Appears in International Journal On Semantic Web and Information Systems - Special Issue on Web-Scale Knowledge Extraction, 2012

# Elementary: Large-scale Knowledge-base Construction via Machine Learning and Statistical Inference

Feng Niu, Ce Zhang, Christopher Ré, and Jude Shavlik
Computer Sciences Department
University of Wisconsin-Madison
{leonn, czhang, chrisre, shavlik}@cs.wisc.edu

Researchers have approached knowledge-base construction (KBC) with a wide range of data resources and techniques. We present Elementary, a prototype KBC system that is able to combine diverse resources and different KBC techniques via machine learning and statistical inference to construct knowledge bases. Using Elementary, we have implemented a solution to the TAC-KBP challenge with quality comparable to the state of the art, as well as an end-to-end online demonstration that automatically and continuously enriches Wikipedia with structured data by reading millions of webpages on a daily basis. We describe several challenges and our solutions in designing, implementing, and deploying Elementary. In particular, we first describe the conceptual framework and architecture of Elementary, and then discuss how we address scalability challenges to enable Web-scale deployment. First, to take advantage of diverse data resources and proven techniques, Elementary employs Markov logic, a succinct yet expressive language to specify probabilistic graphical models. Elementary accepts both domain-knowledge rules and classical machine-learning models such as conditional random fields, thereby integrating different data resources and KBC techniques in a principled manner. Second, to support large-scale KBC with terabytes of data and millions of entities, Elementary leverages high-throughput parallel computing infrastructure such as Hadoop, Condor, and parallel databases. Furthermore, to scale sophisticated statistical inference, Elementary employs a novel decomposition-based approach to Markov logic inference that solves routine subtasks such as classification and coreference with specialized algorithms. We empirically show that this decomposition-based inference approach achieves higher performance than prior inference approaches. To validate the effectiveness of Elementary's approach to KBC, we experimentally show that its ability to incorporate diverse signals has positive impacts on KBC quality.

*Keywords:* Knowledge-base construction, natural language understanding, information extraction, machine learning, statistical inference, systems

## Introduction

Knowledge-base construction (KBC) is the process of populating a knowledge base (KB) with facts (or assertions) extracted from text. It has recently received tremendous interest from academia (Weikum & Theobald, 2010), e.g., CMU's NELL (Carlson et al., 2010; Lao, Mitchell, & Cohen, 2011), MPI's YAGO (Kasneci, Ramanath, Suchanek, & Weikum, 2008; Nakashole, Theobald, & Weikum, 2011), and from industry (Fang, Sarma, Yu, & Bohannon, 2011), e.g., IBM's DeepQA (Ferrucci et al., 2010) and Microsoft's EntityCube (Zhu, Nie, Liu, Zhang, & Wen, 2009). To construct high-quality knowledge bases from text, researchers have considered a wide range of data resources and techniques; e.g., pattern matching with dictionaries listing entity names (Riloff, 1993), bootstrapping from existing knowledge bases like Freebase and YAGO (Suchanek, Kasneci, & Weikum, 2007), disambiguation using web links and search results (Hoffart et al., 2011; Dredze, McNamee, Rao, Gerber, & Finin, 2010), rule-based extraction with regular expressions curated by domain experts (DeRose et al., 2007; Chiticariu et al., 2010), training statistical models with annotated text (Lafferty, McCallum, & Pereira, 2001), etc. All these resources are valuable because they are complementary in terms of cost, quality, and coverage; ideally one would like to be able to use them all. To take advantage of different kinds of data resources, a major problem that KBC systems face is coping with imperfect or conflicting information from multiple sources (Weikum & Theobald, 2010). (We use the term "information" to refer to both data and algorithms that can be used for a KBC task.) To address this issue, several recent KBC projects (Carlson et al., 2010; Kasneci et al., 2008; Nakashole et al., 2011; Zhu et al., 2009; Lao et

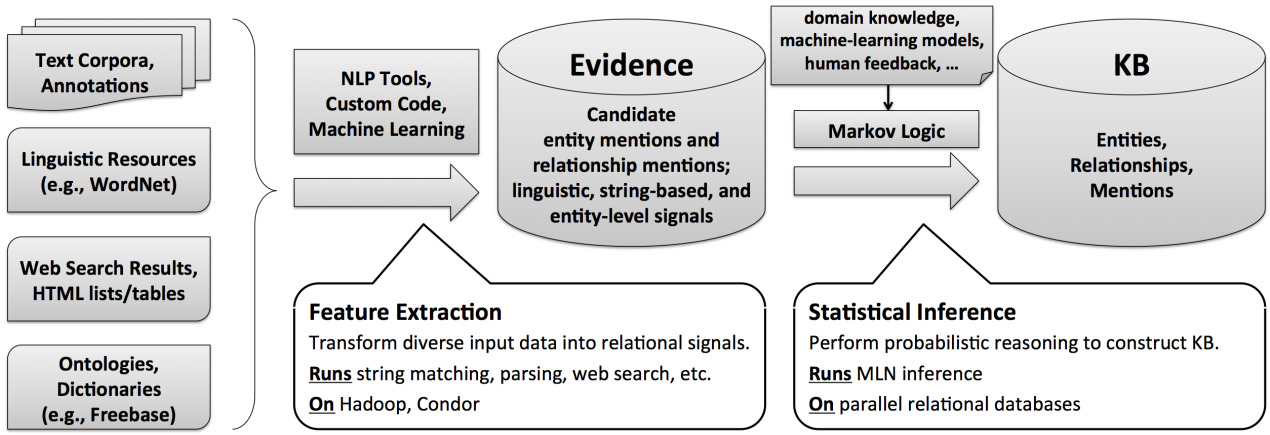


Figure 1. *Architecture of* ELEMENTARY. ELEMENTARY *takes as input diverse data resources, converts them into relational evidence, and then performs statistical inference in Markov Logic to construct knowledge bases.* ELEMENTARY *may invoke machine learning techniques in both feature extraction and the process of constructing the MLN program.*

al., 2011) use statistical inference to combine different data resources.

Motivated by the above observation, we present ELEMENTARY, a prototype system that aims to enable quick development and scalable deployment of KBC systems that combine diverse data resources and best-of-breed algorithms via machine learning and statistical inference (Niu, 2012). This article provides an overview of the motivation and advantages of the ELEMENTARY architecture, while only briefly touching on individual technical challenges that are addressed in our other publications. We structure our presentation around two main challenges that we face in designing, implementing, and deploying ELEMENTARY: (1) how to integrate conflicting information from multiple sources for KBC in a principled way, and (2) how to scale ELEMENTARY for Web-scale KBC.

**Challenge 1: How do we handle conflicting information from multiple sources of data and algorithms for a KBC task in a principled way?** To perform knowledge-base construction with multiple sources of information, a critical challenge is the ability to handle imperfect and conflicting information (**Challenge 1a**). Following several recent KBC projects such as StatSnowball/EntityCube (Zhu et al., 2009) and SOFIE/Prospera (Suchanek, Sozio, & Weikum, 2009; Nakashole et al., 2011), ELEMENTARY uses a probabilistic logic language called *Markov logic* (Richardson & Domingos, 2006) that has been applied to a wide range of text-related applications (Poon & Domingos, 2007; Suchanek et al., 2009; Zhu et al., 2009; Andrzejewski, Livermore, Zhu, Craven, & Recht, 2011). In Markov logic, one can write first-order logic rules with weights (that intuitively model our confidence in a rule); this allows one to capture rules that are likely, but not certain, to be correct. A Markov logic program (aka *Markov logic network*, or simply MLN) specifies what data are available, what predictions to make, and what constraints and correlations there are.

ELEMENTARY adopts the classic Entity-Relationship (ER) model (P. Chen, 1976; Calì, Gottlob, & Pieris, 2010) for the target KB. While there is a direct map between the ER model and predictions in Markov logic, a crucial challenge is how to accommodate information from diverse sources when developing a KBC system (**Challenge 1b**). To address this issue, ELEMENTARY uses a two-phase architecture: as shown in Figure 1, ELEMENTARY first runs a *feature extraction* step to convert input data resources into relational signals called *evidence*; it then feeds evidence into a Markov logic engine where ELEMENTARY performs statistical inference to construct a knowledge base. Besides evidence, one may also translate a source of information into MLN rules. For example, we can translate machine-learning models like logistic regression and conditional random fields (Lafferty et al., 2001) into MLN rules in a straightforward manner. One could also directly add MLN rules to capture prior beliefs, constraints, or correlations over the evidence and the target KB. For directly supplied rules, one may either heuristically specify the weights or perform weight learning (Lowd & Domingos, 2007). To obtain training data for learning, one could either use existing examples such as annotations (if any) or acquire silver-standard examples with the distant-supervision technique (Zhang, Niu, Ré, & Shavlik, 2012).

Although ELEMENTARY provides the mechanism to combine different data and algorithms for KBC, the output quality still depends on *what* input resources are used and *how* they are combined. Thus, a third challenge is how to debug and tune a KBC system built using ELEMENTARY (**Challenge 1c**). Specifically, a developer needs to select or tune data sources, feature extraction

algorithms, the MLN rule set, and MLN rule weights. From our experience of building KBC systems, we find that an effective approach is iterative manual error analysis of the output. To facilitate error analysis, we have developed different tools in ELEMENTARY that provide insights about the how each component of a KBC system impacts output quality.

**Challenge 2: How do we scale ELEMENTARY to Web-scale KBC tasks?** ELEMENTARY processes a KBC task by performing two sequential steps: feature extraction and statistical inference. To scale ELEMENTARY to terabytes of input data, the first challenge is to scale feature extraction (**Challenge 2a**). Toward that goal, we employ high-throughput parallel computing frameworks such as Hadoop[1] and Condor (Thain, Tannenbaum, & Livny, 2005) that can schedule computationally intensive jobs on both cluster machines and idle workstations. For example, using Condor, we were able to utilize hundreds of thousands of machine-hours to perform deep linguistic feature extraction (including named entity recognition and dependency parsing) on a 500M-doc web corpus within one week.

Web-scale KBC involves billions of prediction variables, and so a second challenge is to scale statistical inference in ELEMENTARY (**Challenge 2b**). Following the TUFFY system (Niu, Ré, Doan, & Shavlik, 2011), we leverage mature database technology by implementing data manipulation operations and MLN inference on top of a parallel relational data management system (RDBMS) from Greenplum, Inc. Still, we find that state-of-the-art approaches to MLN inference often do not scale to the amount of data and rich structures involved in a large-scale KBC task. We observe that a KBC task often contains routine subtasks such as classification and coreference resolution; these subtasks have specialized algorithms with high efficiency and quality. Thus, instead of solving a whole Markov logic program with generic inference algorithms, we could split the program into multiple parts and solve subtasks with corresponding specialized algorithms (Niu, Zhang, Ré, & Shavlik, 2011). However, because different subtasks may share the same relation, there may be conflicts between predictions from different subtasks. There are many message-passing schemes for resolving such conflicts, e.g., loopy belief propagation (Ihler, Fisher, & Willsky, 2006), expectation propagation (Minka, 2001), and dual decomposition (Wolsey, 1998; Sontag, Globerson, & Jaakkola, 2010). We choose dual decomposition for two reasons: (1) it has stable convergence behavior, and (2) it can provide dual certificates (i.e., lower or upper bounds to inference quality) and thereby stopping conditions for inference.

To summarize, we address the following challenges in designing, implementing, and deploying ELEMENTARY:

- To design a KBC system that is able to take advantage of diverse data resources and best-of-breed algorithms, we address the following challenges:

  **Challenge 1a**: How do we handle imperfect and inconsistent information?
  **Solution**: ELEMENTARY employs the probabilistic logic language Markov logic.

  **Challenge 1b**: How do we handle the diversity of information sources?
  **Solution**: ELEMENTARY uses a two-phase architecture with feature extraction and statistical inference.

  **Challenge 1c**: How do we debug and tune a KBC system?
  **Solution**: ELEMENTARY provides tools to facilitate manual error analysis.

- To deploy ELEMENTARY to large-scale KBC tasks, we address the following scalability and efficiency challenges:

  **Challenge 2a**: How do we scale feature extraction?
  **Solution**: ELEMENTARY leverages mature infrastructures such as Hadoop and Condor.

  **Challenge 2b**: How do we scale statistical inference in Markov logic?
  **Solution**: ELEMENTARY integrates specialized algorithms for subtasks in KBC.

***Evaluations.*** We perform two types of evaluations. First, to evaluate the efficiency and scalability of ELEMENTARY's decomposition-based approach to MLN inference, we compare ELEMENTARY with state-of-the-art systems for MLN inference. We experimentally show that ELEMENTARY's inference approach has higher scalability and efficiency than stat-of-the-art MLN inference approaches. Second, a fundamental rationale of ELEMENTARY is to support combination of diverse data resources and proven KBC techniques. Our hypothesis is that more data and algorithms help improve KBC quality. We validate this hypothesis by measuring changes in KBC quality as we alter the input to ELEMENTARY on several KBC tasks.

