Learning First-Order Horn Clauses from Web Text

Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld
Turing Center
University of Washington
Computer Science and Engineering
Box 352350
Seattle, WA 98125, USA
stef,etzioni,weld@cs.washington.edu

Jesse Davis
Katholieke Universiteit Leuven
Department of Computer Science
POBox 02402 Celestijnenlaan 200a
B-3001 Heverlee, Belgium
jesse.davis@cs.kuleuven.be

Abstract

Even the entire Web corpus does not *explicitly* answer all questions, yet inference can uncover many *implicit* answers. But where do inference rules come from?

This paper investigates the problem of *learning* inference rules from Web text in an unsupervised, domain-independent manner. The SHERLOCK system, described herein, is a first-order learner that acquires over 30,000 Horn clauses from Web text. SHERLOCK embodies several innovations, including a novel rule scoring function based on Statistical Relevance (Salmon et al., 1971) which is effective on ambiguous, noisy and incomplete Web extractions. Our experiments show that inference over the learned rules discovers three times as many facts (at precision 0.8) as the TEXTRUNNER system which merely extracts facts explicitly stated in Web text.

1 Introduction

Today's Web search engines locate pages that match keyword queries. Even sophisticated Web-based Q/A systems merely locate pages that contain an *explicit* answer to a question. These systems are helpless if the answer has to be inferred from multiple sentences, possibly on different pages. To solve this problem, Schoenmackers *et al.*(2008) introduced the HOLMES system, which infers answers from tuples extracted from text.

HOLMES's distinction is that it is domain independent and that its inference time is *linear* in the size of its input corpus, which enables it to scale to the Web. However, HOLMES's Achilles heel is that it requires hand-coded, first-order, Horn clauses as input. Thus, while HOLMES's inference *run time* is highly scalable, it requires substantial labor and expertise to hand-craft the appropriate set of Horn clauses for each new domain.

Is it possible to *learn* effective first-order Horn clauses automatically from Web text in a domain-independent and scalable manner? We refer to the set of ground facts derived from Web text as *open-domain theories*. Learning Horn clauses has been studied extensively in the Inductive Logic Programming (ILP) literature (Quinlan, 1990; Muggleton, 1995). However, learning Horn clauses from open-domain theories is particularly challenging for several reasons. First, the theories denote instances of an unbounded and unknown set of relations. Second, the ground facts in the theories are noisy, and incomplete. Negative examples are mostly absent, and certainly we cannot make the closed-world assumption typically made by ILP systems. Finally, the names used to denote both entities and relations are rife with both synonyms and polysymes making their referents ambiguous and resulting in a particularly noisy and ambiguous set of ground facts.

This paper presents a new ILP method, which is optimized to operate on open-domain theories derived from massive and diverse corpora such as the Web, and experimentally confirms both its effectiveness and superiority over traditional ILP algorithms in this context. Table 1 shows some example rules that were learned by SHERLOCK.

This work makes the following contributions:

1. We describe the design and implementation of the SHERLOCK system, which utilizes a novel, unsupervised ILP method to learn first-order Horn clauses from open-domain Web text.

IsHeadquarteredIn(Company, State) :- IsBasedIn(Company, City) ∧ IsLocatedIn(City, State);
Contains(Food, Chemical) :- IsMadeFrom(Food, Ingredient) ∧ Contains(Ingredient, Chemical);
Reduce(Medication, Factor) :- KnownGenericallyAs(Medication, Drug) ∧ Reduce(Drug, Factor);
ReturnTo(Writer, Place) :- BornIn(Writer, City) ∧ CapitalOf(City, Place);
Make(Company1, Device) :- Buy(Company1, Company2) ∧ Make(Company2, Device);

Table 1: Example rules learned by SHERLOCK from Web extractions. Note that the italicized rules are unsound.

2. We derive an innovative scoring function that is particularly well-suited to unsupervised learning from noisy text. For Web text, the scoring function yields more accurate rules than several functions from the ILP literature.
3. We demonstrate the utility of SHERLOCK's automatically learned inference rules. Inference using SHERLOCK's learned rules identifies three times as many high quality facts (*e.g.*, precision $\geq$ 0.8) as were originally extracted from the Web text corpus.

The remainder of this paper is organized as follows. We start by describing previous work. Section 3 introduces the SHERLOCK rule learning system, with Section 3.4 describing how it estimates rule quality. We empirically evaluate SHERLOCK in Section 4, and conclude.

2 Previous Work

SHERLOCK is one of the first systems to learn first-order Horn clauses from open-domain Web extractions. The learning method in SHERLOCK belongs to the Inductive logic programming (ILP) subfield of machine learning (Lavrac and Dzeroski, 2001). However, classical ILP systems (*e.g.*, FOIL (Quinlan, 1990) and Progol (Muggleton, 1995)) make strong assumptions that are inappropriate for open domains. First, ILP systems assume high-quality, hand-labeled training examples for each relation of interest. Second, ILP systems assume that constants uniquely denote individuals; however, in Web text strings such as "dad" or "John Smith" are highly ambiguous. Third, ILP system typically assume complete, largely noise-free data whereas tuples extracted from Web text are both noisy and radically incomplete. Finally, ILP systems typically utilize negative examples, which are not available when learning from open-domain facts. One system that does not require negative examples is LIME (McCreath and Sharma, 1997); We compare SHERLOCK with LIME's methods in Section 4.3. Most prior ILP and Markov logic structure learning systems (*e.g.*, (Kok and Domingos, 2005)) are not designed to handle the noise and incompleteness of open-domain, extracted facts.

