http://alias-i.com/lingpipe/
[9]http://lucene.apache.org/

5.2 Competing Algorithms and Parameters

We compare LASER with three other algorithms:

ESPRESSO: Espresso [22], discussed in detail throughout the paper, is our closest competitor.

GHC: GHC [6] (Section 2) did not describe how to generate the terms to build the taxonomy on. To overcome this, we used the C/NC-value method [11] proposed by OntoGain [10] to create multi-word candidate terms for GHC. We slightly modified C/NC to produce single-word terms in addition to multi-word terms since GHC performs poorly when only multi-word terms are input.

WCL-HG: WCL-HG [21] (Section 2) uses Word class lattices learned over the manually tagged Wikipedia sentences [20]. We did not have access to the same or a comparable term extractor as the one used in [21]. Thus, we used ISA subclass terms and phrases extracted by other three approaches as seeds to this algorithm.

Both LASER and ESPRESSO [22] are iterative. LASER starts with the seed patterns in Section 3, and ESPRESSO is given 50 seed SCIPs for AEC and LP and 100 seed SCIPs for MED. Since there exist no validated SCIPs for both corpuses, we randomly selected a set of SCIPs that have been labeled as valid when we evaluated the experiment result for LASER.

During the ISA/HASA Pattern Instantiation iteration, both algorithms pick the top k extracted (extended) SCIPs as seed SCIPs. For the small datasets like AEC and LP, we use all extracted (extended) SCIPs as seed SCIPs ; and k is set to 1500 on MED.

In the Frequent Pattern Discovery iteration, LASER chooses the top m patterns. ESPRESSO produces the top m patterns during the first run and generates patterns that increase in size by one pattern per round, e.g., the second round will find $m+1$ patterns. LASER sets m to 10 when finding ISA relationships on AEC and 5 for all other cases, and ESPRESSO has $m = 5$.

The only parameter for GHC is the number of input terms, n. The n most important terms from corpus chosen by our implementation of the C/NC-value method[11] are given as input, and GHC tries to find ISA relationships among them and build a hierarchy. We set $n = 1000$ for AEC, $n = 2000$ for LP and $n = 5000$ for MED because we want to keep the output size of different algorithms comparable.

WCL-HG iteratively finds a set of hypernyms for the set of seed target concepts and replaces them with the newly found hypernyms. We use $K = 5$ iterations in our experiments. We use 0.38 as the threshold to filter sentences with higher domain weights, where domain weight is a measure indicating how much a sentence has in common with the domain terms (extracted using a terminology extractor) — this is the same threshold reported in [21]. The set of input target values are 1102, 1343 and 2662 for LP, AEC and MED, respectively.

GHC, ESPRESSO and WCL-HG originally used Google as an external source of expanding ISA/HASA relationships. However, Google has recently restricted the use of its search service API. Thus, we used online version of Merriam-Webster dictionary[10] for WCL-HG definition sentences and implemented the web extension part in ESPRESSO with Microsoft Bing search API[11]. Since GHC reports that it's chief

[10]http://www.merriam-webster.com/
[11]http://www.bing.com/toolbox/bingdeveloper/

improvement by using web expansion was a minor precision improvement (2-3%), we did not implement its web expansion with Bing.

We applied the four algorithms to the datasets in Section 5.1. LASER and ESPRESSO find ISA/HASA relationships while GHC and WCL-HG are only able to produce ISA relationships.

In addition to measuring the algorithms' *precision* (i.e., what fraction of the results that are returned are correct), we would like to measure recall (i.e., what fraction of the correct results are returned). However, given that it is infeasible to fully find all ontological relationships in a large text repository, we measured *relative recall* — the number of valid relationships found by the algorithm divided by the total number of valid relationships found by all algorithms [19]. This allows us to also define *relative F-score* by replacing *recall* with *relative recall*. Thus *relative F-score*$_\beta = (1+\beta^2) * \frac{precision * relative\ recall}{\beta^2 * precision + relative\ recall}$, in which recall is weighted β times as important as precision. Therefore, F_1 weights precision and recall equally, $F_{0.5}$ weights precision as 2 times more important than recall, and F_2 weights recall as 2 times more important. We use these F-scores here because precision and recall may be weighted differently in different applications.