***Deployment.*** As a proof of concept, we used ELEMENTARY to implement a solution for the TAC-KBP challenge[2] and obtained result quality that is comparable to the state of the art (Ji, Grishman, Dang, Griffitt, & Ellis, 2010): we achieve a F1 score of 0.80 for entity linking, and a F1 score of 0.31 for slot filling. Furthermore, ELEMENTARY scales the TAC-KBP task to a 500M-webpage corpus with

[1] `http://hadoop.apache.org/`

[2] `http://nlp.cs.qc.cuny.edu/kbp/2010/`. TAC means Text Analysis Conference; KBP means Knowledge-base Population.

2TB of plain English text; from 7B entity mentions and 20B mention pairs, ELEMENTARY produced 114M high-probability relationship mentions within 3 days. In addition, we have deployed ELEMENTARY to an online demonstration system that continuously enriches Wikipedia with facts (and provenance) extracted from millions of webpages crawled on a daily basis.[3]

## Background

KBC is a challenging research topic that researchers have attacked with diverse data resources and many different techniques. We briefly survey some common types of data resources and a few representative techniques that combine them in different ways. Our goal is not to be exhaustive – for more comprehensive surveys, please refer to, e.g., Sarawagi (2008) and Weikum and Theobald (2010) – but to illustrate the diversity of signals used for KBC. We then illustrate Markov logic with an example, and briefly describe dual decomposition, a classic optimization technique that underlies ELEMENTARY's inference approach.

### Data Resources for KBC

We briefly recall two common types of data resources that are valuable for achieving high quality in KBC: mention-level data that deal with the structure in text, and entity-level data that deal with the structure in the target knowledge base (e.g., typing of relation arguments and constraints among relations). The distinction between mentions and entities is illustrated in Figure 2.

***Mention-level Data.*** *Mention-level data* are textual data with *mentions* of entities, relationships, events, etc., or linguistic resources modeling a (natural) language itself. Mentions in a natural language are the raw data from which KBC data is primarily extracted. They provide examples of the variety of ways humans, writing for other humans, encode knowledge. Mention-level data are usually primary input to a KBC task; they are often in the form of raw text corpora and sometimes also include annotated text. One could use annotated text to train statistical information extraction models, e.g., CRFs (Lafferty et al., 2001). There are also unsupervised methods to make use of less structured mention-level data: a natural-language text corpus could be used to build statistical language models (Lin & Pantel, 2001); one could derive from anchor texts (of Wikipedia pages or general webpages) how frequently an expression is used to refer to an entity and use this information to help disambiguate entity mentions (Hoffart et al., 2011).

***Entity-level Data.*** *Entity-level data* are relational data over the domain of conceptual entities (as opposed to language-dependent mentions). They include what are usually called knowledge bases, ontologies, and gazetteers, e.g., YAGO (Nakashole et al., 2011) and Freebase[4]. On the one hand, entity-level data are typical output of KBC systems. On the other hand, existing entity-level data resources are also valuable input for building KBC systems. For example, one could use comprehensive ontologies like YAGO to extract mentions of entities such as people and organizations with pattern matching. One could also use an existing knowledge base to type-check extraction results (Suchanek et al., 2009); e.g., the second argument of the relation `EmployedBy` must be an organization. *Entity-level models* refer to (first-order) logical statements that encode common-sense (or domain-expertise) constraints and rules over relations of interest. For example, rules like "marriage is symmetric" and "a person's birth date is unique" intuitively would help improve biographical relation extraction. Indeed, such high-level constraints have been shown to have significant impacts on KBC quality (Nakashole et al., 2011).

### Common Techniques for KBC

We discuss several common techniques to make use of the above data resources for KBC: classical rule-based approaches, classical statistical approaches, distant supervision, and human feedback.

***Classical Rule-based Approaches.*** Classical rule-based approaches to KBC – e.g., DBLife (DeRose et al., 2007; Shen, Doan, Naughton, & Ramakrishnan, 2007) and SystemT (Chiticariu et al., 2010; Michelakis, Krishnamurthy, Haas, & Vaithyanathan, 2009) – build on the observation that, for many KBC tasks, there are often a small number of rules that can achieve both high precision and high recall. For example, to identify mentions of people in the database research community, DBLife performs string matching between the corpus and heuristic variations of a dictionary of canonical person names (e.g., abbreviations and first/last name ordering). Because these heuristics cover most commonly used variation types, and because person-name ambiguities are rare, DBLife is able to achieve both high recall and high precision in this subtask. The development process of rule-based KBC systems is increasingly being assisted by statistical techniques (Arasu & Garcia-Molina, 2003; Mooney, 1999; Michelakis et al., 2009).

***Statistical Machine Learning Approaches.*** Statistical machine learning approaches (Lafferty et al., 2001; F. Chen, Feng, Christopher, & Wang, 2012; Mintz, Bills, Snow, & Jurafsky, 2009), on the other hand, target KBC tasks that cannot be reasonably covered by a small set of deterministic rules. For example, to extract `HasSpouse` relationships from English text, one would be hard pressed to enumerate a set of possible expressions with high precision and high recall. To address this issue, classical statistical approaches employ

[3] `http://research.cs.wisc.edu/hazy/deepdive`

[4] `http://freebase.com`

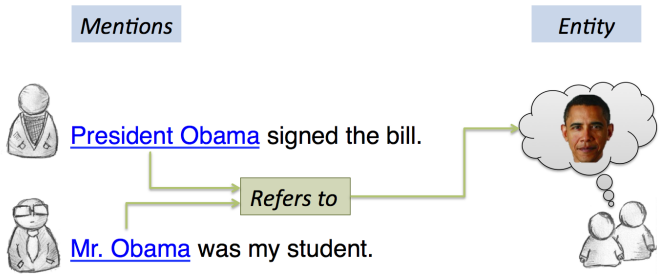


Figure 2. *Illustrating mentions in text of the entity Barack Obama.*

machine learning models such as logistic regression and conditional random fields (Lafferty et al., 2001; Sutton & McCallum, 2006) to learn model parameters from training examples, e.g., annotations of sentences that mention a `HasSpouse` relationship. A trend in classical statistical KBC approaches is to incorporate high-level knowledge (rules) such as "similar text spans are more likely to have the same label" (Sutton & McCallum, 2004; Finkel, Grenager, & Manning, 2005).

***Distant Supervision.*** High-quality annotation data are usually scarce and expensive to obtain. *Distant supervision* is an increasingly popular techique to use entity-level data to generate (silver-standard) annotations from mention-level data (Hearst, 1992; Brin, 1999; Mintz et al., 2009; Zhang et al., 2012). The idea is to use the facts in an existing knowledge base to heuristically identify fact mentions in a text corpus via entity-mention mapping. For example, to populate a relation about people's birth places, one may start with a small set of tuples such as `BirthPlace`(Obama, Hawaii). Then one could consider sentences mentioning both "Obama" and "Hawaii" (e.g., "Obama was born in Hawaii") as positive examples. Textual patterns like "was born in" would then emerge as being indicative for the `BirthPlace` relation and be used to extract more facts. The significance of this technique is that we can get large amounts of training data without human input. While distant supervision often generates noisy examples, the hope is that machine learning techniques – e.g., $\ell_1$-regularization (Zhu et al., 2009) – help reduce such noisiness and select truly indicative patterns. Furthermore, as demonstrated by StatSnowball (Zhu et al., 2009), one may integrate a classical machine-learning model trained via distant supervision into an MLN program.

***Human Feedback.*** Developing KBC systems should be a continuous process – a KBC system is built and deployed, and then iteratively improved as more and more human feedback is integrated. Human feedback may come from different sources, e.g., developers who can spot and correct systematic errors, end-users who can provide low-level (e.g., sentence-level) feedback, and workers on crowdsourcing platforms who can generate large volumes of annotations. We can directly incorporate such human feedback to improve KBC result quality. Furthermore, we may use human feedback as additional training data (e.g., on top of distant-supervision training data) to improve the KBC system itself.

### Markov Logic Networks

To illustrate how MLNs can be used for knowledge-base construction, we walk through a program that extracts affiliations between people and organizations from Web text. In KBC, a typical first step is to use standard NLP toolkits to generate raw data such as plausible mentions of people and organizations in a Web corpus and their co-occurrences. But transforming such raw signals into high-quality and semantically coherent knowledge bases is a challenging task. For example, a major challenge is that a single real-world entity may be referred to in many different ways, e.g., *"UCB"* and *"UC-Berkeley"*. To address such challenges, MLNs provide a framework where we can express logical assertions that are only likely to be true (and quantify such likelihood). Below we explain the key concepts in this framework by walking through an example.

The inference engine of Elementary takes as input a standard MLN program, performs statistical inference, and outputs its results into one or more relations that are stored in a relational database (PostgreSQL). An MLN program consists of three parts: *schema*, *evidence*, and *rules*. To tell Elementary what data will be provided or generated, the user provides a *schema*. Some relations are standard database relations, and we call these relations *evidence*. Intuitively, evidence relations contain tuples that we assume are correct. In the schema of Figure 3, the first eight relations are evidence relations. For example, we know that 'Ullman' and 'Stanford Univ.' co-occur in some webpage, and that 'Doc201' is the homepage of 'Joe'. Other evidence includes string similarity information. In addition to evidence relations, there are also relations whose content we do not know, but we want the MLN program to predict; they are called *query relations*. In Figure 3, `affil` is a query relation since we want the MLN to predict affiliation relationships between persons and organizations. The other two query relations are `pCoref` and `oCoref`, for person and

| Schema | Evidence |
|---|---|
| pSimHard(per1, per2) | coOccurs('Ullman', 'Stanford Univ.') |
| pSimSoft(per1, per2) | coOccurs('Jeff Ullman', 'Stanford') |
| oSimHard(org1, org2) | coOccurs('Gray', 'San Jose Lab') |
| pSimSoft(org1, org2) | coOccurs('J. Gray', 'IBM San Jose') |
| coOccurs(per, org) | coOccurs('Mike', 'UC-Berkeley') |
| homepage(per, page) | coOccurs('Mike', 'UCB') |
| oMention(page, org) | coOccurs('Joe', 'UCB') |
| faculty(org, per) | faculty('MIT', 'Chomsky') |
| *affil(per, org) | homepage('Joe', 'Doc201') |
| *oCoref(org1, org2) | oMention('Doc201', 'IBM') |
| *pCoref(per1, per2) | ... |

**Rules**

| weight | rule | |
|---|---|---|
| $+\infty$ | $\texttt{pCoref}(p, p)$ | $(F_1)$ |
| $+\infty$ | $\texttt{pCoref}(p1, p2) => \texttt{pCoref}(p2, p1)$ | $(F_2)$ |
| $+\infty$ | $\texttt{pCoref}(x, y), \texttt{pCoref}(y, z) => \texttt{pCoref}(x, z)$ | $(F_3)$ |
| 6 | $\texttt{pSimHard}(p1, p2) => \texttt{pCoref}(p1, p2)$ | $(F_4)$ |
| 2 | $\texttt{pSimSoft}(p1, p2) => \texttt{pCoref}(p1, p2)$ | $(F_5)$ |
| $+\infty$ | $\texttt{faculty}(o, p) => \texttt{affil}(p, o)$ | $(F_6)$ |
| 8 | $\texttt{homepage}(p, d), \texttt{oMention}(d, o) => \texttt{affil}(p, o)$ | $(F_7)$ |
| 3 | $\texttt{coOccurs}(p, o1), \texttt{oCoref}(o1, o2) => \texttt{affil}(p, o2)$ | $(F_8)$ |
| 4 | $\texttt{coOccurs}(p1, o), \texttt{pCoref}(p1, p2) => \texttt{affil}(p2, o)$ | $(F_9)$ |
| ... | | |

Figure 3. *An example* MLN *program that performs three tasks jointly: 1. discover affiliation relationships between people and organizations (*affil*); 2. resolve coreference among people mentions (*pCoref*); and 3. resolve coreference among organization mentions (*oCoref*). The remaining eight relations are evidence relations. In particular,* coOccurs *stores person-organization co-occurrences; *Sim* relations are string similarities.*

organization coreference, respectively.

In addition to schema and evidence, ELEMENTARY also requires a set of MLN rules that encode our knowledge about the correlations and constraints over the relations. An MLN rule is a first-order logic formula associated with a real number (or infinity) called a *weight*. Infinite-weighted rules are called hard rules, which means that they must hold in any prediction that the MLN system makes. In contrast, rules with finite weights are soft rules: a positive weight indicates confidence in the rule's correctness.[5]

**Example 1** An important type of hard rule is a standard SQL query, e.g., to transform the results for use in the application. A more sophisticated example of a hard rule is to encode that coreference has the transitive property, which is captured by the hard rule $F_3$. Rules $F_8$ and $F_9$ use person-organization co-occurrences (coOccurs) together with coreference (pCoref and oCoref) to deduce affiliation relationships (affil). These rules are soft since co-occurrence does not necessarily imply affiliation.

Intuitively, when a soft rule is violated, we pay a *cost* equal to the absolute value of its weight (described below). For example, if coOccurs('Ullman', 'Stanford Univ.') and pCoref('Ullman', 'Jeff Ullman'), but not affil('Jeff Ullman', 'Stanford Univ.'), then we pay a cost of 4 because of $F_9$. The goal of an MLN inference algorithm is to find a global prediction that minimizes the sum of such costs.