NELL (Carlson et al., 2010) performs coupled semi-supervised learning to extract a large knowledge base of instances, relations, and inference rules, bootstrapping from a few seed examples of each class and relation of interest and a few constraints among them. In contrast, SHERLOCK focuses mainly on learning inference rules, but does so without any manually specified seeds or constraints.

Craven *et al.*(1998) also used ILP to help information extraction on the Web, but required training examples and focused on a single domain.

Two other notable systems that learn inference rules from text are DIRT (Lin and Pantel, 2001) and RESOLVER (Yates and Etzioni, 2007). However, both DIRT and RESOLVER learn only a limited set of rules capturing synonyms, paraphrases, and simple entailments, not more expressive multi-part Horn clauses. For example, these systems may learn the rule X acquired $Y \implies X$ bought Y, which captures different ways of describing a purchase. Applications of these rules often depend on context (*e.g.*, if a person acquires a skill, that does not mean they bought the skill). To add the necessary context, ISP (Pantel et al., 2007) learned selectional preferences (Resnik, 1997) for DIRT's rules. The selectional preferences act as type restrictions

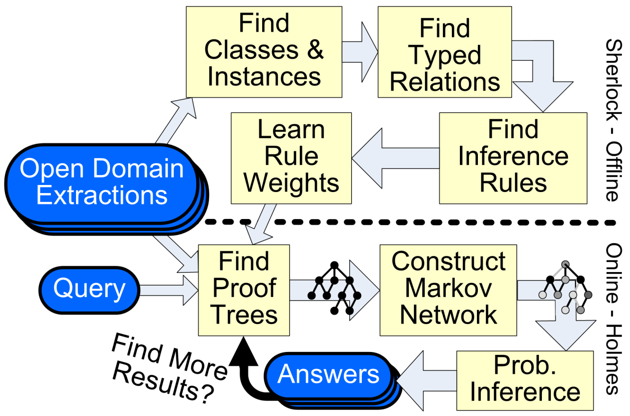

Figure 1: Architecture of SHERLOCK. SHERLOCK learns inference rules offline and provides them to the HOLMES inference engine, which uses the rules to answer queries.

on the arguments, and attempt to filter out incorrect inferences. While these approaches are useful, they are strictly more limited than the rules learned by SHERLOCK.

The Recognizing Textual Entailment (RTE) task (Dagan et al., 2005) is to determine whether one sentence entails another. Approaches to RTE include those of Tatu and Moldovan (2007), which generates inference rules from WordNet lexical chains and a set of axiom templates, and Pennacchiotti and Zanzotto (2007), which learns inference rules based on similarity across entailment pairs. In contrast with this work, RTE systems reason over full sentences, but benefit by being given the sentences and training data. SHERLOCK operates over simpler Web extractions, but is not given guidance about which facts may interact.

3 System Description

SHERLOCK takes as input a large set of open domain facts, and returns a set of weighted Horn-clause inference rules. Other systems (*e.g.*, HOLMES) use the rules to answer questions, infer additional facts, *etc.*

SHERLOCK's basic architecture is depicted in Figure 1. To learn inference rules, SHERLOCK performs the following steps:

1. Identify a "productive" set of classes and instances of those classes
2. Discover relations between classes
3. Learn inference rules using the discovered relations and determine the confidence in each rule

The first two steps help deal with the synonyms, homonyms, and noise present in open-domain theories by identifying a smaller, cleaner, and more cohesive set of facts to learn rules over.

SHERLOCK learns inference rules from a collection of open-domain extractions produced by TEXTRUNNER (Banko et al., 2007). The rules learned by SHERLOCK are input to an inference engine and used to find answers to a user's query. In this paper, SHERLOCK utilizes HOLMES as its inference engine when answering queries, and uses extracted facts of the form `R(arg1, arg2)` provided by the authors of TEXTRUNNER, but the techniques presented are more broadly applicable.

3.1 Finding Classes and Instances

SHERLOCK first searches for a set of well-defined classes and class instances. Instances of the same class tend to behave similarly, so identifying a good set of instances will make it easier to discover the general properties of the entire class.

Options for identifying interesting classes include manually created methods (WordNet (Miller et al., 1990)), textual patterns (Hearst, 1992), automated clustering (Lin and Pantel, 2002), and combinations (Snow et al., 2006). We use Hearst patterns because they are simple, capture how classes and instances are mentioned in Web text, and yield intuitive, explainable groups.

Hearst (1992) identified a set of textual patterns which indicate hyponymy (*e.g.*, '*Class* such as *Instance*'). Using these patterns, we extracted 29 million (instance, class) pairs from a large Web crawl. We then cleaned them using word stemming, normalization, and by dropping modifiers.