5.3 Comparison of ISA Results

Using the stopping criteria in Section 4.4, LASER ran two ISA Pattern Instantiation iterations on all datasets. ESPRESSO only ran one ISA Pattern Instantiation iteration before it reached its stopping criteria.

Table 1 shows the total number of all output ISA relationships for each algorithm and the corresponding precision. We manually validated all relationships produced for AEC. For the other two corpora, we validated random 100 results if there were more than 1,000 relationships, otherwise did complete validation. LASER1 and LASER2 represent the relationships directly extracted from patterns during iteration 1 and 2; LASER is the total result from all iterations. HW denotes the results containing extended relationships found by the SCIP Extension step (Section 4.2). HW+W represents the result with both head word extension and web extension. Finally, DEF indicates expanded results using web definitions (Merriam-Webster dictionary in this case). In this scenario, for every definition ⟨DEF⟩ found for the input phrase ⟨TARGET⟩, we add the following to the set of candidate sentences: "⟨TARGET⟩ is a ⟨DEF⟩".

ESPRESSO achieves the best precision on the two smaller datasets and LASER achieves the best precision on MED. Head word extension increases the number of relationships found by both LASER and ESPRESSO, with precision remaining about the same or decreasing a little bit because of errors in finding head words. ESPRESSO's web expansion produces many additional relationships, but it markedly degrades precision.

There are two numbers in each precision column of ESPRESSO HW+W and WCL-HG+DEF; the first measures precision on relationships found in the domain. The second measures precision if the relationship is valid in *any* domain. For example, ISA(accessories, necklace) is extracted by ESPRESSO on the AEC dataset. This is not valid in the architecture domain because necklace is not a concept in this domain — in this domain, accessories stands for construction or mechanical equipment.

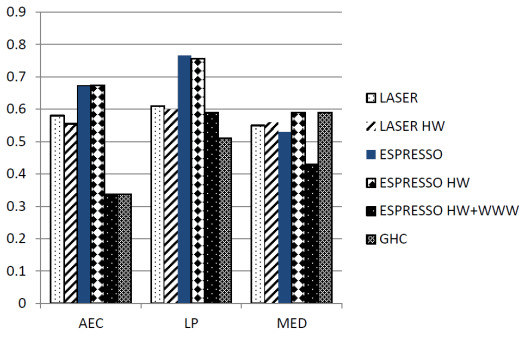

Figure 2: Precision Result for ISA

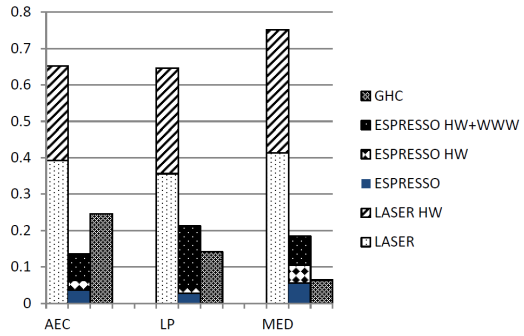

Figure 3: Relative Recall Result for ISA

GHC relies heavily on ISA relationships between a term and its head word, e.g., ISA(system, heat recovery system), which are fairly trivial. Neither LASER, nor ESPRESSO output these relationships. The input terms extracted for MED contain a higher percentage of multi-word terms (32%) than those of AEC (23%), so GHC performs much better on the MED corpus: more "trivial" relationships can be found.

The relatively poor performance of WCL-HG compared to the results in the initial publications [20] may be due to the following reasons. (1) Lacking access to a good set of input domain terms, (2) Patterns learned by WCLs over Wikipedia definition sentences are not effective over complex and technical corpuses such as AEC. (3) As noted in [20], in many cases WCLs are only able to match a substring of the complete match phrase. E.g., over AEC, they return ISA(furring, application) as a match, which is too general and is a substring of the correct match ISA(furring, application of thin wood).