Similarly, affiliation relationships can be used to deduce non-obvious coreferences. For instance, using the fact that 'Mike' is affiliated with both 'UC-Berkeley' and 'UCB',

[5]Roughly speaking, these weights correspond to the log odds of the probability that the statement is true. (The log odds of probability $p$ is $\log \frac{p}{1-p}$.) In general, these weights do not have a simple probabilistic interpretation (Richardson & Domingos, 2006).

ELEMENTARY may infer that 'UC-Berkeley' and 'UCB' refer to the same organization (rules on `oCoref` are omitted from Figure 3). If ELEMENTARY knows that 'Joe' co-occurs with 'UCB', then it is able to infer Joe's affiliation with 'UC-Berkeley'. Such interdependencies between predictions are sometimes called *joint inference*.

***Semantics.*** An MLN program defines a probability distribution over possible worlds. Formally, we first fix a schema $\sigma$ (as in Figure 3) and a domain $D$. Given as input a set of formulae $\bar{F} = F_1, \ldots, F_N$ with weights $w_1, \ldots, w_N$, they define a probability distribution over *possible worlds* as follows. Given a formula $F_k$ with free variables $\bar{x} = (x_1, \cdots, x_m)$, for each $\bar{d} \in D^m$ we create a new formula $g_{\bar{d}}$ called a *ground formula* where $g_{\bar{d}}$ denotes the result of substituting each variable $x_i$ of $F_k$ with $d_i$. We assign the weight $w_k$ to $g_{\bar{d}}$. Denote by $G = (\bar{g}, w)$ the set of all such weighted ground formulae of $\bar{F}$. Essentially, $G$ forms a Markov random field (MRF) (Koller & Friedman, 2009) over a set of Boolean random variables (representing the truth value of each possible *ground tuple*). Let $w$ be a function that maps each ground formula to its assigned weight. Fix an MLN $\bar{F}$, then for any possible world (instance) $I$ we say a ground formula $g$ is *violated* if $w(g) > 0$ and $g$ is false in $I$, or if $w(g) < 0$ and $g$ is true in $I$. We denote the set of ground formulae violated in a world $I$ as $V(I)$. The cost of the world $I$ is

$$\text{cost}_{\text{MLN}}(I) = \sum_{g \in V(I)} |w(g)| \tag{1}$$

Through $\text{cost}_{\text{MLN}}$, an MLN defines a probability distribution over all instances using the exponential family of distributions that are the basis for graphical models (Wainwright & Jordan, 2008):

$$\Pr[I] = Z^{-1} \exp\{-\text{cost}_{\text{MLN}}(I)\}$$

where $Z$ is a normalizing constant.

***Inference.*** There are two main types of inference with MLNs: *MAP (maximum a posterior) inference*, where we want to find a most likely world, i.e., a world with the lowest cost, and *marginal inference*, where we want to compute the marginal probability of each unknown tuple. Both types of inference are intractable, and so existing MLN systems implement generic (search or sampling) algorithms for inference. Although ELEMENTARY supports both types of inference, we focus on MAP inference to simplify the presentation.

***Learning Weights.*** Weight learning takes as input an MLN program with unweighted rules and an instance of all relations (including evidence and queries), and tries to find an assignment of rule weights that maximizes the probability of the given instance. In ELEMENTARY, weights can be set by the user or automatically learned. ELEMENTARY implements state-of-the-art MLN weight learning algorithms (Lowd & Domingos, 2007), but we focus on inference in this article. There are also algorithms for learning rule structures from scratch (Kok & Domingos, 2005); it is future work to implement structure learning inside ELEMENTARY.

**Dual Decomposition**

We illustrate the basic idea of *dual decomposition* with a simple example. Consider the problem of minimizing a real-valued function $f(x_1, x_2, x_3)$. Suppose that $f$ can be written as

$$f(x_1, x_2, x_3) = f_1(x_1, x_2) + f_2(x_2, x_3).$$

Further suppose that we have black boxes to solve $f_1$ and $f_2$ (plus linear terms). To apply these black boxes to minimize $f$ we need to cope with the fact that $f_1$ and $f_2$ share the variable $x_2$. Following dual decomposition, we can rewrite $\min_{x_1,x_2,x_3} f(x_1, x_2, x_3)$ into the form

$$\min_{x_1,x_{21},x_{22},x_3} f_1(x_1, x_{21}) + f_2(x_{22}, x_3) \text{ s.t. } x_{21} = x_{22},$$

where we essentially made two copies of $x_2$ and enforce that they are identical. The significance of such rewriting is that we can apply Lagrangian relaxation to the equality constraint to decompose the formula into two independent pieces. To do this, we introduce a scalar variable $\lambda \in \mathbb{R}$ (called a *Lagrange multiplier*) and define $g(\lambda) =$

$$\min_{x_1,x_{21},x_{22},x_3} f_1(x_1, x_{21}) + f_2(x_{22}, x_3) + \lambda(x_{21} - x_{22}).$$

For any $\lambda$, we have $g(\lambda) \leq \min_{x_1,x_2,x_3} f(x_1, x_2, x_3)$.[6] Thus, the tightest bound is to maximize $g(\lambda)$; the problem $\max_\lambda g(\lambda)$ is a *dual problem* for the problem on $f$. If the optimal solution of this dual problem is feasible (here, $x_{21} = x_{22}$), then the dual optimal solution is also an optimum of the original program (Wolsey, 1998, p. 168).

The key benefit of this relaxation is that, instead of a single problem on $f$, we can now compute $g(\lambda)$ by solving two independent problems (each problem is grouped by parentheses) that may be easier to solve:

$$g(\lambda) = \min_{x_1,x_{21}} (f_1(x_1, x_{21}) + \lambda x_{21}) + \min_{x_{22},x_3} (f_2(x_{22}, x_3) - \lambda x_{22}).$$

To compute $\max_\lambda g(\lambda)$, we can use standard techniques such as *projected subgradient* (Wolsey, 1998, p. 174). Notice that dual decomposition can be used for MLN inference if $x_i$ are truth values of ground tuples and one defines $f$ to be $\text{cost}_{\text{MLN}}(I)$ as in Equation 1.

## Challenge 1: Knowledge-base Construction with Elementary

We describe the conceptual challenges in the ELEMENTARY approach to KBC and how ELEMENTARY addresses them.

[6] One can always take $x_{21} = x_{22} = x_2$ in the minimization, and the value of the two objective functions are equal.

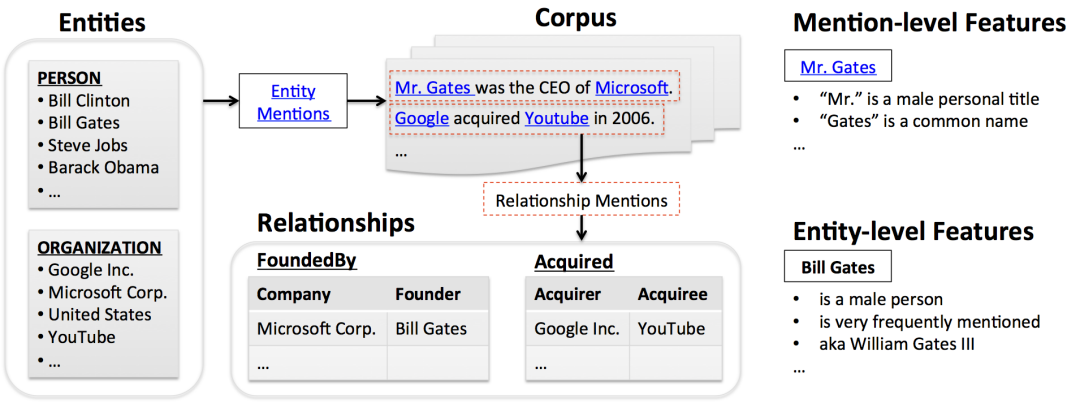


Figure 4. *An illustration of the KBC model in* ELEMENTARY.

### Challenges 1a: Markov Logic in Elementary

A fundamental rationale of ELEMENTARY is to embrace all data resources and algorithms that may contribute to a given KBC task. An immediate challenge to this approach is that we need a way to handle imperfect and inconsistent information from multiple sources. To address this issue in a principled manner, we adopt the popular Markov logic language. In addition to the syntax and semantics described in the previous section, from our three years' experience of building KBC applications with Markov logic, we have found two language extensions to be instrumental to ELEMENTARY: *scoping rules* and *parameterized weights*.

***Scoping Rules.*** By default, the arguments in an MLN predicate are considered to be independent to each other; i.e., for a predicate of arity $a$, there are $|D|^a$ possible ground tuples where $D$ is the domain. Thus, given a query predicate `AdjacentMentions`(mention, mention), a typical MLN system – e.g., ALCHEMY[7] or TUFFY (Niu, Ré, et al., 2011) – would enumerate and perform inference on #mentions$^2$ ground tuples instead of only the #mention$-1$ truly adjacent mention pairs. To address this issue, ELEMENTARY allows a developer to explicitly scope a query predicate with rules of the form

$$\texttt{AdjacentMentions}(m1, m2) :=$$
$$\texttt{Mentions}(m1, p1) \wedge \texttt{Mentions}(m2, p2) \wedge p1 = p2 - 1$$

where the evidence relation `Mentions` lists all mentions and their positional indexes (consecutive integers). ELEMENTARY implements scoping rules by translating them into SQL statements that are used to limit the content of the in-database representation of MLN predicates.

***Parameterized Weights.*** As we will see, an MLN can express classical statistical models such as logistic regression and conditional random fields (Lafferty et al., 2001). Such classical statistical models usually have simple structure but many parameters. To compactly represent these models in MLN, we introduce *parameterized weights*, i.e., weights that are functions of one or more variables in an MLN rule. For example, a logistic regression model on the `TokenLabel` relation may have a parameter for each word-label pair; we can represent such a model using the following MLN rule

$$wgt(w, l) :$$
$$\texttt{Tokens}(tok) \wedge \texttt{HasWord}(tok, w) \Rightarrow \texttt{TokenLabel}(tok, l)$$

where $wgt(w, l)$ is a parameterized weight that depends on the word variable $w$ and the label variable $l$. Besides compactness, a key advantage of parameterized weights is that one can write a program that depends on the input data without actually providing the data. Another benefit is that of efficiency: a rule with parameterized weights may correspond to hundreds of thousands of MLN rules; because there can be non-negligible overhead in processing an MLN rule (e.g., initializing in-memory or in-database data structures), without this compact representation, we have found that the overhead of processing an MLN program can be prohibitive. ELEMENTARY implements parameterized weights by storing functions like $wgt(w, l)$ as look-up tables in a database that can be joined with predicate tables.

### Challenge 1b: Conceptual Model and Architecture for KBC

Markov logic is a generic language for statistical inference. To use it for KBC tasks, we need to set up a development environment, including (1) a conceptual model for KBC that is compatible with MLN, and (2) an architecture that is able to accommodate diverse information sources. We describe them in turn.

***A KBC Model.*** We describe a simple but general model for KBC (see Figure 4). To represent the target KB, we adopt the classic Entity-Relationship (ER) model (P. Chen, 1976; Calì et al., 2010): the schema of the target KB is

[7] `http://alchemy.cs.washington.edu/`

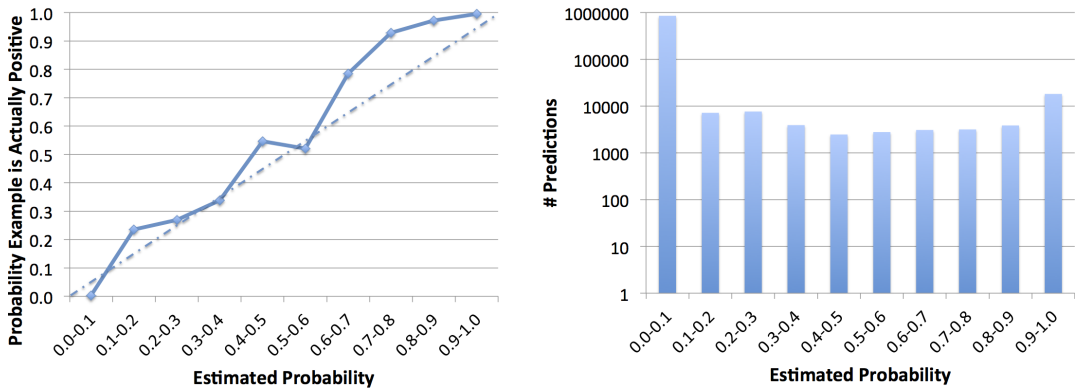


Figure 5. *Probability calibration results of a well-trained relation-extraction model on TAC-KBP. Each prediction has an associated probability. We partition all predictions into ten uniform-width probability bins, and measure the accuracy (left) and frequency (right) of each bin. The dashed line in the left figure indicates the ideal curve where the estimated probabilities exactly match the actual accuracy.*

specified by an ER graph $G = (\bar{E}, \bar{R})$ where $\bar{E}$ is one or more sets of entities, and $\bar{R}$ is one or more relationships. Define $\mathcal{E}(G) = \cup_{E \in \bar{E}} E$, i.e., the set of known entities. To specify a KBC task to Elementary, one provides the schema $G$ and a corpus of documents $D$. Each document $d_i \in D$ consists of a set of possibly overlapping *text spans* (e.g., tokens, phrases, or sentences) $T(d_i)$. Define $\mathcal{T}(D) = \cup_{d_i \in D} T(d_i)$. Our goal is to accurately populate the following tables:

- Entity-mention tables $M_E(E, \mathcal{T}(D))$ for each entity type $E \in \bar{E}$.[8]
- Relationship-mention tables $M_{R_i} \subseteq \mathcal{T}(D)^{k+1}$ for each $R_i \in \bar{R}$, where $k$ is the arity of $R_i$, the first $k$ attributes are entity mentions, and the last attribute is a relationship mention.
- Relationship tables $R_i \in \bar{R}$.

