StatSnowball: a Statistical Approach to Extracting Entity Relationships[*]

Jun Zhu[†⋆] Zaiqing Nie[‡] Xiaojing Liu[§] Bo Zhang [†⋆] Ji-Rong Wen[‡]

[†]Dept. of Comp. Sci. & Tech.
Tsinghua University
Beijing, 100084 P.R China
{jun-zhu@mails, dcszb@mail}.thu.edu.cn

[‡]Microsoft Research Asia
No. 49 Zhichun Road
Beijing, 100080 P.R China
{znie,jrwen}@microsoft.com

[§]Dept. of EEIS
University of Sci. & Tech. of China
Hefei, 230027 P.R China
xiaojiangliu84@hotmail.com

ABSTRACT

Traditional relation extraction methods require pre-specified relations and relation-specific human-tagged examples. Bootstrapping systems significantly reduce the number of training examples, but they usually apply heuristic-based methods to combine a set of strict hard rules, which limit the ability to generalize and thus generate a low recall. Furthermore, existing bootstrapping methods do not perform open information extraction (Open IE), which can identify various types of relations without requiring pre-specifications. In this paper, we propose a statistical extraction framework called *Statistical Snowball* (StatSnowball), which is a bootstrapping system and can perform both traditional relation extraction and Open IE.

StatSnowball uses the discriminative Markov logic networks (MLNs) and softens hard rules by learning their weights in a maximum likelihood estimate sense. MLN is a general model, and can be configured to perform different levels of relation extraction. In StatSnwoball, pattern selection is performed by solving an ℓ_1-norm penalized maximum likelihood estimation, which enjoys well-founded theories and efficient solvers. We extensively evaluate the performance of StatSnowball in different configurations on both a small but fully labeled data set and large-scale Web data. Empirical results show that StatSnowball can achieve a significantly higher recall without sacrificing the high precision during iterations with a small number of seeds, and the joint inference of MLN can improve the performance. Finally, StatSnowball is efficient and we have developed a working entity relation search engine called *Renlifang* based on it.

Categories and Subject Descriptors

I.5.1 [**Pattern Recognition**]: Models - Statistical

General Terms

Algorithms, Experimentation

[*]This work was done when Jun Zhu and Xiaojiang Liu were visiting Microsoft Research Asia.

[⋆]Jun and Bo are also with the State Key Laboratory of Intelligent Technology and Systems; and the Tsinghua National Laboratory for Information Science and Technology.

Keywords

Relation extraction, statistical model, Markov logic networks

1. INTRODUCTION

The World Wide Web has been growing rapidly as a huge information repository, which contains various kinds of valuable semantic information about real-world entities, such as people, organizations, and locations. We have been working on object-level search engines, which automatically extract and integrate the semantic information about entities and return a list of ranked entities instead of webpages to answer user queries [18]. In object-level search engines, it is particularly important to mine entity relations from the Web to automatically build an entity relationship graph to link all the extracted information together. With the entity relationship graph, users will be able to easily navigate through their interested entities just like the way they navigate through hyperlinked webpages. This paper focuses on entity relation mining from the Web.

1.1 Motivating Example

We have deployed an entity relationship search engine in Chinese search market called *Renlifang*[1]. Renlifang is a different kind of search engine we have been building, one that explores relationships between entities. In Renlifang, users can query the system about people, locations, and organizations and explore their relationships. Currently Renlifang only serves in the Chinese language domain. These entities and their relationships are automatically mined from billions of crawled Chinese webpages. For each crawled webpage in Renlifang, the system extracts entity information and detects relationships, covering a spectrum of everyday individuals and well-known people, locations, or organizations. Below we list the key features of Renlifang:

Entity Relationship Mining and Navigation. Renlifang enables users to explore highly relevant information during searches to discover interesting relationships about entities associated with their queries.

Finding Expertise. Renlifang can return a ranked list of people known for dancing or any other topic.

Web-Prominence Ranking. Renlifang detects the popularity of an entity and enables users to browse entities in different categories ranked by their prominence on the Web during a given time period.

[1]http://renlifang.msra.cn

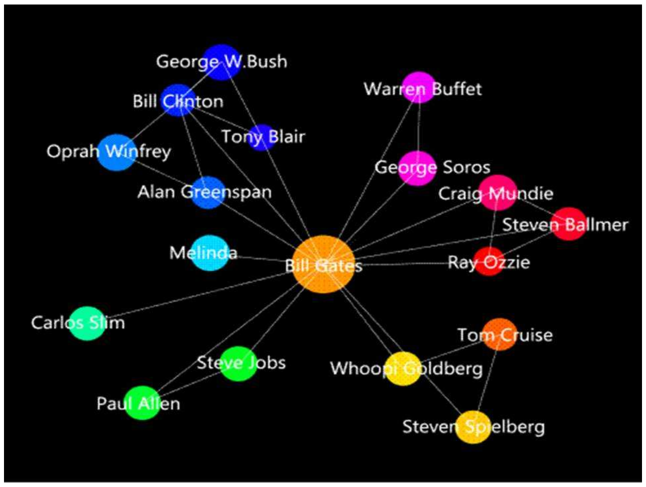

Figure 1: An entity relationship graph for the query "Bill Gates" generated by EntityCube (i.e. the English version of Renlifang).

People Bio Ranking. Renlifang ranks text blocks from webpages by the likelihood of their being biography blocks.