Table 2 gives the relative recall and different F-scores for algorithms on all corpuses. Testing the validity of all relationships from the two larger datasets is impractical, so for those, we estimate *relative recall* by: *relative recall* $\approx \frac{precision * |SCIPs|}{\sum precision_x * |SCIPs|_x}$ — the number of valid relationships produced by an algorithm is estimated by the product of sample precision and the number of all generated SCIPs. Summing the estimated valid relationships for all competing algorithms, yields the number of all valid relationships from all systems' output, which is an overestimate of the real value. Therefore, the estimated relative recall is an under estimate but still reflects the difference between systems.

LASER HW outperforms the other two algorithms and

Table 1: Precision and Total Number of ISA Results

System	AEC Precision	AEC Total	LP Precision	LP Total	MED Precision	MED Total
LASER1	0.593	617	0.63	2198	0.6	19338
LASER1 HW	0.564	1070	0.5	3995	**0.61**	34390
LASER2	0.453	64	0.644	104	0.37	5323
LASER2 HW	0.459	111	0.642	179	0.42	9674
LASER	0.58	681	0.61	2302	0.55	24661
LASER HW	0.555	1181	0.6	**4174**	0.56	**44064**
ESPRESSO	0.673	55	**0.766**	141	0.53	3472
ESPRESSO HW	**0.674**	95	0.755	229	0.59	5814
ESPRESSO HW+W	0.337/0.562	406	0.59/0.68	1396	0.43/0.5	14077
WCL-HG	0.189	53	0.3	77	0.37	6210
WCL-HG+DEF	0.139/0.39	**2353**	0.3/0.44	455	0.23/0.38	7584
GHC	0.337	734	0.51	1074	0.59	3557

Table 2: Relative Recall and F-score of ISA

	AEC				LP				MED			
System	RR	F_1	$F_{0.5}$	F_2	RR	F_1	$F_{0.5}$	F_2	RR	F_1	$F_{0.5}$	F_2
LASER	0.30	0.40	0.49	0.33	0.35	0.44	0.53	0.38	0.39	0.46	0.51	0.41
LASER HW	**0.49**	**0.52**	**0.54**	**0.50**	**0.62**	**0.61**	**0.60**	**0.62**	**0.71**	**0.63**	**0.58**	**0.67**
ESPRESSO	0.03	0.06	0.13	0.04	0.03	0.05	0.12	0.04	0.05	0.09	0.18	0.06
ESPRESSO HW	0.05	0.09	0.19	0.06	0.04	0.08	0.17	0.05	0.10	0.18	0.31	0.13
ESPRESSO HW+W	0.11	0.17	0.24	0.13	0.21	0.31	0.44	0.24	0.18	0.26	0.34	0.21
WCL-HG	0.01	0.02	0.04	0.01	0.01	0.02	0.04	0.01	0.10	0.16	0.24	0.12
WCL-HG+DEF	0.24	0.17	0.15	0.21	0.03	0.05	0.11	0.04	0.07	0.11	0.16	0.08
GHC	0.19	0.24	0.29	0.21	0.14	0.22	0.34	0.17	0.05	0.09	0.19	0.06