Note that $R_i$ can be derived from $M_E$ and $M_{R_i}$. By the same token, $M_E$ and $M_{R_i}$ provide provenance that connects the KB back to the documents supporting each fact. Intuitively, the goal is to produce an instance $J$ of these tables that is as large as possible (high recall) and as correct as possible (high precision). Elementary populates the target KB based on signals from mention-level structures (over text spans) and entity-level structures (over the target KB). Conceptually, we characterize text spans with a relation $F_m(\mathcal{T}(D), V_m)$ where $V_m$ is the set of possible *features* for each text span. There could be many types of features, e.g., relative position in a document, contained words, matched regular expressions, web search results, etc. Similarly, we introduce entity-level structures with a relation $F_e(\mathcal{E}(G), V_e)$ where $V_e$ is the set of possible feature values for an entity. Examples of $V_e$ include canonical names and known entity attributes (e.g., age, gender, alias, etc.). Furthermore, one may also introduce feature relations that involve multiple mentions (e.g., consecutive mention pairs) or multiple entities (e.g., entity pairs that frequently co-occur). Depending on the data sources and the feature-extraction process, there may be errors or inconsistencies among the feature values. Elementary makes predictions about the target KB via machine learning and statistical inference over these features. A key issue for machine learning is the availability of training data, and an increasingly popular technique addressing this issue is distant supervision. We found that scaling the input data resources is indeed an effective approach to improving the quality of KBC via distant supervision (Zhang et al., 2012).

***Elementary Architecture.*** Markov logic operates on relational data. To accommodate diverse information sources, Elementary employs a two-phase architecture when processing a KBC task: feature extraction and statistical inference (see Figure 1). Intuitively, feature extraction concerns what signals may contribute to KBC and how to generate them from input data resources (e.g., pattern matching and selection), while statistical inference concerns what (deterministic or statistical) correlations and constraints over the signals are valuable to KBC and how to efficiently resolve them. All input data resources are first transformed into relational data via the feature-extraction step. For example, one may employ standard NLP tools to decode the structure (e.g., part-of-speech tags and parse trees) in text, run pattern matching to identify candidate entity mentions, perform topic modeling to provide additional features for documents or text spans, etc. If a KBC technique (e.g., a particular learning or inference algorithm) can be expressed in MLNs, one can also translate them

[8] For simplicity, in this conceptual model we assume that all entities are known, but Elementary supports generating novel entities for the KB as well (e.g., by clustering "dangling" mentions).

into MLN rules. Otherwise, one can execute the technique in the feature-extraction step and "materialize" its result in evidence or MLN rules. Once we have converted all signals into relational evidence, the second step is to perform statistical inference to construct a knowledge base as described in the above KBC model. From our experience of building large-scale knowledge bases and applications on top of them, we have found that it is critical to efficiently process structured queries over large volumes of structured data. Therefore, we have built Elementary on top of a relational database management system (RDBMS).

**Challenge 1c: Debugging and Tuning**

The ability to integrate different data and algorithms for KBC does not directly lead to high-quality results: a developer still needs to decide what signals to use and how to combine them. When training data are available (through either direct supervision or distant supervision), machine learning techniques can help tune system parameters. For example, one can integrate into an MLN trained statistical models such as logistic regression and conditional random fields, or perform weight learning on an entire MLN. In addition to machine learning, we have also found error analysis on the output of a KBC system to be an effective method for debugging and tuning. For example, when developing solutions to the TAC-KBP challenge, we improved the slot-filling F1 score from 0.15 to 0.31 with several iterations of error analysis. In each iteration, we identify errors in samples of Elementary's output, trace them back to input signals through a provenance utility in Elementary's inference engine, classify the errors by causes (e.g., noise in an evidence relation or pathological MLN rules), and then apply appropriate remedies. As we will see, different signals do have different impacts on the result quality for a KBC task. Thus, to facilitate the development process, we have been equipping Elementary with utilities to help a developer debug and fine-tune a KBC system. For example, in a debugging mode, Elementary reports provenance about each prediction it generates – it outputs related MLN rules and grounding that lead to each prediction. In addition, Elementary also performs aggregation on such provenance to help identify spurious data sources or MLN rules. As another example, to evaluate whether a statistical model functions properly, Elementary can output probability calibration results, including the ground-truth accuracy and frequency distribution over different probability levels (see Figure 5).

**Putting It All Together**

The high-level steps in applying Elementary to a KBC task are as follows:

1. Choose the entities and relationships to recognize in text.[9]
2. Choose a corpus of documents, and collect all data resources that may help with the current KBC task.
3. Perform feature extraction on the input data by running standard NLP tools, machine learning algorithms, or custom code to create a relational representation of all input data, i.e., evidence.
4. Create an MLN program that integrates existing machine-learning models, domain-knowledge rules, or other sources of information on the evidence and the target KB. If needed, perform weight learning.
5. Run statistical inference with the MLN program and the evidence to make predictions for the target KB.
6. Inspect results and if appropriate, go to (3) or (4).

To demonstrate the flexibility of Elementary for KBC, we use several examples to illustrate how Elementary integrates diverse data resources and different techniques for KBC. Recall that there are two phases in Elementary: feature extraction and statistical inference. When integrating an existing KBC technique, one needs to decide what goes into feature extraction and what goes into statistical inference. To illustrate, we consider several common techniques for entity linking, relation extraction, and incorporating domain knowledge, respectively.

***Entity Linking.*** Entity linking is the task of mapping text spans to entities, i.e., populating the $M_E(E, \mathcal{T}(D))$ tables in the formal model. To identify candidate mentions of entities, there are several common techniques, e.g., (1) perform string matching against dictionaries; (2) use regular expressions or trigger words (e.g., "Prof." for person mentions); (3) run named-entity-recognition tools. All these techniques can be implemented in the feature-extraction phase of Elementary. Sometimes a mention may correspond to multiple candidate entities. To determine which entity is correct, one could use various techniques and data resources. For example, a heuristic is that "if the string of a mention is identical to the canonical name of an entity, then this mention is likely to refer to this entity"; one can express this heuristic in a (soft or hard) MLN rule:

$$\begin{aligned} &\text{MentionText}(\textit{mention}, \textit{string}) \\ \wedge\ &\text{EntityName}(\textit{entity}, \textit{string}) \\ \Rightarrow\ &M_E(\textit{entity}, \textit{mention}). \end{aligned}$$

When a named-entity-recognition (NER) tool is used in feature extraction, one can use the mention types output by

[9] We leave open information extraction (Wu & Weld, 2010) as future work.

the tool as a constraint for entity linking. For example, let NerMentionType(*mention*, *type*) be an evidence relation storing NER output; then we can add the constraint

$$M_{\texttt{PERSON}}(\mathit{entity}, \mathit{mention})$$
$$\Rightarrow \text{NerMentionType}(\mathit{mention}, \texttt{PERSON}).$$

As another example, suppose there is an evidence relation Anchor(*string*, *entity*, $freq$) indicating how frequently (measured by the numeric attribute $freq$) an anchor text *string* links to the Wikipedia page representing entity *entity*. Then one could use these signals for entity disambiguity with the rule

$$wgt(freq) : \text{Anchor}(\mathit{string}, \mathit{entity}, freq)$$
$$\wedge \text{MentionText}(\mathit{mention}, \mathit{string})$$
$$\Rightarrow M_E(\mathit{entity}, \mathit{mention}).$$

where the $wgt(freq)$ syntax means that the rule weight is a function of the $freq$ attribute of Anchor. Note that one can also adapt Anchor for other data resources, e.g., web search results with mention phrases as queries and Wikipedia links as proxies for entities. Another common technique for entity linking is coreference. For example, let SamePerson($m1, m2$) be an equivalence relation indicating which pairs of person mentions are coreferent, i.e., referring to the same entity. Then one can use the following heuristics to propagate entity linking results between coreferent mentions[10]:

$$\text{MentionText}(m1, s) \wedge \text{MentionText}(m2, s)$$
$$\wedge \text{InSameDoc}(m1, m2) \wedge \text{FullName}(s)$$
$$\Rightarrow \text{SamePerson}(m1, m2).$$
$$\text{MentionText}(m1, s1) \wedge \text{MentionText}(m2, s2)$$
$$\wedge \text{InSamePara}(m1, m2) \wedge \text{ShortFor}(s1, s2)$$
$$\Rightarrow \text{SamePerson}(m1, m2).$$
$$M_{\texttt{PERSON}}(e, m1) \wedge \text{SamePerson}(m1, m2)$$
$$\Rightarrow M_{\texttt{PERSON}}(e, m2).$$

***Relation Extraction.*** Relation extraction is the task of determining whether a relationship is mentioned in a text span, i.e., populating the relationship mention tables $M_{R_i}$. For natural-language text, the most common approach is to perform classification based on linguistic patterns. Researchers have found many different kinds of linguistic patterns to be helpful; e.g., shallow features such as word sequences and part-of-speech tags between entity mentions (Carlson et al., 2010; Zhu et al., 2009), deep linguistic features such as dependency paths (Lin & Pantel, 2001; Wu & Weld, 2010; Mintz et al., 2009), and other features such as the n-gram-itemset (Nakashole et al., 2011). To improve coverage and reduce noisiness of these patterns, researchers have also invented different feature selection and expansion techniques; e.g., frequency-based filtering (Wu & Weld, 2010; Nakashole et al., 2011), feature selection with $\ell_1$-norm regularization (Zhu et al., 2009), and feature expansion based on pattern similarity (Lin & Pantel, 2001; Nakashole et al., 2011). In Elementary, one can implement different combinations of the above techniques in the feature extraction phase, and incorporate the signals with (soft or hard) rules of the form

$$\text{WordSequence}(s, m1, m2, \text{“}\texttt{was born in}\text{”})$$
$$\Rightarrow M_{\texttt{BirthPlace}}(m1, m2, s).$$

where $s$ is a sentence and $m1$ and $m2$ are two entity mentions (i.e., text spans) within $s$.

To learn the association between linguistic patterns and relationships, there are two common approaches: *direct supervision* and *distant supervision* (Mintz et al., 2009). In direct supervision, one uses mention-level annotations of relationship mentions as training data to learn a statistical model between patterns and relationships (Lafferty et al., 2001). As mention-level annotations are usually rare and expensive to obtain, an increasingly popular approach is distant supervision, where one uses entity-level relationships and entity linking to heuristically collect silver-standard annotations from a text corpus (Mintz et al., 2009; Zhu et al., 2009; Nakashole et al., 2011; Carlson et al., 2010). To illustrate, let KnownBirthPlace(person, location) be an existing knowledge base containing tuples like KnownBirthPlace(`Barack_Obama`, `Hawaii`), then one can perform distant supervision in Elementary by learning weights for MLN rules of the form

$$wgt(pat) : \text{WordSequence}(s, m1, m2, pat)$$
$$\wedge M_{\texttt{PERSON}}(per, m1)$$
$$\wedge M_{\texttt{LOCATION}}(loc, m2)$$
$$\Rightarrow \text{KnownBirthPlace}(per, loc).$$

where $w(pat)$ is a parameterized weight that depends on the pattern $pat$; intuitively it models how indicative this pattern is for the BirthPlace relationship. For example, Elementary may learn that “was born in” is indicative of BirthPlace based on the tuple KnownBirthPlace(`Barack_Obama`, `Hawaii`), the sentence $s$ =“Obama was born in Hawaii,” and entity linking results such as $M_{\texttt{PERSON}}$(`Barack_Obama`, `Obama`).

***Domain Knowledge.*** Several recent projects (Zhu et al., 2009; Nakashole et al., 2011; Carlson et al., 2010) show that entity-level resources can effectively improve the precision of KBC. For example, Prospera (Nakashole et al., 2011) found that validating the argument types of relationships can improve the quality of relation extraction. One can incorporate such constraints with rules like

$$\infty : M_{\texttt{BirthPlace}}(m1, m2, s) \Rightarrow \exists\, e1 \;\; M_{\texttt{PERSON}}(e1, m1)$$

[10] The weights are not shown for clarity.

which enforces that the first argument of a BirthPlace mention must be a person entity. Besides typing, one can also specify semantic constraints on relationships such as "a person can have at most one birth place" and "HasChild and HasParent are symmetric":

$$\infty : \quad \text{BirthPlace}(p, l1) \wedge l1 \neq l2 \Rightarrow \text{BirthPlace}(p, l2).$$

$$\infty : \quad \text{HasChild}(p1, p2) \Leftrightarrow \text{HasParent}(p2, p1).$$

Another type of domain knowledge involves corrections to systematic errors in the statistical extraction models. For example, from sentences like "Obama lived in Hawaii for seven years," distant supervision may erroneously conclude that the linguistic pattern "lived in" is strongly indicative of BirthPlace. Once a developer identifies such an error (say via debugging), one could correct this error by simply adding the rule

$$\infty : \text{WordSequence}(s, m1, m2, \text{“}\texttt{lived in}\text{”}) \Rightarrow \neg M_{\texttt{BirthPlace}}(m1, m2, s).$$

Note that this rule only excludes mention-level relationships. Thus, if there are sentences "$X$ lived in $Y$" and "$X$ was born in $Y$", the final (entity-level) KB may still contain the tuple $\texttt{BirthPlace}(e_X, e_Y)$ based on the second sentence, where $e_X$ (resp. $e_Y$) denotes the entity referred to by $X$ (resp. $Y$).