Unfortunately, the patterns make systematic errors (*e.g.*, extracting Canada as the name of a city from the phrase 'Toronto, Canada and other cities.') To address this issue, we discard the low frequency classes of each instance. This heuristic reduces the noise due to systematic error while still capturing the important senses of each word. Additionally, we use the extraction frequency to estimate the probability that a particular mention of an instance refers to each of its potential classes (*e.g.*, New York appears as a city 40% of the time, a state 35% of the time, and a place, area, or center the rest of the time).

Ambiguity presents a significant obstacle when learning inference rules. For example, the corpus contains the sentences 'broccoli contains this vitamin' and 'this vitamin prevents scurvy,' but it is unclear if the sentences refer to the same vitamin. The two main sources of ambiguity we observed are references to a more general class instead of a specific instance (*e.g.*, 'vitamin'), and references to a person by only their first or last name. We eliminate the first by removing terms that frequently appear as the class name with other instances, and the second by removing common first and last names.

The 250 most frequently mentioned class names include a large number of interesting classes (*e.g.*, companies, cities, foods, nutrients, locations) as well as ambiguous concepts (*e.g.*, ideas, things). We focus on the less ambiguous classes by eliminating any class not appearing as a descendant of physical entity, social group, physical condition, or event in WordNet. Beyond this filtering we make no use of a type hierarchy and treat classes independently.

In our corpus, we identify 1.1 million distinct, cleaned (instance, class) pairs for 156 classes.

3.2 Discovering Relations between Classes

Next, SHERLOCK discovers how classes relate to and interact with each other. Prior work in relation discovery (Shinyama and Sekine, 2006) has investigated the problem of finding relationships between different classes. However, the goal of this work is to learn rules on top of the discovered typed relations. We use a few simple heuristics to automatically identify interesting relations.

For every pair of classes (C_1, C_2), we find a set of typed, candidate relations from the 100 most frequent relations in the corpus where the first argument is an instance of C_1 and the second argument is an instance of C_2. For extraction terms with multiple senses (*e.g.*, New York), we split their weight based on how frequently they appear with each class in the Hearst patterns.

However, many discovered relations are rare and meaningless, arising from either an extraction error or word-sense ambiguity. For example, the extraction 'Apple is based in Cupertino' gives some evidence that a fruit may possibly be based in a city. We attempt to filter out incorrectly-typed relations using two heuristics. We first discard any relation whose weighted frequency falls below a threshold, since rare relations are more likely to arise due to extraction errors or word-sense ambiguity. We also remove relations whose pointwise mutual information (PMI) is below a threshold T=$exp(2) \approx 7.4$:

$$PMI(R(C_1, C_2)) = \frac{p(R, C_1, C_2)}{p(R, \cdot, \cdot) * p(\cdot, C_1, \cdot) * p(\cdot, \cdot, C_2)}$$

where $p(R, \cdot, \cdot)$ is the probability a random extraction has relation R, $p(\cdot, C_1, \cdot)$ is the probability a random extraction has an instance of C_1 as its first argument, $p(\cdot, \cdot, C_2)$ is similar for the second argument, and $p(R, C_1, C_2)$ is the probability that a random extraction has relation R and instances of C_1 and C_2 as its first and second arguments, respectively. A low PMI indicates the relation occurred by random chance, which is typically due to ambiguous terms or extraction errors.

Finally, we use two TEXTRUNNER specific cleaning heuristics: we ignore a small set of stop-relations ('be', 'have', and 'be *preposition*') and extractions whose arguments are more than four tokens apart. This process identifies 10,000 typed relations.

3.3 Learning Inference Rules

SHERLOCK attempts to learn inference rules for each typed relation in turn. SHERLOCK receives a target relation, R, a set of observed examples of the relation, E^+, a maximum clause length k, a minimum support, s, and an acceptance threshold, t, as input. SHERLOCK generates all first-order, definite clauses up to length k, where R appears as the head of the clause. It retains each clause that:

1. Contains no unbound variables
2. Infers at least s examples from E^+
3. Scores at least t according to the score function

We now propose a novel score function, and empirically validate our choice in Sections 4.3 and 4.4.

3.4 Evaluating Rules by Statistical Relevance

The problem of evaluating candidate rules has been studied by many researchers, but typically in either a supervised or propositional context whereas we are learning first-order Horn-clauses from a noisy set of positive examples. Moreover, due to the incomplete nature of the input corpus and the imperfect yield of extraction—many true facts are not stated explicitly in the set of ground assertions used by the learner to evaluate rules.

The absence of negative examples, coupled with noise, means that standard ILP evaluation functions (*e.g.*, (Quinlan, 1990) and (Dzeroski and Bratko, 1992)) are not appropriate. Furthermore, when evaluating a particular rule with consequent C and antecedent A, it is natural to consider $p(C|A)$ but, due to missing data, this absolute probability estimate is often misleading: in many cases C will hold given A but the fact C is not mentioned in the corpus.

Thus to evaluate rules over extractions, we need to consider *relative* probability estimates. *I.e.*, is $p(C|A) \gg p(C)$? If so, then A is said to be *statistically relevant* to C (Salmon et al., 1971).

Statistical relevance tries to infer the simplest set of factors which explain an observation. It can be viewed as searching for the simplest propositional Horn-clause which increases the likelihood of a goal proposition g. The two key ideas in determining statistical relevance are discovering factors which substantially increase the likelihood of g (even if the probabilities are small in an absolute sense), and dismissing irrelevant factors.