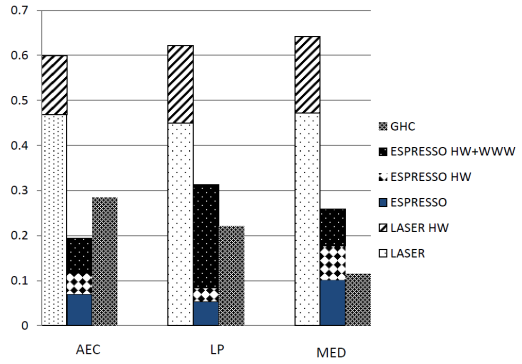

Figure 4: Relative F-score Result for ISA

corresponding extensions in terms of relative recall and F-scores, thanks to the large output and stable precision. In contrast, ESPRESSO suffers from low relative recall. This behavior is consistent on both small and large datasets, which reflects a problem of starting an iterative algorithm from seed SCIPs. Although Espresso's set of SCIPs are valid, the distribution of these seeds in the corpus is unknown beforehand, leading to possibly re-discovering the same pattern repeatedly and hence a consistently low recall. GHC has better relative recall and F-score than ESPRESSO on AEC even when its precision is low on AEC. On the large corpus, GHC has worse relative recall mainly because the agglomerative clustering algorithm does not scale well. As we show later, even running GHC with 5000 terms took more than two days. WCL-HG has its largest relative recall on AEC dataset. The reason is that the algorithm is able to find a large number of web definitions for the input seed target phrases. As a result, it finds almost all its matches from web definitions. However, most of these matches are either not complete or are not relevant in the given domain, which results in a low precision.

Figure 3 shows the relative recall of ISA relationships. Since the variation in precision across different parts of the algorithms (e.g., the difference between ESPRESSO and ESPRESSO HW) is relatively low, the F-scores (Figure 4) are very similar to the relative recall graph. We can conclude that Head Word Extension and Web Extension both improve F-scores, and LASER HW dominates consistently.

In Figure 5 we plot precision, relative recall, relative F1-Score as a function of X, in which top X% stands for the top X% SCIPs having highest scores on AEC. Extractions are not ranked for WCL-HG and GHC, therefore they do not appear in this figure. The precision of LASER remains relatively steady even when we dig to the bottom of the scores, i.e., as X increases. Additionally, both LASER's relative recall and F-score increase rapidly as X increases, showing LASER's dominance over the competition.

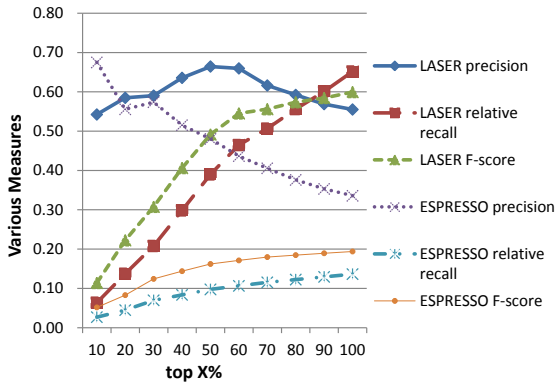

Figure 5: Comparing the Top X% ISA SCIPs

Table 3: Precision and Total Number of HASA Results

System	AEC Precision	AEC Total	LP Precision	LP Total	MED Precision	MED Total
LASER1	0.415	82	**0.626**	673	**0.42**	4011
LASER2	0	0	0.429	7	0.25	417
LASER	**0.415**	**82**	0.624	**680**	0.39	**4428**
ESPRESSO	0.25	4	0.588	51	0.35	428
ESPRESSO HW	0.11	9	0.521	71	0.31	649
ESPRESSO HW+W	0.1	10	0.454/0.471	121	0.34/0.39	1072

Table 4: Relative Recall and F-score of HASA

	AEC				LP				MED			
System	RR	F_1	$F_{0.5}$	F_2	RR	F_1	$F_{0.5}$	F_2	RR	F_1	$F_{0.5}$	F_2
LASER	**0.971**	**0.581**	**0.469**	**0.766**	**0.885**	**0.732**	**0.663**	**0.817**	**0.826**	**0.530**	**0.436**	**0.675**
ESPRESSO	0.029	0.052	0.099	0.035	0.063	0.114	0.221	0.077	0.072	0.119	0.197	0.085
ESPRESSO HW	0.029	0.046	0.071	0.034	0.077	0.134	0.242	0.093	0.096	0.147	0.215	0.112
ESPRESSO HW+W	0.029	0.045	0.067	0.034	0.115	0.184	0.286	0.135	0.174	0.230	0.286	0.193

5.4 Comparison of HASA Results

LASER ran one HASA Pattern Instantiation iteration on AEC and two iterations on other two datasets. ESPRESSO still ran only one HASA Pattern Instantiation iteration. Table 3 shows that both the precision and number of HASA relationships are worse than ISA relationships for all algorithms. This is because in a corpus, HASA relationships are not as frequent as ISA relationships.