### Challenge 2: Scalability of Elementary

We discuss the scalability challenges in Elementary. We first briefly describe how we scale feature extraction with high-throughput parallel computing infrastructure such as Hadoop[11] and Condor (Thain et al., 2005). We then explain how to scale up MLN inference by integrating specialized algorithms with dual decomposition.

#### Challenge 2a: Scaling Feature Extraction

To scale the Elementary architecture to Web-scale KBC tasks, we employ high-throughput parallel computing frameworks such as Hadoop and Condor. We use the Hadoop File System for storage and MapReduce (Dean & Ghemawat, 2004) for feature extraction. However, when trying to deploy Elementary to Web-scale KBC tasks that involve terabytes of data and deep natural-language-processing features, we found a 100-node Hadoop MapReduce cluster to be insufficient. The reasons are two-fold: (1) Hadoop's all-or-nothing approach to failure handling hinders throughput, and (2) the number of cluster machines for Hadoop is limited. Fortunately, the Condor infrastructure supports a best-effort failure model, i.e., a job may finish successfully even when Condor fails to process a small portion of the input data. Moreover, Condor can schedule jobs on both cluster machines and idle workstations. This allows us to simultaneously leverage thousands of machines from across a department, an entire campus, or even the nation-wide Open Science Grid.[12] For example, using Condor, we were able to utilize hundreds of thousands of machine-hours to perform deep linguistic feature extraction (including named entity recognition and dependency parsing) on the 500M-doc ClueWeb09 corpus[13] within a week. For statistical inference, we use the parallel RDBMS from Greenplum, Inc.

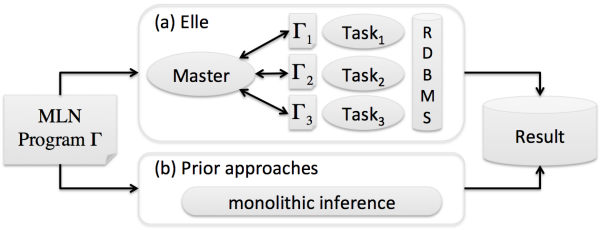


Figure 6. Elementary *breaks a Markov logic program,* $\Gamma$*, into several, smaller subtasks (shown in Panel a), while prior approaches are monolithic (shown in Panel b).*

#### Challenge 2b: (Part 1) Scaling MLN Inference with Specialized Tasks

We observe that a KBC task usually involves routine subtasks such as classification and coreference resolution. Moreover, such subtasks may correspond to a subset of rules in an MLN program. Thus, instead of running a generic MLN inference algorithm on the entire program (as done by state-of-the-art MLN inference algorithms, see Figure 6), we can partition the rule set into subtasks and invoke specialized algorithms for corresponding subtasks (Niu, Zhang, et al., 2011). In this section, we describe several such tasks that are common in KBC tasks (see Table 1). For each task, we describe how it arises in an MLN, a specialized algorithm, and an informal argument of why being aware of the special structure may outperform generic inference approaches.

***Classification.*** Classification is a fundamental inference problem and ubiquitous in applications. Classification arises in Markov logic as a query predicate $R(x, y)$ with hard rules of the form

$$R(x, y_1) \wedge y_1 \neq y_2 \Rightarrow \neg R(x, y_2),$$

which mandates that each object (represented by a possible value of $x$) can only be assigned at most one label (represented by a possible value of $y$). If the only query relation in a subprogram $\Gamma_i$ is $R$ and $R$ is mentioned at most once in each rule in $\Gamma_i$ (except the rule above), then

[11] http://hadoop.apache.org/

[12] http://www.opensciencegrid.org

[13] http://lemurproject.org/clueweb09.php/

Table 1. *Example specialized tasks and their implementations in* ELEMENTARY.

| Task | Implementation |
|---|---|
| Classification | Logistic Regression (Boyd & Vandenberghe, 2004) |
| Segmentation | Conditional Random Fields (Lafferty et al., 2001) |
| Coreference | Correlation Clustering (Ailon, Charikar, & Newman, 2008) |

$\Gamma_i$ is essentially a (multi-class) *logistic regression* (LR) model. The inference problem for LR is trivial given model parameters (here rule weights) and feature values (here ground formulae). For example, logistic regression models can be used for entity linking – one can view all candidate entities as classification labels. Below are some sample rules for person entity linking:

$$\begin{aligned}
&\infty : M_{\texttt{PERSON}}(entity1, mention) \wedge entity1 \neq entity2 \\
&\Rightarrow \neg M_{\texttt{PERSON}}(entity2, mention). \\
&2 : \text{MentionString}(mention, string) \\
&\quad \wedge \text{EntityString}(entity, string) \\
&\Rightarrow M_{\texttt{PERSON}}(entity, mention). \\
&5 : \text{MentionString}(m, str) \\
&\quad \wedge \text{StringContains}(str, \text{“}\texttt{Mr.}\text{”}) \\
&\quad \wedge \text{IsFemale}(ent) \\
&\Rightarrow \neg M_{\texttt{PERSON}}(ent, m).
\end{aligned}$$

On the other hand, prior inference approaches would operate on the MRF (i.e., ground network) generated from the input MLN. Suppose there are $N$ objects and $K$ labels. Then it would require $N\binom{K}{2}$ factors to represent the above unique-label rule in an MRF. For tasks such as entity linking (e.g., mapping textual mentions to Wikipedia entities), the value of $K$ could be in the millions.

***Segmentation.*** Segmentation is the task of partitioning sequence data (e.g., word sequences) while classifying each partition with a set of labels. Examples include segmenting an English sentence into different types of phrases (noun phrases, verb phrases, etc.) and segmenting a citation line into different fields (authors, title, etc.). A popular approach to segmentation is linear-chain conditional random fields (CRFs) (Lafferty et al., 2001), a machine-learning model that can be solved efficiently with dynamic programming algorithms. A linear-chain CRF model consists of unigram features and bigram features. In Markov logic, unigram features have the same form as LR rules, while bigram features are soft rules of the form

$$wgt(y_1, y_2): \quad L(x_1, x_2) \Rightarrow R(x_1, y_1) \wedge R(x_2, y_2),$$

where $R$ is a classification predicate and $L$ is an evidence relation containing correlated object pairs (e.g., consecutive tokens in a sentence). For example, when segmenting sentences into phrases, $R$ may be TokenInPhrase(token, phrase) and $L$ may be FollowedBy(token1, token2), where the argument "phrase" takes values from `NounPhrase`, `VerbPhrase`, etc. If the correlation structure $L$ represents chains over the objects (i.e., $x$ values), then we can solve a subproblem with unigram and bigram rules like these with the Viterbi algorithm (Lafferty et al., 2001).

***Coreference.*** Another common task is coreference resolution (coref), e.g., given a set of strings (say phrases in a document) we want to decide which strings represent the same real-world entity. This arises in MLNs as a query relation $R$ that is subject to hard rules encoding an equivalent relation, including the reflexivity, symmetry, and transitivity properties. The input to a coreference task is a single relation $B(o1, o2, wgt)$ where $wgt = \beta_{o1,o2} \in \mathbb{R}$ indicates how likely the objects $o1, o2$ are coreferent (with 0 being neutral). In ELEMENTARY, $B$ is indirectly specified with rules of the form

$$1.5 : \text{Similar}(o1, o2) \Rightarrow R(o1, o2)$$

where Similar is an evidence relation but, in general, the left hand side can be a subformula involving any number of evidence relations. The above rule essentially adds all tuples in Similar to $B$ with a weight 1.5. The output of a coreference task is an instance of relation $R(o1, o2)$ that indicates which pairs of objects are coreferent. Assuming that $\beta_{o1,o2} = 0$ if $(o1, o2)$ is not covered by the relation $B$, then each valid $R$ incurs a cost (called *disagreement cost*)

$$\text{cost}_{\text{coref}}(R) = \sum_{\substack{o1,o2:(o1,o2)\notin R \\ \text{and } \beta_{o1,o2}>0}} |\beta_{o1,o2}| + \sum_{\substack{o1,o2:(o1,o2)\in R \\ \text{and } \beta_{o1,o2}<0}} |\beta_{o1,o2}|.$$

In Figure 3, $F_1$ through $F_5$ can be mapped to a coreference task for the relation `pCoref`. $F_1$ through $F_3$ encode the reflexivity, symmetry, and transitivity properties of `pCoref`, and the ground formulae of $\texttt{F}_4$ and $\texttt{F}_5$ specify the weighted-edge relation $B$. The goal is to find a relation $R$ that achieves the minimum cost.

Coreference is a well-studied problem (Fellegi & Sunter, 1969), and there are approximate inference techniques for coreference resolution (Ailon et al., 2008; Arasu, Ré, & Suciu, 2009; Singh, Subramanya, Pereira, & McCallum, 2011). ELEMENTARY implements both *correlation clustering algorithms* (Ailon et al., 2008) and the sampling algorithm of Singh et al. (2011). In contrast, explicitly representing a coreference task in an MRF may be inefficient: a direct implementation of transitivity requires $N^3$ factors where $N$ is the number of objects.

***Task Detection.*** In our prototype Elementary system, we allow a user to manually decompose an MLN program and specify a specific inference algorithm to be run on each subprogram. We have also implemented pattern-matching heuristics to detect structures in an MLN program and instantiate the algorithms described above (Niu, Zhang, et al., 2011). Our heuristics successfully detected the decomposition schemes in our experiments. If the user does not provide an algorithm for a subprogram and Elementary cannot automatically match this subprogram to any specialized algorithm, we run a generic MLN inference algorithm, namely WalkSAT (Kautz, Selman, & Jiang, 1997).[14]

**Challenge 2b: (Part 2) Resolving Inconsistencies with Dual Decomposition**

Although program decomposition allows Elementary to integrate specialized algorithms for MLN inference, a crucial challenge is that, because a query relation may be shared by multiple tasks, there may be inconsistencies between predictions from different tasks. There are several general inference schemes to address this issue, e.g., (loopy) belief propagation (Ihler et al., 2006), expectation propagation (Minka, 2001), and dual decomposition (Wolsey, 1998; Sontag et al., 2010). We choose dual decomposition for the following reasons: (1) dual decomposition's convergence behavior has theoretical guarantees, whereas belief propagation and expectation propagation do not; and (2) dual decomposition provides the possibility of dual certificates (i.e., lower or upper bounds to inference quality) and thereby stopping conditions for inference.[15] To perform dual decomposition for MLNs, Elementary takes as input the (first-order) rule set of an MLN program, and partitions it into multiple tasks each of which can be solved with different algorithms. One design decision that we make in Elementary is that entire relations are shared or not – as opposed to individual ground tuples. This choice allows Elementary to use an RDBMS for all data movement, which can be formulated as SQL queries; in turn this allows us to automatically scale low-level issues such as memory management. Dual decomposition introduces auxiliary variables called Lagrange multipliers; they are used to pass messages between different tasks so as to iteratively reconcile conflicting predictions. We illustrate dual decomposition in Elementary with an example.

**Example 2** Consider a simple MLN $\Gamma$ modeling the impact of news on companies' stock performance:

$$\begin{array}{lll} 1 & \texttt{GoodNews}(p) \Rightarrow \texttt{Bullish}(p) & \phi_1 \\ 1 & \texttt{BadNews}(p) \Rightarrow \texttt{Bearish}(p) & \phi_2 \\ 5 & \texttt{Bullish}(p) \Leftrightarrow \neg\texttt{Bearish}(p) & \phi_3 \end{array}$$

where GoodNews and BadNews are evidence and the other two relations are queries. Consider the decomposition

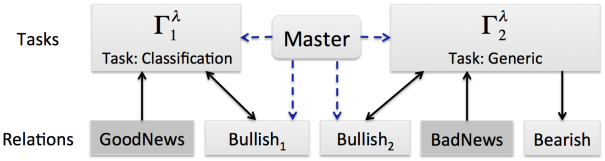


Figure 7. *A program-level decomposition for Example 2. Shaded boxes are evidence relations. Solid arrows indicate data flow; dash arrows are control.*

$\Gamma_1 = \{\phi_1\}$ and $\Gamma_2 = \{\phi_2, \phi_3\}$. $\Gamma_1$ and $\Gamma_2$ share the relation Bullish; so we create two copies of this relation: $\texttt{Bullish}_1$ and $\texttt{Bullish}_2$, one for each subprogram. We introduce Lagrange multipliers $\lambda_p$, one for each possible ground tuple $\texttt{Bullish}(p)$. We thereby obtain a new program $\Gamma^\lambda$:

$$\begin{array}{lll} 1 & \texttt{GoodNews}(p) \Rightarrow \texttt{Bullish}_1(p) & \phi_1' \\ \lambda_p & \texttt{Bullish}_1(p) & \varphi_1 \\ 1 & \texttt{BadNews}(p) \Rightarrow \texttt{Bearish}(p) & \phi_2 \\ 5 & \texttt{Bullish}_2(p) \Leftrightarrow \neg\texttt{Bearish}(p) & \phi_3' \\ -\lambda_p & \texttt{Bullish}_2(p) & \varphi_2 \end{array}$$

where each $\varphi_i$ $(i = 1, 2)$ represents a set of singleton rules, one for each value of $p$ (i.e., for each specific person in a given testbed). This program contains two subprograms, $\Gamma_1^\lambda = \{\phi_1', \varphi_1\}$ and $\Gamma_2^\lambda = \{\phi_2, \phi_3', \varphi_2\}$, that can be solved independently with any MLN inference algorithm.