To illustrate these concepts, consider the following example. Suppose our goal is to predict if New York City will have a storm (S). On an arbitrary day, the probability of having a storm is fairly low ($p(S) \ll 1$). However, if we know that the atmospheric pressure on that day is low, this substantially increases the probability of having a storm (although that absolute probability may still be small). According to the principle of statistical relevance, low atmospheric pressure (LP) is a factor which predicts storms (S :- LP), since $p(S|LP) \gg p(S)$.

The principle of statistical relevance also identifies and removes irrelevant factors. For example, let M denote the gender of New York's mayor. Since $p(S|LP, M) \gg p(S)$, it naïvely appears that storms in New York depend on the gender of the mayor in addition to the air pressure. The statistical relevance principle sidesteps this trap by removing any factors which are conditionally independent of the goal, given the remaining factors. For example, we observe $p(S|LP)$=$p(S|LP, M)$, and so we say that M is not statistically relevant to S. This test applies Occam's razor by searching for the simplest rule which explains the goal.

Statistical relevance appears useful in the open-domain context, since all the necessary probabilities can be estimated from only positive examples. Furthermore, approximating relative probabilities in the presence of missing data is much more reliable than determining absolute probabilities.

Unfortunately, Salmon defined statistical relevance in a propositional context. One technical contribution of our work is to lift statistical relevance to first order Horn-clauses as follows. For the Horn-clause $\mathtt{Head}(\mathtt{v_1}, ..., \mathtt{v_n})$:-$\mathtt{Body}(\mathtt{v_1}, ..., \mathtt{v_m})$ (where $\mathtt{Body}(\mathtt{v_1}, ..., \mathtt{v_m})$ is a conjunction of function-free, non-negated, first-order relations, and $v_i \in V$ is the set of typed variables used in the rule), we say the body helps explain the head if:

1. Observing an instance of the body substantially increases the probability of observing the head.
2. The body contains no irrelevant (conditionally independent) terms.

We evaluate conditional independence of terms using ILP's technique of Θ-subsumption, ensuring there is no more general clause that is similarly predictive of the head. Formally, clause C_1 Θ-subsumes clause C_2 if and only if there exists a substitution Θ such that $C_1\Theta \subseteq C_2$ where each clause is treated as the set of its literals. For example, $\mathtt{R}(\mathtt{x}, \mathtt{y})$ Θ-subsumes $\mathtt{R}(\mathtt{x}, \mathtt{x})$, since $\{\mathtt{R}(\mathtt{x}, \mathtt{y})\}\Theta \subseteq \{\mathtt{R}(\mathtt{x}, \mathtt{x})\}$ when Θ=$\{\mathtt{y}/\mathtt{x}\}$. Intuitively, if C_1 Θ-subsumes C_2, it means that C_1 is more general than C_2.

Definition 1 *A first-order Horn-clause* $\mathtt{Head}(...)$:-$\mathtt{Body}(...)$ *is* statistically relevant *if* $p(\mathtt{Head}(...)|\mathtt{Body}(...)) \gg p(\mathtt{Head}(...))$ *and if there is no clause body* $\mathtt{B}'(...)\Theta \subseteq \mathtt{Body}(...)$ *such that* $p(\mathtt{Head}(...)|\mathtt{Body}(...)) \approx p(\mathtt{Head}(...)|\mathtt{B}'(...))$

In practice it is difficult to determine the probabilities exactly, so when checking for statistical relevance we ensure that the probability of the rule is at least a factor t greater than the probability of any subsuming rule, that is, $p(\mathtt{Head}(...)|\mathtt{Body}(...)) \geq t * p(\mathtt{Head}(...)|\mathtt{B}'(...))$

We estimate $p(\mathtt{Head}(...)|\mathtt{B}(...))$ from the observed facts by assuming values of $\mathtt{Head}(...)$ are generated by sampling values of $\mathtt{B}(...)$ as follows: for variables v_s shared between $\mathtt{Head}(...)$ and $\mathtt{B}(...)$, we sample values of v_s uniformly from all observed groundings of B(...). For variables v_i, if any, that appear in Head(...) but not in B(...), we sample their values according to a distribution $p(v_i|class_i)$. We estimate $p(v_i|class_i)$ based on the relative frequency that v_i was extracted using a Hearst pattern with $class_i$.

Finally, we ensure the differences are statistically significant using the likelihood ratio statistic:

$$2N_r \sum_{\substack{\texttt{H(...)}\in \\ \{\texttt{Head(...)},\neg\texttt{Head(...)}\}}} p(\texttt{H(...)}|\texttt{Body(...)}) * log\frac{p(\texttt{H(...)}|\texttt{Body(...)})}{p(\texttt{H(...)}|\texttt{B}'\texttt{(...)})}$$

where $p(\neg\texttt{Head(...)}|\texttt{B(...)}) = 1-p(\texttt{Head(...)}|\texttt{B(...)})$ and N_r is the number of results inferred by the rule Head(...):-Body(...). This test is distributed approximately as χ^2 with one degree of freedom. It is similar to the *statistical significance* test used in mFOIL (Dzeroski and Bratko, 1992), but has two modifications since SHERLOCK doesn't have training data. In lieu of positive and negative examples, we use whether or not the inferred head value was observed, and compare against the distribution of a subsuming clause B′(...) rather than a known prior.