LASER outperforms ESPRESSO in every case for all datasets. One thing to note is that LASER only extends ISA SCIPs in the SCIP Extension step (Section 4.2), but ESPRESSO extends both ISA and HASA SCIPs. We made this choice because HASA has different semantic meanings from ISA and contains many subtypes [12]. For example, HASA(treatment of occlusive disease, endarterectomy) is a valid relationship from MED, but its head word extension HASA(treatment, endarterectomy) does not make sense because "treatment" is too abstract that "endarterectomy" is not part of "treatment" in the general sense. ESPRESSO's drop in precision when it applies HASA headword extension also reflects this.

We have also investigated that the precision, recall, and F-score for HASA at various top-X percentages are very similar to the case for ISA (Figure 5). LASER's precision is again stable and overall, it outperforms ESPRESSO.

5.5 Comparison of Running Time

Table 5 shows the running times for extracting ISA relationships. LASER and LASER HW have the same running time because both versions of LASER require headword extension for seed generation; the only difference is whether we count these extended relationships during evaluation. This is also true for ESPRESSO and ESPRESSO HW.

LASER is the most efficient algorithm and is between 1.4 times and two orders of magnitude faster than other algorithms. Indeed LASER's constrained sequential pattern mining approach to finding new patterns is much more efficient than ESPRESSO's frequent substring finding using a suffix tree. ESPRESSO HW+W takes even longer because search engines constrain the frequencies of queries. This can only get worse as more search engines limit their access. The running time for GHC is quadratic in the number of input terms because agglomerative clustering requires pairwise term similarity. This becomes GHC's bottleneck when the number of input terms gets larger. Indeed, it takes GHC more than two days to finish on an input of 5,000 terms! The bottleneck for WCL-HG is matching each input sentence against all WCLs to find the potential matches. This can take up to 2 days with web definition expansions over MED. The running time for extracting HASA relationships is similar to the ISA case, and we omit the results for lack of space.

Table 5: ISA extraction running time (in seconds)

System	AEC	LP	MED
LASER	65	107	6,862
ESPRESSO	518	308	9,627
ESPRESSO HW+W	10,939	10,439	72,154
WCL-HG	894	224	41 hours
WCL-HG+DEF	2,710	1043	2 days
GHC	1,160	1,709	>2 days

In summary, LASER (equals or) outperforms the other algorithms on precision, relative recall and F-score for both ISA and HASA relationships in most cases. While ESPRESSO suffers from low recall and GHC finds too many "trivial" relationships, LASER outputs a large number of relationships on both small and large corpora, which shows the superiority of using an iterative framework that starts from reliable seed patterns. Using parse tree information and identifying appropriate noun phrases from nested noun phrases, contribute to the discovery of more complex and accurate relationships. This parse-once-use-many-times strategy and the adaptation of constrained frequent sequential pattern mining make LASER very efficient, while the competing algorithms have serious running time bottlenecks.

6. CONCLUSIONS AND FUTURE WORK

Many state-of-the-art algorithms for learning ontologies from free text confine themselves to concepts represented as single-word terms or common compounds. In contrast, we find a richer ontology by covering multi-word terms. We build on and extend previous pattern-based iterative frameworks [14, 22], and make the following contributions: (1) We identify concepts in ISA/HASA relationships by analyzing parse trees instead of simple POS tags, and use an efficient parse-once-use-many-times strategy. (2) We develop a novel algorithm to determine the appropriate noun phrases from nested noun phrases present in the corpus. (3) We tailor sequential pattern mining to find constrained frequent patterns consisting of words, POS tags, and wildcards.