As illustrated in Figure 7, the output of our decomposition method is a bipartite graph between a set of subprograms and a set of relations. Elementary attaches an inference algorithm to each subprogram; we call this pair of algorithm and subprogram a *task*.

***Message Passing.*** We apply the master-slave message passing scheme (Komodakis, Paragios, & Tziritas, 2007) to coordinate tasks. The master-slave approach alternates between two steps: (1) perform inference on each part independently to obtain each part's predictions on shared variables, and (2) a process called the *Master* examines the (possibly conflicting) predictions and sends messages in the form of Lagrange multipliers to each task.

The Master chooses the values of the Lagrange multipliers via an optimization process (Komodakis et al., 2007). In particular, we can view the Master as optimizing $\max_\lambda g(\lambda)$ using an iterative solver, in our case projected subgradient ascent (Wolsey, 1998, p. 174). Specifically, let $p$ be a tuple of a query relation $r$; in the given decomposition, $p$ may be shared between $k$ subprograms and so there are $k$ copies of

[14] We focus on MAP inference in this work, though Elementary is able to run marginal inference as well.

[15] We leave a more detailed study of dual certificates in Elementary to future work.

$p$ – call them $p_1, \ldots, p_k$. For $i = 1, \ldots, k$, let $p_i^t \in \{0, 1\}$ denote the value of $p$ predicted by task $i$ at step $t$ and $\lambda_i^t$ denote the corresponding Lagrange multiplier. At step $t$, the Master updates $\lambda_i^t$ by comparing the predicted value of $p_i$ by task $i$ to the average value for $p$ output by all inference tasks. This leads to the following update rule:

$$\lambda_i^{t+1} := \lambda_i^t + \alpha_t \left( p_i^t - \frac{|\{j : p_j = 1\}|}{k} \right),$$

where $\alpha_t$ is the step size for this update. For example, one may choose the divergent step sizes $\alpha_t = 1/t$ that have theoretical convergence guarantees (Anstreicher & Wolsey, 2009). The subgradient ascent procedure stops either when all copies have reached an agreement or when ELEMENTARY has run a pre-specified maximum number of iterations.

A novelty of ELEMENTARY is that we can leverage a RDBMS to efficiently compute the subgradient on an entire relation. To see why, let $\lambda_p^i$ be the multipliers for a shared tuple $p$ of a relation $r$; $\lambda_p^i$ is stored as an extra attribute in each copy of $r$. For simplicity, assume that $p$ is shared by two tasks. Note that at each iteration, $\lambda_p^i$ changes only if the two copies of $r$ do not agree on $p$ (i.e., exactly one copy has $p$ missing). Thus, we can update the entire vector $\lambda_r$ with an (outer) join between the two copies of $r$ using SQL.

***Producing a Solution.*** If a shared relation is not subject to any hard rules, ELEMENTARY takes majority votes from the predictions of related tasks. (If all copies of this relation have converged, the votes would be unanimous.) To ensure that hard rules in the input MLN program are not violated in the final output, we insist that for any query relation $r$, all hard rules involving $r$ (if any) be assigned to a single task, and that the final value of $r$ be taken from this task. This guarantees that the final output is a possible world for $\Gamma$ (provided that the hard rules are satisfiable).

## Related Work

There is a trend to build KBC systems with increasingly sophisticated statistical inference. For example, CMU's NELL (Carlson et al., 2010) integrates four different extraction components that implement complementary techniques and consume various data resources (e.g., natural-language text, lists and tables in webpages, and human feedback). MPI's SOFIE/Prospera (Suchanek et al., 2009; Nakashole et al., 2011) combines pattern-based relation extraction approaches with domain-knowledge rules, and performs consistency checking against the existing YAGO knowledge base. Microsoft's StatsSnowball/EntityCube (Zhu et al., 2009) performs iterative distant supervision while using the $\ell_1$-norm regularization technique to reduce noisy extraction patterns; similar to ELEMENTARY, StatsSnowball also incorporates domain knowledge with Markov logic and observed quality benefit in doing so. Finally, behind IBM's DeepQA project's remarkable success at the *Jepardy* Challenge, there is a "massively parallel probabilistic evidence-based architecture" that combines "more than 100 different techniques" (Ferrucci et al., 2010).

Researchers have proposed different approaches to improving MLN inference performance in the context of text applications. In StatSnowball, Zhu et al. (2009) demonstrate high quality results of an MLN-based approach. To address the scalability issue of generic MLN inference, they make additional independence assumptions in their programs. In contrast, the goal of ELEMENTARY is to automatically scale up statistical inference while sticking to MLN semantics. Theobald, Sozio, Suchanek, and Nakashole (2010) design specialized MaxSAT algorithms that efficiently solve MLN programs of special forms. In contrast, we study how to scale general MLN programs. Riedel (2008) proposed a cutting-plane meta-algorithm that iteratively performs grounding and inference, but the underlying grounding and inference procedures are still for generic MLNs. In Tuffy, Niu, Ré, et al. (2011) improve the scalability of MLN inference with an RDBMS, but their system is still a monolithic approach that consists of generic inference procedures. ELEMENTARY specializes to MLNs. There are, however, other statistical-inference frameworks such as PRMs (Friedman, Getoor, Koller, & Pfeffer, 1999), BLOG (Milch et al., 2005), Factorie (McCallum, Schultz, & Singh, 2009; Wick, McCallum, & Miklau, 2010), and PrDB (Sen, Deshpande, & Getoor, 2009). Our hope is that the techniques developed here can be adapted to these frameworks as well.

Dual decomposition is a classic and general technique in optimization that decomposes an objective function into multiple smaller subproblems; in turn these subproblems communicate to optimize a global objective via Lagrange multipliers (Bertsekas, 1999). Recently dual decomposition has been applied to inference in graphical models such as MRFs. In the master-slave scheme (Komodakis et al., 2007; Komodakis & Paragios, 2009), the MAP solution from each subproblem is communicated and the Lagrange multipliers are updated with the projected gradient method at each iteration. Our prototype implementation of ELEMENTARY uses the master-slave scheme. It is future work to adapt the closely related tree-reweighted (TRW) algorithms (Wainwright, Jaakkola, & Willsky, 2005; Kolmogorov, 2006) that decompose an MRF into a convex combination of spanning trees each of which can be solved efficiently. Researchers have also applied dual decomposition in settings besides graphical models. For example, Rush, Sontag, Collins, and Jaakkola (2010) employ dual decomposition to jointly perform tagging and parsing in natural language processing. ELEMENTARY can be viewed as extending this line of work to higher-level tasks (e.g., classification).

Table 2. *Dataset sizes. The first five columns are the number of relations in the input MLN, the number of MLN rules, the number of evidence tuples, and the number of MRF factors after grounding with* Tuffy*, and the size of in-database representation of the MRF. The last four columns indicate the number of tasks of each type in the decomposition used by* Elementary *(see Table 1).*

| | Relations | Rules | Evidence | MRF | DB | LR | CRF | Coref | WalkSAT |
|---|---|---|---|---|---|---|---|---|---|
| **IERJ** | 18 | 357 | 150K | 564K | 44MB | 1 | 0 | 1 | 1 |
| **KBP** | 7 | 6 | 4.3M | 20M | 1.6GB | 2 | 0 | 0 | 1 |
| **KBP+** | 7 | 6 | 240M | 64B | 5.1TB | 2 | 0 | 0 | 1 |

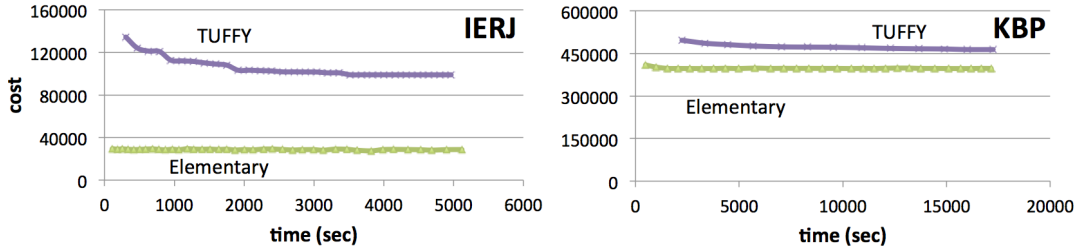


Figure 8. *High-level performance results of* Elementary *and* Tuffy*. For each dataset and each system, we plot a time-cost curve.* Tuffy *was not able to produce feasible solutions, and the corresponding curves were obtained by "softening" hard rules to have weight 100.*

## Empirical Evaluations

We empirically evaluate Elementary by validating the following two main hypotheses: (1) the decomposition-based approach to Markov logic inference enables Elementary to have higher scalability and efficiency than prior inference approaches; and (2) we can significantly improve the quality of KBC tasks by integrating more data resources and techniques with statistical inference inside Elementary. Since scalability and efficiency is a premise for Elementary's ability to produce high-quality KBC results, we present the efficiency experiments first, and then the quality experiments.

### Efficiency Experiments

A main hypothesis is that Elementary's decomposition-based approach to statistical inference in Markov logic outperforms prior inference approaches in terms of efficiency and scalability. To validate this, we compare the performance of Elementary with state-of-the-art MLN systems, namely Tuffy (Niu, Ré, et al., 2011) and Alchemy on two datasets. We show that Elementary indeed has higher efficiency than prior inference approaches.

***Datasets and MLNs.*** We use a publicly available MLN testbed called **IERJ** from Alchemy's website[16], and one MLN that we developed for the TAC-KBP challenge (Table 2): (1) **IERJ**, where one performs joint segmentation and entity resolution on Cora citations; see the "Jnt-Seg-ER" program in (Poon & Domingos, 2007); and (2) **KBP**, which is our implementation of the TAC-KBP (knowledge-base population) challenge of populating a KB with 34 relations from a 1.8M-doc corpus by performing two related tasks: a) *entity linking*: extract all entity mentions and map them to entries in Wikipedia, and b) *slot filling*: determine relationships between entities. We use an MLN that performs entity linking, slot filling, entity-level knowledge-base population, and fact verification from an existing partial knowledge base. In KBP, we use a 116K-doc subset of the TAC-KBP corpus. To test the scalability of Elementary, we also run the KBP program on the full 1.8M-doc TAC-KBP corpus ("**KBP+**").

***Methodology.*** We run Tuffy (Niu, Ré, et al., 2011) and Alchemy as state-of-the-art monolithic MLN inference systems. We implement Elementary on top of our open-source Tuffy system.[17] As Tuffy has similar or superior performance to Alchemy on each dataset, here we use Tuffy as a representative for state-of-the-art MLN inference. In Elementary, we use the sampling algorithm of Singh et al. (2011) for coref tasks. Elementary's decomposition scheme for each dataset can be found in Table 2.

Both Tuffy and Elementary are implemented in Java and use PostgreSQL 9.0.4 as the underlying database system. Unless specified otherwise, all experiments are run on a RHEL 6.1 server with four 2.00GHz Intel Xeon CPUs (40 total physical cores plus hyperthreading) and 256 GB of RAM. Although all three approaches can be parallelized

[16] `http://alchemy.cs.washington.edu/`

[17] `http://research.cs.wisc.edu/hazy/tuffy`

Table 3. *Performance and quality comparison on individual tasks. "Initial" (resp. "Final") is the time when a system produced the first (resp. converged) result. "F1" is the F1 score of the final output. For system-task pairs with infinite cost, we also "soften" hard rules with a weight 100, and report corresponding cost and F1 in parentheses. Each cost/F1 value is an average over five runs.*

| Task | System | Initial | Final | Cost | F1 |
|---|---|---|---|---|---|
| **LR** | ELEMENTARY | 21 s | 21 s | 2.4e4 | 0.79 |
| | TUFFY | 58 s | 59 s | 2.4e4 | 0.79 |
| | ALCHEMY | 3140 s | 3152 s | 3.4e4 | 0.14 |
| **CRF** | ELEMENTARY | 35 s | 35 s | 4.6e5 | 0.90 |
| | TUFFY | 148 s | 186 s | ∞ (6.4e5) | 0.14 (0.14) |
| | ALCHEMY | 740 s | 760 s | 1.4e6 | 0.10 |
| **CC** | ELEMENTARY | 11 s | 11 s | 1.8e4 | 0.35 |
| | TUFFY | 977 s | 1730 s | ∞ (2.0e4) | 0.33 (0.32) |
| | ALCHEMY | 2622 s | 2640 s | ∞ (4.6e5) | 0.54 (0.49) |

on most datasets, we use single-thread runtime when plotting graphs. Following standard practice in numerical optimization (Anstreicher & Wolsey, 2009), we use the diminishing step-size rule $\alpha_k = 1/k$ (where $k$ is the number of iterations) for ELEMENTARY. We found that alternative step size rules may result in faster convergence. We run MAP inference on each system with each dataset and plot the MLN cost against runtime.