This method of evaluating rules has two important differences from ILP under a closed world assumption. First, our probability estimates consider the fact that examples provide varying amounts of information. Second, statistical relevance finds rules with large increases in relative probability, not necessarily a large absolute probability. This is crucial in an open domain setting where most facts are false, which means the trivial rule that everything is false will have high accuracy. Even for true rules, the observed estimates $p(\texttt{Head(...)}|\texttt{Body(...)}) \ll 1$ due to missing data and noise.

3.5 Making Inferences

In order to benefit from learned rules, we need an inference engine; with its linear-time scalability, HOLMES is a natural choice (Schoenmackers et al., 2008). As input HOLMES requires a target atom H(...), an evidence set E and weighted rule set R as input. It performs a form of *knowledge based model construction* (Wellman et al., 1992), first finding facts using logical inference, then estimating the confidence of each using a Markov Logic Network (Richardson and Domingos, 2006).

Prior to running inference, it is necessary to assign a weight to each rule learned by SHERLOCK. Since the rules and inferences are based on a set of noisy and incomplete extractions, the algorithms used for both weight learning and inference should capture the following characteristics of our problem:

C1. Any arbitrary unknown fact is highly unlikely to be true.

C2. The more frequently a fact is extracted from the Web, the more likely it is to be true. However, facts in E should have a confidence bounded by a threshold $p_{max} < 1$. E contains systematic extraction errors, so we want uncertainty in even the most frequently extracted facts.

C3. An inference that combines uncertain facts should be less likely than each fact it uses.

Next, we describe the needed modifications to the weight learning and inference algorithm to achieve the desired behavior.

3.5.1 Weight Learning

We use the discriminative weight learning procedure described by Huynh and Mooney (2008). Setting the weights involves counting the number of true groundings for each rule in the data (Richardson and Domingos, 2006). However, the noisy nature of Web extractions will make this an overestimate. Consequently, we compute $n_i(E)$, the number of true groundings of rule i, as follows:

$$n_i(E) = \sum_j \max_k \prod_{B(...)\in Body_{ijk}} p(B(...)) \qquad (1)$$

where E is the evidence, j ranges over heads of the rule, $Body_{ijk}$ is the body of the kth grounding for jth head of rule i, and $p(B(...))$ is approximated using a logistic function of the number of times $B(...)$ was extracted,[1] scaled to be in the range [0,0.75]. This models **C2** by giving increasing but bounded confidence for more frequently extracted facts. In practice, this also helps address **C3** by giving longer rules smaller values of n_i, which reflects that inferences arrived at through a combination of multiple, noisy facts should have lower confidence. Longer rules are also more likely to have multiple groundings that infer a particular head, so keeping only the most likely grounding prevents a head from receiving undue weight from any single rule.

[1]We note that this approximation is equivalent to an MLN which uses only the two rules defined in 3.5.2

Finally, we place a very strong Gaussian prior (*i.e.*, L_2 penalty) on the weights. Longer rules have a higher prior to capture the notion that they are more likely to make incorrect inferences. Without a high prior, each rule would receive an unduly high weight as we have no negative examples.

3.5.2 Probabilistic Inference

After learning the weights, we add the following two rules to our rule set:

1. H(...) with negative weight w_{prior}
2. H(...):-ExtractedFrom(H(...),$sentence_i$) with weight 1

The first rule models **C1**, by saying that most facts are false. The second rule models **C2**, by stating the probability of fact depends on the number of times it was extracted. The weights of these rules are fixed. We do not include these rules during weight learning as doing so swamps the effects of the other inference rules (*i.e.*, forces them to zero).

HOLMES attempts to infer the truth value of each ground atom H(...) in turn by treating all other extractions E in our corpus as evidence. Inference also requires computing $n_i(E)$ which we do according to Equation 1 as in weight learning.

4 Experiments

One can attempt to evaluate a rule learner by estimating the quality of learned rules, or by measuring their impact on a system that uses the learned rules. Since the notion of 'rule quality' is vague except in the context of an application, we evaluate SHERLOCK in the context of the HOLMES inference-based question answering system.

Our evaluation focuses on three main questions:

1. Does inference utilizing learned Horn rules improve the precision/recall of question answering and by how much?
2. How do different rule-scoring functions affect the performance of learning?
3. What role does each of SHERLOCK's components have in the resulting performance?

4.1 Methodology

Our objective with rule learning was to improve the system's ability to answer questions such as 'What foods prevent disease?' So we focus our evaluation on the task of computing as many instances as possible of an atomic pattern Rel(x, y). In this example, Rel would be bound to 'Prevents', x would have type 'Food' and y would have type 'Disease.'

But which relations should be used in the test? There is a large variance in behavior across relations, so examining any particular relation may give misleading results. Instead, we examine the global performance of the system by querying HOLMES for all open-domain relations identified in Section 3.2 as follows:

1. Score all candidate rules according to the rule scoring metric M, accept all rules with a score at least t_M (tuned on a small development set of rules), and learn weights for all accepted rules.
2. Find all facts inferred by the rules and use the rule weights to estimate the fact probabilities.
3. Reduce type information. For each fact, (*e.g.*, BasedIn(Diebold, Ohio)) which has been deduced with multiple type signatures (*e.g.*, Ohio is both a state and a geographic location), keep only the one with maximum probability (*i.e.*, conservatively assuming dependence).
4. Place all results into bins based on their probabilities, and estimate the precision and the number of correct facts in the bin using a random sample.