We empirically show on three real web datasets that LASER stably extracts rich and complex concepts and ISA/HASA relationships between them, regardless of the size of corpus or data sparsity. In terms of precision, it is comparable to or

better than the competitors while in terms of relative recall and F-scores it significantly and consistently outperforms them. Finally, we show that LASER has a significantly better running time compared to the competing algorithms and better scales.

An interesting future challenge is to post-process concepts found by LASER with statistical methods to boost the precision even further while maintaining scalability.

7. REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *ICSW*, 2007.

[2] M. Berland and E. Charniak. Finding parts in very large corpora. In *ACL*, pages 57–64. ACL, 1999.

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

[4] S. Brin. Extracting patterns and relations from the world wide web. In *WebDB*, 1999.

[5] S. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *ACL*, 1999.

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

[7] P. Cimiano and J. Völker. Text2onto. *Natural Language Processing and Information Systems*, pages 227–238, 2005.

[8] T. Cover and J. Thomas. *Elements of information theory*, volume 6. Wiley Online Library, 1991.

[9] P. Derose, W. Shen, F. Chen, A. Doan, and R. Ramakrishnan. Building structured web community portals: A top-down, compositional, and incremental approach. In *VLDB*, 2007.

[10] E. Drymonas, K. Zervanou, and E. Petrakis. Unsupervised ontology acquisition from plain texts: the OntoGain system. *Natural Language Processing and Information Systems*, pages 277–287, 2010.

[11] K. Frantzi, S. Ananiadou, and H. Mima. Automatic recognition of multi-word terms:. the c-value/nc-value method. *International Journal on Digital Libraries*, 3(2):115–130, 2000.

[12] R. Girju, A. Badulescu, and D. Moldovan. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1):83–135, 2006.

[13] Z. Harris. Distributional structure. *Word*, 1954.

[14] M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*. ACL, 1992.

[15] W. Hersh, C. Buckley, T. Leone, and D. Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *SIGIR*, 1994.

[16] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *Arxiv preprint cmp-lg/9709008*, 1997.

[17] A. Maedche and S. Staab. Semi-automatic engineering of ontologies from text. In *SEKE*, pages 231–239, 2000.

[18] C. Manning and H. Schütze. *Foundations of statistical natural language processing*, volume 59. MIT Press, 1999.

[19] A. Moosavi, T. Li, L. Lakshmanan, and R. Pottinger. Ontectas: Bridging the gap between collaborative tagging systems and structured data. In *CAiSE*, 2011.

[20] R. Navigli and P. Velardi. Learning word-class lattices for definition and hypernym extraction. In *ACL*, pages 1318–1327, 2010.

[21] R. Navigli, P. Velardi, and S. Faralli. A graph-based algorithm for inducing lexical taxonomies from scratch. In *IJCAI*, 2011.

[22] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *COLING*, pages 113–120. ACL, 2006.

[23] P. Pantel and D. Ravichandran. Automatically labeling semantic classes. In *HLT/NAACL*, 2004.

[24] T. Pedersen, S. Patwardhan, and J. Michelizzi. WordNet:: Similarity: measuring the relatedness of concepts. In *HLT/NAACL*, pages 38–41, 2004.

[25] H. Poon and P. Domingos. Unsupervised ontology induction from text. In *ACL*, pages 296–305, 2010.

[26] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *JAIR*, 11:95–130, 1999.

[27] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *SIGIR*, pages 206–213, 1999.

[28] Srikant and Agrawal. Mining sequential patterns: Generalizations and performance improve. *EDBT*, 1996.

[29] F. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[30] F. Suchanek, M. Sozio, and G. Weikum. SOFIE: A self-organizing framework for information extraction. In *WWW*, 2009.

[31] F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In *CIKM*, pages 41–50, 2007.

[32] F. Wu and D. S. Weld. Automatically refining the wikipedia infobox ontology. In *WWW*, pages 635–644, 2008.

[33] E. Zavitsanos, G. Paliouras, G. Vouros, and S. Petridis. Learning subsumption hierarchies of ontology concepts from texts. *WIAS*, 8(1):37–51, 2010.