***Results.*** Recall that the goal of MAP inference is to lower the MLN cost as quickly as possible. From Figure 8 we see that ELEMENTARY achieves dramatically better performance compared to TUFFY: while TUFFY fails to find a feasible solution (i.e., a solution with finite cost) after 5000 seconds on IERJ and 5 hours on KBP, ELEMENTARY converges to a feasible solution within minutes on each dataset. There are complex structures in these MLNs; e.g., transitivity for entity resolution and uniqueness constraints for entity linking in KBP. TUFFY were not able to find feasible solutions that satisfy such complex constraints, and the corresponding curves were obtained by replacing hard rules with a "softened" version with weight 100. Still, we see that the results from TUFFY are substantially worse than ELEMENTARY. From the above comparison, we conclude that the decomposition-based approach to inference is able to achieve significantly higher efficiency than prior (monolithic) approaches.

***Scalability.*** To test scalability, we also run ELEMENTARY on the large KBP+ dataset with a parallel RDBMS (from Greenplum Inc.). This MLN converges within a few iterations; an iteration takes about five hours in ELEMENTARY.

***Specialized Tasks.*** We validate that the ability to integrate specialized tasks into MLN inference is key to ELEMENTARY's high efficiency and quality. To do this, we demonstrate that ELEMENTARY's specialized algorithms outperform generic MLN inference algorithms in both quality and efficiency when solving specialized tasks. To evaluate this claim, we run ELEMENTARY, TUFFY, and ALCHEMY on three MLN programs that each encode one of the following tasks: logistic regression-based classification (LR), linear-chain CRF-based classification (CRF), and correlation clustering (CC). To measure application-level quality (F1 scores), we select some datasets with ground truth: we use a subset of the Cora dataset[18] for CC, and a subset of the CoNLL 2000 chunking dataset[19] for LR and CRF. As shown in Table 3, while it always takes less than a minute for ELEMENTARY to finish each task, the other three approaches take much longer. Moreover, both inference quality (i.e., cost) and application-level quality (i.e., F1) of ELEMENTARY are better than TUFFY and ALCHEMY.[20]

### Quality Experiments

A second main hypothesis of this paper is that one can improve KBC result quality by combining more signals while resolving inconsistencies among these signals with statistical inference. Although the experiment results of several recent KBC projects have suggested that this is the case (Carlson et al., 2010; Kasneci et al., 2008; Nakashole et al., 2011; Zhu et al., 2009), we use several more datasets to validate this hypothesis with ELEMENTARY. Specifically, we use ELEMENTARY to implement solutions to six different KBC tasks and measure how the result quality changes as we vary the the amount of signals (in the form of input data resources or MLN rules) in each solution. We find that overall more signals do tend to lead to higher quality in KBC tasks.

***Datasets and MLNs.*** We consider six KBC tasks for the quality experiments:

1. **TAC**, which is the TAC-KBP challenge as described in the previous subsection. The MLN combines signals

[18] `http://alchemy.cs.washington.edu/data/cora`

[19] `http://www.cnts.ua.ac.be/conll2000/chunking/`

[20] The only exception is ALCHEMY's F1 on CC, an indication that the CC program is suboptimal for the application.

Table 4. *Signals in the four versions of MLN programs for each KBC task. DS means "distant supervision"; EL means "entity linking"; RE means "relation extraction."*

| | Base | Base+EL | Base+RE | Full |
|---|---|---|---|---|
| **TAC** | string-based EL, DS for RE | +web search | +domain knowledge | all |
| **CIA** | string-based EL, DS for RE | +name variations | +domain knowledge | all |
| **IMDB** | string-based EL, DS for RE | +erroneous names | +domain knowledge | all |
| **NFL** | CRF-based winner/loser labeling | N/A | +domain knowledge | all |
| **Enron** | person-phone co-occurrence heuristic | +person coref | +domain knowledge | all |
| **DBLife** | person-org co-occurrence heuristic | +name variations | +relaxed co-occurrences | all |

from Standord NER[21], dependency paths from the Ensemble parser (Surdeanu & Manning, 2010), web search results from Bing[22] (querying mention strings), and developers' feedback on linguistic patterns (in the form of MLN rules). For relation extraction, we perform distant supervision with Freebase as the training KB; Freebase is disjoint from the TAC-KBP benchmark ground truth. ELEMENTARY's quality for TAC-KBP is comparable to the state of the art (Ji et al., 2010) – we achieved a F1 score of 0.80 on entity linking (the best score in KBP2010 was 0.82; human performance is about 0.90) and 0.31 on slot filling (the best score in KBP2010 was 0.65, but all the other teams were lower than 0.30).

2. **CIA**, where the task is to populate the ternary person-title-in-country (PTIC) relation by processing the TAC corpus. The MLN combines signals such as a logistic regression model learned from distant supervision on linguistic patterns between entity mentions (including word sequences and dependency paths), name variations for persons, titles, and countries, and developer's feedback on the linguistic patterns. The ground truth is from the CIA World Factbook.[23] Following recent literature on distant supervision (Mintz et al., 2009; Yao, Riedel, & McCallum, 2010; Hoffmann, Zhang, Ling, Zettlemoyer, & Weld, 2011), we randomly split the World Factbook KB into equal-sized training set and testing set, and perform distant supervision with the training set to obtain a relation-extraction model. We measure output quality on the testing set portion of the KB. We will also report results after swapping the training and testing sets.

3. **IMDB**, where the task is to extract movie-director and movie-actor relationships from the TAC corpus. The MLN combines signals such as a logistic regression-based distant supervision model using linguistic patterns between entity mentions (including word sequences and dependency paths), a manually crafted list of erroneous movie names, and developer's feedback on linguistic patterns. The ground truth is from the IMDB dataset.[24] For relation-extraction model training and quality evaluation, we follow the same methodology as in CIA.

4. **NFL**, where the task is to extract National Football League game results (winners and losers) in the 2006-07 season from about 1.1K sports news articles. The MLN combines signals from a CRF-based team mention extractor, a dictionary of NFL team-name variations, and domain knowledge such as "a team cannot be both a winner and a loser on the same day." The CRF component is trained on game results from another season. The weights of the other rules are heuristically set. We use the actual game results as ground truth.[25]

5. **Enron**, where the task is to identify person mentions and associated phone numbers from 680 emails in the Enron dataset.[26] The MLN is derived from domain knowledge used by a rule-based IE system (Chiticariu et al., 2010; Liu, Chiticariu, Chu, Jagadish, & Reiss, 2010) – it combines signals such as a list of common person names and variations, regular expressions for phone numbers, email senders' names, and domain knowledge that "a person doesn't have many phone numbers." The rule weights are heuristically set; no weight learning is involved. We use our manual annotation of the 680 emails as ground truth.

6. **DBLife**, where the task is to extract persons, organizations, and affiliation relationships between them from a collection of academic webpages.[27] The MLN is derived from domain knowledge used by another rule-based IE system (DeRose et al., 2007) – it combines signals such as person names and variations

[21] http://nlp.stanford.edu/software/index.shtml
[22] http://www.bing.com/toolbox/bingdeveloper/
[23] https://www.cia.gov/library/publications/the-world-factbook/
[24] http://www.imdb.com/interfaces
[25] http://www.pro-football-reference.com/
[26] http://www.cs.cmu.edu/~einat/datasets.html
[27] http://dblife.cs.wisc.edu

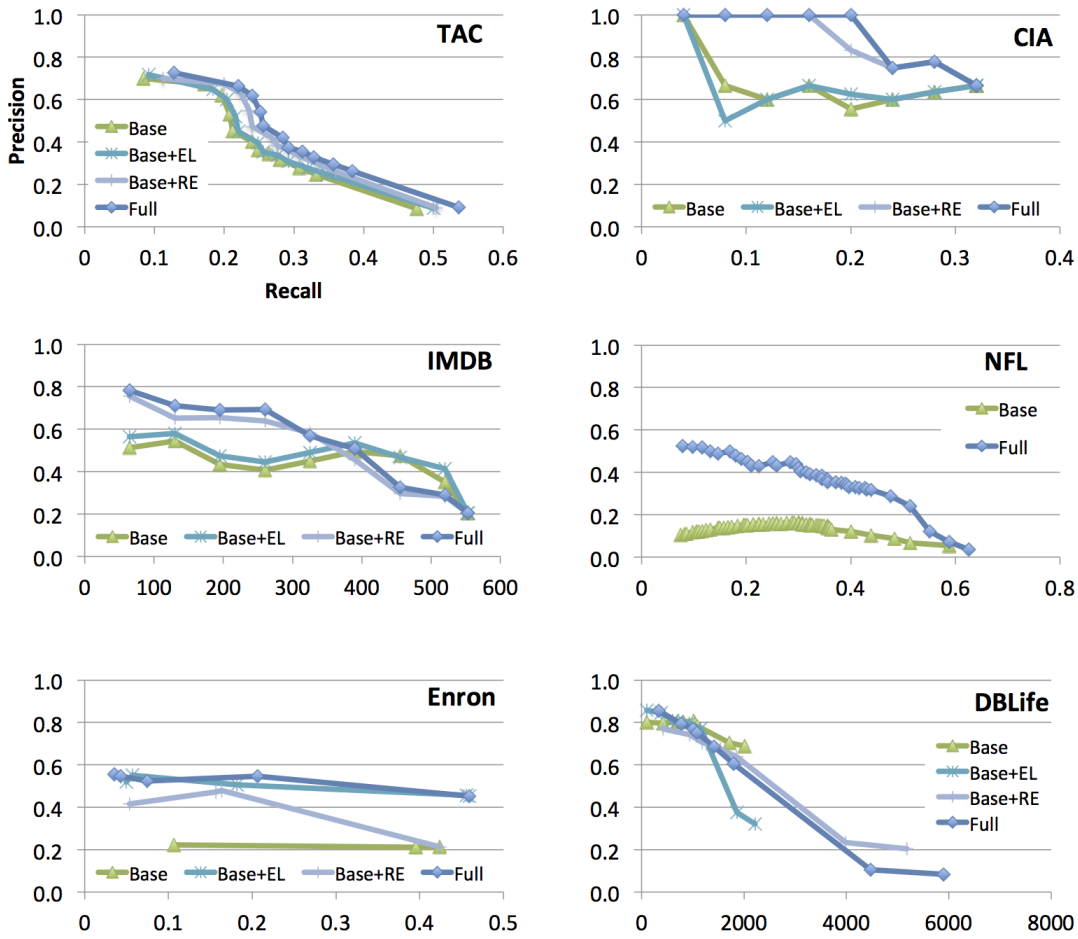


Figure 9. *Result quality of KBC tasks improves as we add more signals into* ELEMENTARY. *On each of the six tasks, we run four versions of* ELEMENTARY *programs and plot the precision-recall curves for each version:* ***Base*** *is the baseline version,* ***Base+EL*** *has enhanced signals (see Table 4) for entity linking,* ***Base+RE*** *has enhanced signals (see Table 4) for relation extraction, and* ***Full*** *has enhanced signals for both entity linking and relation extraction. Recall axes with integer labels correspond to datasets where we use the true positive count to measure recall; the reason is that the corpus covers only a small portion of the ground-truth KB. NFL has only two curves because we do not have enhanced EL signals.*

(e.g., first/last name only, titles), organization names and a string-based similarity metric, and several levels of person-organization co-occurrence (adjacent, same-line, adjacent-line, etc.). The rule weights are heuristically set; no weight learning is involved. We use the ACM author profile data as ground truth.[28]

***Methodology.*** For each of the above KBC tasks, we consider four versions of MLN programs with different amounts of signals (Table 4): **Base** is a baseline, **Base+EL** (resp. **Base+RE**) has enhanced entity-linking (EL) (resp. relation-extraction (RE)) signals, and **Full** has all signals. For each task, we run each version of MLN program with marginal inference on ELEMENTARY until convergence. We then take the output (sorted by marginal probabilities) and plot a precision-recall curve. In precision-recall curves, higher is better.

***Results.*** As shown in Figure 9, the overall result is that more signals do help improve KBC quality: on each task, the quality of either Base+EL or Base+RE improves upon Base, and the Full program performs the best among all four versions. Specifically, on the TAC task, enhancing either EL or RE signals improves both precision and recall; furthermore, the combination of such enhancements (resulting in the Full program) achieves even higher quality than all the other three settings. These results suggest that integrating signals with statistical inference is a promising approach to KBC challenges such as TAC-KBP. On the single-relation KBC tasks CIA, NFL, and Enron, we observe even more significant quality improvement as

[28] `http://www.acm.org/membership/author_pages`

more signals are used by ELEMENTARY: on each dataset, the Full program has almost twice as high precision at each recall level compared to the Base program. On IMDB, the addition of enhanced EL signals slightly but consistently improves precision (Base+EL over Base, and Full over Base+RE); the addition of enhanced RE signals effectively helps segregate high-confident results from low-confident results (the slope of the curve becomes steeper). On DBLife, the enhanced EL or RE signals help improve the precision of the most-confident results (the first 500 predictions or so), while extending the largest recall value substantially. From the above results, we conclude that one can improve the quality of KBC tasks by integrating diverse data and signals via statistical inference. To further verify the above observations, we also swap the training and testing sets for IMDB and CIA. On either IMDB or CIA, we obtained essentially the same results as in Figure 9.