In these experiments we consider rules with up to $k = 2$ relations in the body. We use a corpus of 1 million raw extractions, corresponding to 250,000 distinct facts. SHERLOCK found 5 million candidate rules that infer at least two of the observed facts. Unless otherwise noted, we use SHERLOCK's rule scoring function to evaluate the rules (Section 3.4).

The results represent a wide variety of domains, covering a total of 10,672 typed relations. We observe between a dozen and 2,375 distinct, ground facts for each relation. SHERLOCK learned a total of 31,000 inference rules.[2] Learning all rules, rule

[2]The learned rules are available at: http://www.cs.washington.edu/research/sherlock-hornclauses/

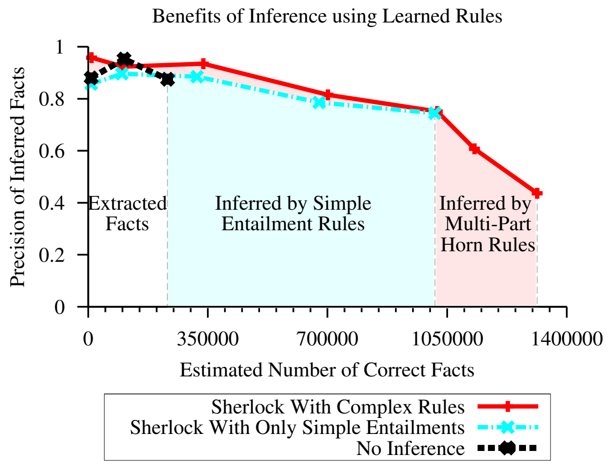

Figure 2: Inference discovers many facts which are not explicitly extracted, identifying 3x as many high quality facts (precision 0.8) and more than 5x as many facts overall. Horn-clauses with multiple relations in the body infer 30% more correct facts than are identified by simpler entailment rules, inferring many facts not present in the corpus in any form.

weights, and performing the inference took 50 minutes on a 72 core cluster. However, we note that for half of the relations SHERLOCK accepts no inference rules, and remind the reader that the performance on any particular relation may be substantially different, and depends on the facts observed in the corpus and on the rules learned.

4.2 Benefits of Inference

We first evaluate the utility of the learned Horn rules by contrasting the precision and number of correct and incorrect facts identified with and without inference over learned rules. We compare against two simpler variants of SHERLOCK. The first is a no-inference baseline that uses no rules, returning only facts that are explicitly extracted. The second baseline only accepts rules of length $k = 1$, allowing it to make simple entailments but not more complicated inferences using multiple facts.

Figure 2 compares the precision and estimated number of correct facts with and without inference. As is apparent, the learned inference rules substantially increase the number of known facts, quadrupling the number of correct facts beyond what are explicitly extracted.

The Horn rules having a body-length of two identify 30% more facts than the simpler length-one rules. Furthermore, we find the Horn rules yield slightly increased precision at comparable levels of recall, although the increase is not statistically significant. This behavior can be attributed to learning smaller weights for the length-two rules than the length-one rules, allowing the length-two rules provide a small amount of additional evidence as to which facts are true, but typically not enough to overcome the confidence of a more reliable length-one rule.

Analyzing the errors, we found that about one third of SHERLOCK's mistakes are due to metonymy and word sense ambiguity (*e.g.*, confusing Vancouver, British Columbia with Vancouver, Washington), one third are due to inferences based on incorrectly-extracted facts (*e.g.*, inferences based on the incorrect fact $\texttt{IsLocatedIn}(\texttt{New York}, \texttt{Suffolk County})$, which was extracted from sentences like 'Deer Park, New York is located in Suffolk County'), and the rest are due to unsound or incorrect inference rules (*e.g.*, $\texttt{BasedIn}(\texttt{Company}, \texttt{City})$:- $\texttt{BasedIn}(\texttt{Company}, \texttt{Country}) \wedge \texttt{CapitalOf}(\texttt{City}, \texttt{Country})$). Without negative examples it is difficult to distinguish correct rules from these unsound rules, since the unsound rules are correct more often than expected by chance.

Finally, we note that although simple, length-one rules capture many of the results, in some respects they are just rephrasing facts that are extracted in another form. However, the more complex, length-two rules synthesize facts extracted from multiple pages, and infer results that are not stated anywhere in the corpus.

4.3 Effect of Scoring Function

We now examine how SHERLOCK's rule scoring function affects its results, by comparing it with three rule scoring functions used in prior work:

LIME. The LIME ILP system (McCreath and Sharma, 1997) proposed a metric that generalized Muggleton's (1997) positive-only score function by modeling noise and limited sample sizes.

M-Estimate of rule precision. This is a common approach for handling noise in ILP (Dzeroski and Bratko, 1992). It requires negative examples, which we generated by randomly swapping arguments between positive examples.

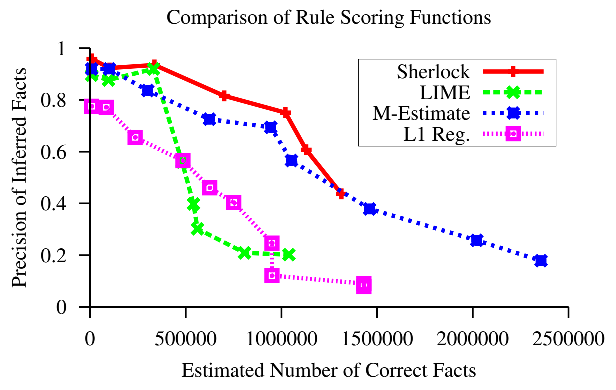

Figure 3: SHERLOCK identifies rules that lead to more accurate inferences over a large set of open-domain extracted facts, deducing 2x as many facts at precision 0.8.

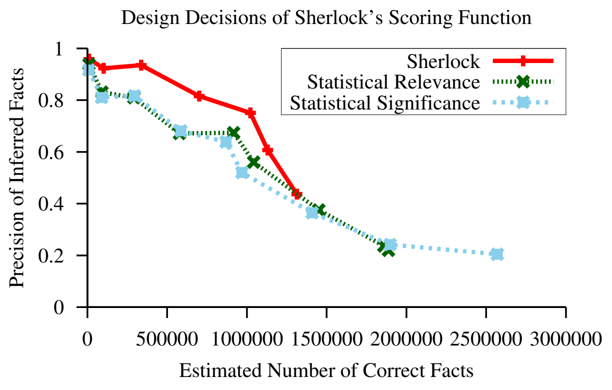

Figure 4: By requiring rules to have both statistical relevance and statistical significance, SHERLOCK rejects many error-prone rules that are accepted by the metrics individually. The better rule set yields more accurate inferences, but identifies fewer correct facts.

L1 Regularization. As proposed in (Huynh and Mooney, 2008), this learns weights for all candidate rules using L_1-regularization (encouraging sparsity) instead of L_2-regularization, and retains only those with non-zero weight.

Figure 3 compares the precision and estimated number of correct facts inferred by the rules of each scoring function. SHERLOCK has consistently higher precision, and finds twice as many correct facts at precision 0.8.

M-Estimate accepted eight times as many rules as SHERLOCK, increasing the number of inferred facts at the cost of precision and longer inference times. Most of the errors in M-Estimate and L1 Regularization come from incorrect or unsound rules, whereas most of the errors for LIME stem from systematic extraction errors.

4.4 Scoring Function Design Decisions

SHERLOCK requires a rule to have statistical relevance and statistical significance. We perform an ablation study to understand how each of these contribute to SHERLOCK's results.

Figure 4 compares the precision and estimated number of correct facts obtained when requiring rules to be only statistically relevant, only statistically significant, or both. As is expected, there is a precision/recall tradeoff. SHERLOCK has higher precision, finding more than twice as many results at precision 0.8 and reducing the error by 39% at a recall of 1 million correct facts. Statistical significance finds twice as many correct facts as SHERLOCK, but the extra facts it discovers have precision < 0.4.

Comparing the rules accepted in each case, we found that statistical relevance and statistical significance each accepted about 180,000 rules, compared to about 31,000 for SHERLOCK. The smaller set of rules accepted by SHERLOCK not only leads to higher precision inferences, but also speeds up inference time by a factor of seven.

In a qualitative analysis, we found the statistical relevance metric overestimates probabilities for sparse rules, which leads to a number of very high scoring but meaningless rules. The statistical significance metric handles sparse rules better, but is still overconfident in the case of many unsound rules.

4.5 Analysis of Weight Learning

Finally, we empirically validate the modifications of the weight learning algorithm from Section 3.5.1.

The learned-rule weights only affect the probabilities of the inferred facts, not the inferred facts themselves, so to measure the influence of the weight learning algorithm we examine the recall at precision 0.8 and the area under the precision-recall curve (AuC). We build a test set by holding SHERLOCK's inference rules constant and randomly sampling 700 inferred facts. We test the effects of:

- Fixed *vs.* Variable Penalty - Do we use the same L_2 penalty on the weights for all rules or a stronger L_2 penalty for longer rules?
- Full *vs.* Weighted Grounding Counts - Do we count all unweighted rule groundings (as in (Huynh and Mooney, 2008)), or only the best weighted one (as in Equation 1)?

	Recall (p=0.8)	AuC
Variable Penalty, Weighted Counts (used by SHERLOCK)	**0.35**	**0.735**
Variable Penalty, Full Counts	0.28	0.726
Fixed Penalty, Weighted Counts	0.27	0.675
Fixed Penalty, Full Counts	0.17	0.488

Table 2: SHERLOCK's modified weight learning algorithm gives better probability estimates over noisy and incomplete Web extractions. Most of the gains come from penalizing longer rules more, but using weighted grounding counts further improves recall by 0.07, which corresponds to almost 100,000 additional facts at precision 0.8.

We vary each of these independently, and give the performance of all 4 combinations in Table 2.