## Conclusion and Future Work

Motivated by several recent knowledge-base construction projects, we study how to build KBC systems with machine learning and statistical inference to combine data and algorithms from multiple sources. We describe a Markov logic-based KBC model and architecture, and demonstrate how to integrate different kinds of data resources and KBC techniques. Operationally, ELEMENTARY consists of two main processing steps: feature extraction and statistical inference. We scale to handle feature extraction from terabytes of text with mature parallel-computing infrastructure such as Hadoop and Condor. To scale up statistical inference, we propose a decomposition-based approach to MLN inference. We experimentally show that our approach has higher efficiency and scalability than prior MLN inference approaches. To evaluate the overall ELEMENTARY approach to KBC, we empirically show that, as we introduce more signals into ELEMENTARY, the result quality improves across tasks.

ELEMENTARY is an evolving prototype, and there are many more challenges we plan to address in future work. Below are some examples:

- To incorporate a KBC technique, it must either fit in the MLN language (e.g., CRFs, LRs, and rules expressible with SQL joins), or we have to somehow transform the technique into relational evidence. Thus, it may be difficult to accommodate certain KBC techniques (especially those involving continuous variables and complex aggregation steps). An interesting challenge is to find a way to support these techniques. For example, we could consider extending Markov logic to support more general factor graphs. Another possible direction is to introduce elements from imperative programming languages such as conditioning and looping (McCallum et al., 2009).

- Another interesting topic is how to automatically perform MLN decomposition and select inference algorithms based on the structure of the MLN. Also, currently inference on subtasks is repeatedly executed from scratch at each iteration; how do we improve this, say, with incremental inference or warm-start?

- A valuable feature for ELEMENTARY would be additional support for debugging: to facilitate the development process, ideally we would like ELEMENTARY to be able to help a developer understand the connection between adding/dropping a particular signal and changes in the output. Possible approaches to such functionality include feature selection, active learning (Settles, 2012), and crowdsourcing.

## Acknowledgement

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. CR is also generously supported by NSF CAREER award under IIS-1054009, ONR award N000141210041, and gifts or research awards from Google, Greenplum, Johnson Controls, Inc., LogicBlox, and Oracle. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, AFRL, or the US government. We are thankful for the generous support from the Center for High Throughput Computing, the Open Science Grid, and Miron Livny's Condor research group at UW-Madison. All URLs in this paper are active as of July 20, 2012.

## References

Ailon, N., Charikar, M., & Newman, A. (2008). Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM*, *55*, 23:1–23:27.

Andrzejewski, D., Livermore, L., Zhu, X., Craven, M., & Recht, B. (2011). A framework for incorporating general domain knowledge into latent Dirichlet allocation using first-order logic. In *Proceedings of the International Joint Conferences on Artificial Intelligence* (pp. 1171–1177).

Anstreicher, K., & Wolsey, L. (2009). Two well-known properties of subgradient optimization. *Mathematical Programming*, *120*(1), 213–220.

Arasu, A., & Garcia-Molina, H. (2003). Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data* (pp. 337–348).

Arasu, A., Ré, C., & Suciu, D. (2009). Large-scale deduplication with constraints using Dedupalog. In *Proceedings of the International Conference on Data Engineering* (pp. 952–963).

Bertsekas, D. (1999). *Nonlinear Programming*. Athena Scientific.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. New York: Cambridge University Press.

Brin, S. (1999). Extracting patterns and relations from the world wide web. In *Proceedings of the International Conference on World Wide Web* (pp. 172–183).

Calì, A., Gottlob, G., & Pieris, A. (2010). Query answering under expressive entity-relationship schemata. *Conceptual Modeling–ER 2010*, *1*, 347–361.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E., & Mitchell, T. (2010). Toward an architecture for never-ending language learning. In *Proceedings of the Conference on Artificial Intelligence* (pp. 1306–1313).

Chen, F., Feng, X., Christopher, R., & Wang, M. (2012). Optimizing statistical information extraction programs over evolving text. In *Proceedings of the International Conference on Data Engineering* (p. 870-881).

Chen, P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, *1*, 9–36.

Chiticariu, L., Krishnamurthy, R., Li, Y., Raghavan, S., Reiss, F., & Vaithyanathan, S. (2010). SystemT: An algebraic approach to declarative information extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (pp. 128–137).

Dean, J., & Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. In *Proceeding of the USENIX Symposium on Operating Systems Design and Implementation* (p. 137-150).

DeRose, P., Shen, W., Chen, F., Lee, Y., Burdick, D., Doan, A., et al. (2007). DBLife: A community information management platform for the database research community. In *Proceeding of the Conference on Innovative Data Systems Research* (pp. 169–172).

Dredze, M., McNamee, P., Rao, D., Gerber, A., & Finin, T. (2010). Entity disambiguation for knowledge base population. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (pp. 277–285).

Fang, L., Sarma, A. D., Yu, C., & Bohannon, P. (2011, November). Rex: Explaining relationships between entity pairs. *Proc. VLDB Endow.*, *5*(3), 241–252.

Fellegi, I., & Sunter, A. (1969). A theory for record linkage. *Journal of the American Statistical Association*, *64*, 1183-1210.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., et al. (2010). Building Watson: An overview of the DeepQA project. *AI Magazine*, *31*(3), 59–79.

Finkel, J., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (pp. 363–370).

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. In *Proceedings of the International Joint Conferences on Artificial Intelligence* (pp. 307–333).

Hearst, M. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on computational linguistics-volume 2* (pp. 539–545).

Hoffart, J., Yosef, M. A., Bordino, I., FâĹŽÂžrstenau, H., Pin, M., Spaniol, M., et al. (2011). Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (p. 782-792).

Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L., & Weld, D. (2011). Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (pp. 541–550).

Ihler, A., Fisher, J., & Willsky, A. (2006). Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, *6*(1), 905–936.

Ji, H., Grishman, R., Dang, H., Griffitt, K., & Ellis, J. (2010). Overview of the TAC 2010 knowledge base population track. In *Text Analysis Conference.*

Kasneci, G., Ramanath, M., Suchanek, F., & Weikum, G. (2008). The YAGO-NAGA approach to knowledge discovery. *SIGMOD Record*, *37*(4), 41–47.

Kautz, H., Selman, B., & Jiang, Y. (1997). A general stochastic approach to solving problems with hard and soft constraints. *The Satisfiability Problem: Theory and Applications*, *17*, 573–586.

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. In *Proceedings of the 22nd international conference on Machine learning* (pp. 441–448).

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(10), 1568–1583.

Komodakis, N., & Paragios, N. (2009). Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *Proceeding of the IEEE Conference*

*on Computer Vision and Pattern Recognition* (pp. 2985–2992).

Komodakis, N., Paragios, N., & Tziritas, G. (2007). MRF optimization via dual decomposition: Message-passing revisited. In *Proceeding of the IEEE International Conference on Computer Vision* (pp. 1–8).

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning* (pp. 282–289).

Lao, N., Mitchell, T., & Cohen, W. (2011). Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 529–539).

Lin, D., & Pantel, P. (2001). DIRT-discovery of inference rules from text. *Knowledge Discovery and Data Mining*, *1*, 323-328.

Liu, B., Chiticariu, L., Chu, V., Jagadish, H., & Reiss, F. (2010). Automatic rule refinement for information extraction. In *Proceedings of International Conference on Very Large Data Bases* (pp. 588–597).

Lowd, D., & Domingos, P. (2007). Efficient weight learning for Markov logic networks. In *European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 200–211).

McCallum, A., Schultz, K., & Singh, S. (2009). Factorie: Probabilistic programming via imperatively defined factor graphs. In *Proceedings of the Annual Conference on Neural Information Processing Systems* (p. 285-âĂŞ292).

Michelakis, E., Krishnamurthy, R., Haas, P., & Vaithyanathan, S. (2009). Uncertainty management in rule-based information extraction systems. In *Proceedings of the SIGMOD International Conference on Management of Data* (pp. 101–114).

Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D., & Kolobov, A. (2005). BLOG: Probabilistic models with unknown objects. In *Proceedings of the International Joint Conferences on Artificial Intelligence* (pp. 1352–1359).

Minka, T. (2001). Expectation propagation for approximate Bayesian inference. In *Proceeding of the Conference on Uncertainty in Artificial Intelligence* (pp. 362–369).

Mintz, M., Bills, S., Snow, R., & Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (pp. 1003–1011).

Mooney, R. (1999). Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (pp. 328–334).

Nakashole, N., Theobald, M., & Weikum, G. (2011). Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the Web Search and Data Mining* (pp. 227–236).

Niu, F. (2012). *Web-scale Knowledge-base Construction via Statistical Inference and Learning (tentative title)*. Unpublished doctoral dissertation, Computer Sciences Department, University of Wisconsin-Madison.

Niu, F., Ré, C., Doan, A., & Shavlik, J. (2011). Tuffy: Scaling up statistical inference in Markov logic networks using an RDBMS. In *Proceedings of International Conference on Very Large Data Bases* (pp. 373–384).

Niu, F., Zhang, C., Ré, C., & Shavlik, J. (2011). Felix: Scaling inference for Markov logic with an operator-based approach. *ArXiv e-prints*.

Poon, H., & Domingos, P. (2007). Joint inference in information extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 913–918).

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Riedel, S. (2008). Improving the accuracy and efficiency of MAP inference for Markov logic. In *Proceeding of the Conference on Uncertainty in Artificial Intelligence* (pp. 468–475).

Riloff, E. (1993). Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 811–811).

Rush, A., Sontag, D., Collins, M., & Jaakkola, T. (2010). On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1–11).

Sarawagi, S. (2008). Information extraction. *Foundations and Trends in Databases*, *1*(3), 261–377.

Sen, P., Deshpande, A., & Getoor, L. (2009). PrDB: Managing and exploiting rich correlations in probabilistic databases. In *Proceedings of International Conference on Very Large Data Bases* (pp. 1065–1090).

Settles, B. (2012). *Active Learning*. Morgan & Claypool.

Shen, W., Doan, A., Naughton, J., & Ramakrishnan, R. (2007). Declarative information extraction using datalog with embedded extraction predicates. In *Proceedings of International Conference on Very Large Data Bases.*

Singh, S., Subramanya, A., Pereira, F., & McCallum, A. (2011). Large-scale cross-document coreference using distributed inference and hierarchical models. In

*Proceeding of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 793–803).

Sontag, D., Globerson, A., & Jaakkola, T. (2010). Introduction to dual decomposition for inference. *Optimization for Machine Learning*, *1*, 1–11.

Suchanek, F., Kasneci, G., & Weikum, G. (2007). Yago: A core of semantic knowledge. In *Proceedings of the International Conference on World Wide Web* (pp. 697–706).

Suchanek, F., Sozio, M., & Weikum, G. (2009). SOFIE: A self-organizing framework for information extraction. In *Proceedings of the International Conference on World Wide Web* (pp. 631–640).

Surdeanu, M., & Manning, C. (2010). Ensemble models for dependency parsing: cheap and good? In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics* (pp. 649–652).

Sutton, C., & McCallum, A. (2004). *Collective segmentation and labeling of distant entities in information extraction* (Tech. Rep. No. 04-49). Department of Computer Science, University of Massachusetts.

Sutton, C., & McCallum, A. (2006). An introduction to conditional random fields for relational learning. In L. Getoor & B. Taskar (Eds.), *Introduction to statistical relational learning.* MIT Press.

Thain, D., Tannenbaum, T., & Livny, M. (2005). Distributed computing in practice: The Condor experience. *Concurrency and Computation: Practice and Experience*, *17*(2-4), 323–356.

Theobald, M., Sozio, M., Suchanek, F., & Nakashole, N. (2010). *URDF: Efficient Reasoning in Uncertain RDF Knowledge Bases with Soft and Hard Rules* (Tech. Rep.). MPI.

Wainwright, M., Jaakkola, T., & Willsky, A. (2005). MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, *51*(11), 3697–3717.

Wainwright, M., & Jordan, M. (2008). *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers.

Weikum, G., & Theobald, M. (2010). From information to knowledge: Harvesting entities and relationships from web sources. In *Proceedings of the ACM Symposium on Principles of Database Systems* (pp. 65–76).

Wick, M., McCallum, A., & Miklau, G. (2010). Scalable probabilistic databases with factor graphs and MCMC. In *Proceedings of International Conference on Very Large Data Bases* (p. 794-804).

Wolsey, L. (1998). *Integer Programming*. Wiley.

Wu, F., & Weld, D. (2010). Open information extraction using Wikipedia. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (pp. 118–127).

Yao, L., Riedel, S., & McCallum, A. (2010). Collective cross-document relation extraction without labelled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1013–1023).

Zhang, C., Niu, F., Ré, C., & Shavlik, J. (2012). Big data versus the crowd: Looking for relationships in all the right places. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

Zhu, J., Nie, Z., Liu, X., Zhang, B., & Wen, J. (2009). Statsnowball: A statistical approach to extracting entity relationships. In *Proceedings of the International Conference on World Wide Web* (pp. 101–110).