The modifications from Section 3.5.1 improve both the AuC and the recall at precision 0.8. Most of the improvement is due to using stronger penalties on longer rules, but using the weighted counts in Equation 1 improves recall by a factor of 1.25 at precision 0.8. While this may not seem like much, the scale is such that it leads to almost 100,000 additional correct facts at precision 0.8.

5 Conclusion

This paper addressed the problem of learning first-order Horn clauses from the noisy and heterogeneous corpus of open-domain facts extracted from Web text. We showed that SHERLOCK is able to learn Horn clauses in a large-scale, domain-independent manner. Furthermore, the learned rules are valuable, because they infer a substantial number of facts which were not extracted from the corpus.

While SHERLOCK belongs to the broad category of ILP learners, it has a number of novel features that enable it to succeed in the challenging, open-domain context. First, SHERLOCK automatically identifies a set of high-quality extracted facts, using several simple but effective heuristics to defeat noise and ambiguity. Second, SHERLOCK is unsupervised and does not require negative examples; this enables it to scale to an unbounded number of relations. Third, it utilizes a novel rule-scoring function, which is tolerant of the noise, ambiguity, and missing data issues prevalent in facts extracted from Web text. The experiments in Figure 3 show that, for open-domain facts, SHERLOCK's method represents a substantial improvement over traditional ILP scoring functions.

Directions for future work include inducing longer inference rules, investigating better methods for combining the rules, allowing deeper inferences across multiple rules, evaluating our system on other corpora and devising better techniques for handling word sense ambiguity.

Acknowledgements

We thank Sonal Gupta and the anonymous reviewers for their helpful comments. This research was supported in part by NSF grant IIS-0803481, ONR grant N00014-08-1-0431, the WRF / TJ Cable Professorship and carried out at the University of Washington's Turing Center. The University of Washington gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract nos. FA8750-09-C-0179 and FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the Web. In *Procs. of IJCAI*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*.

M. Craven, D. DiPasquo, D. Freitag, A.K. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to Extract Symbolic Knowledge from the World Wide Web. In *Procs. of the 15th Conference of the American Association for Artificial Intelligence*, pages 509–516, Madison, US. AAAI Press, Menlo Park, US.

I. Dagan, O. Glickman, and B. Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 1–8.

S. Dzeroski and I. Bratko. 1992. Handling noise in inductive logic programming. In *Proceedings of the 2nd*

International Workshop on Inductive Logic Programming.

M. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Procs. of the 14th International Conference on Computational Linguistics*, pages 539–545, Nantes, France.

T.N. Huynh and R.J. Mooney. 2008. Discriminative structure and parameter learning for Markov logic networks. In *Proceedings of the 25th international conference on Machine learning*, pages 416–423. ACM.

Stanley Kok and Pedro Domingos. 2005. Learning the structure of markov logic networks. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 441–448, New York, NY, USA. ACM.

N. Lavrac and S. Dzeroski, editors. 2001. *Relational Data Mining*. Springer-Verlag, Berlin, September.

D. Lin and P. Pantel. 2001. DIRT – Discovery of Inference Rules from Text. In *KDD*.

D. Lin and P. Pantel. 2002. Concept discovery from text. In *Proceedings of the 19th International Conference on Computational linguistics (COLING-02)*, pages 1–7.

E. McCreath and A. Sharma. 1997. ILP with noise and fixed example size: a Bayesian approach. In *Proceedings of the Fifteenth international joint conference on Artifical intelligence-Volume 2*, pages 1310–1315. Morgan Kaufmann Publishers Inc.

G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

S. Muggleton. 1995. Inverse entailment and Progol. *New Generation Computing*, 13:245–286.

S. Muggleton. 1997. Learning from positive data. *Lecture Notes in Computer Science*, 1314:358–376.

P. Pantel, R. Bhagat, B. Coppola, T. Chklovski, and E. Hovy. 2007. ISP: Learning inferential selectional preferences. In *Proceedings of NAACL HLT*, volume 7, pages 564–571.

M. Pennacchiotti and F.M. Zanzotto. 2007. Learning Shallow Semantic Rules for Textual Entailment. *Proceedings of RANLP 2007*.

J. R. Quinlan. 1990. Learning logical definitions from relations. *Machine Learning*, 5:239–2666.

Philip Resnik. 1997. Selectional preference and sense disambiguation. In *Proc. of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*

M. Richardson and P. Domingos. 2006. Markov Logic Networks. *Machine Learning*, 62(1-2):107–136.

W.C. Salmon, R.C. Jeffrey, and J.G. Greeno. 1971. *Statistical explanation & statistical relevance*. Univ of Pittsburgh Pr.

S. Schoenmackers, O. Etzioni, and D. Weld. 2008. Scaling Textual Inference to the Web. In *Procs. of EMNLP*.

Y. Shinyama and S. Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Procs. of HLT/NAACL*.

R. Snow, D. Jurafsky, and A. Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *COLING/ACL 2006*.

M. Tatu and D. Moldovan. 2007. COGEX at RTE3. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 22–27.

M.P. Wellman, J.S. Breese, and R.P. Goldman. 1992. From knowledge bases to decision models. *The Knowledge Engineering Review*, 7(1):35–53.

A. Yates and O. Etzioni. 2007. Unsupervised resolution of objects and relations on the Web. In *Procs. of HLT*.