Renlifang has been well received by Chinese Internet users and mainstream media in China (including CCTV and Phoenix TV) with positive comments and millions of daily page-views during the peak days. The English version of Renlifang is called EntityCube, and it is currently under development[2]. In Figure 1, we show an automatically generated entity relationship graph using our English Renlifang prototype.

Based on the overwhelming response from Chinese Internet users of the Renlifang entity relationship search, we found that automatically extracting a large number of highly accurate entity relations is important to improve performance. However, existing work on entity relation extraction in the literature could not meet the requirements of a Web-scale entity relationship search engine.

Relation extraction has been promoted by the Message Understanding Conference (MUCs) and Automatic Content Extraction (ACE) program. The task has been traditionally studied so as to extract *predefined* semantic relations between pairs of entities in text, e.g., in supervised learning methods [27, 8, 9, 26] and bootstrapping systems [5, 1]. Supervised methods require a set of relation-specific human tagged examples to learn an extractor. Labeling training examples is tedious and labor intensive, thus, it makes supervised methods difficult to apply to Web-scale applications like Renlifang. Bootstrapping methods [5, 1, 7] significantly reduce the number of training examples by iteratively discovering extraction patterns and identifying entity relations with a small number of seeds, either target relation tuples [1] or general extraction templates [7]. Take the well-known Snowball [1], which serves as the basis of our proposed approach, as an example. Snowball takes a small set of seed tuples as inputs, and employs the pattern-entity duality [5] to iteratively generate extraction patterns and identify new relation tuples. From the generated patterns and identified tuples, some confidence measures are carefully crafted to select good ones and add them to Snowball as new knowledge. Evaluating patterns and tuples is one key component, since it is crucial to select good patterns and good new seed tuples to make sure the system will not be drifted by errors.

[2]www.entitycube.com

Although the bootstrapping architecture is promising, Snowball has at least two obvious limitations, which make it unsuitable for the Web-scale relation extraction as motivated by Renlifang. First, since the target of Snowball is to extract a specific type of relation (e.g., companies and their headquarters) the extraction patterns in Snowball are mainly based on strict keyword-matching. Although these patterns can identify highly accurate results, the recall will be limited. Second, Snowball does not have an elegant evaluation measure, such as the probability/likelihood of a probabilistic model, to evaluate generated patterns. The carefully crafted measures and pattern selection criteria are not directly adaptable to general patterns (e.g., POS tag sequences), which can significantly improve the recall as shown in our empirical studies. This is because many tuples extracted by a general pattern are more likely not to be the target relations of Snowball, although they can be other types of relations. In this case, the confidence scores will be very small, and it is inappropriate to use the criteria as used in Snowball to select these patterns.

In this paper, we address these issues as suffered by Snowball to improve the recall while keeping a high precision. We present a system called *Statistical Snowball* (StatSnowball). StatSnowball adopts the bootstrapping architecture and applies the recently developed feature selection method using ℓ_1-norm [25, 11] to select extraction patterns—both keyword matching and general patterns. Starting with a handful set of initial seeds, it iteratively generates new extraction patterns; performs an ℓ_1-norm regularized maximum likelihood estimation (MLE) to select good patterns; and extracts new relation tuples. StatSnowball is a general framework and the statistical model can be any probabilistic model. StatSnowball uses the general discriminative Markov logic networks (MLN) [21], which subsume logistic regression (LR) and conditional random fields (CRF) [15]. Discriminative models can incorporate arbitrary useful features without strong independence assumptions as made in generative models, like naïve Bayes (NB) and Hidden Markov Models (HMM).

By incorporating general patterns, StatSnowball can perform both traditional relation extraction like Snowball to extract pre-specified relations and open information extraction (Open IE) [3] to identify general types of relations. Open IE is a novel domain-independent extraction paradigm, which has been studied in both the natural language document corpus [22] and the Web environment [3]. Although the existing Open IE systems are self-supervised, they require a set of human-selected features in order to learn a good extractor. In contrast, StatSnowball automatically generates and selects the extraction patterns. Moreover, the Open IE systems require expensive deep linguistic parsing techniques to correctly label training samples, while StatSnwoball only uses cheaper and more robust shallow parsing techniques to generate its patterns. Finally, by using the MLN model, StatSnowball can perform joint inference, while the O-CRFs [4] treat sentences independently. Our empirical studies demonstrate the promise of using joint inference.

To the best of our knowledge, StatSnwoball is the first working system that takes a bootstrapping architecture and applies the well-developed ℓ_1-norm regularized MLE to incrementally identify entity relations. Specifically, we make the following contributions:

(a) We present a novel framework called StatSnowball by using ℓ_1-regularized feature selection to incrementally discover extraction patterns and identify relation tuples. Compared to the closely related Snowball, StatSnowball contains the following advantages:

   i StatSnowball can perform both traditional relation extraction as in Snowball and Open IE.

   ii The probabilistic foundation provides StatSnowball a principled approach to evaluating and selecting patterns. In Snowball, however, the confidence measures lack an elegant interpretation and they are difficult to be applied to general patterns.

   iii StatSnowball automatically learns the weights of generated patterns, which are represented as formulae in MLN, while Snowball applies some heuristic rules to assign the weights.

   iv StatSnowball can be easily extended. In current implementation, StatSnowball takes the same strategy as Snowball to separately identify named entities and entity relations. StatSnowball can easily integrate these two tasks into one probabilistic model and thus can achieve a higher performance by exploring their mutual enhancements. The promise of integrated extraction has been shown in different applications [20, 28]. But the heuristic-based Snowball is hard to be coherently integrated with the state-of-the-art statistical named entity extraction systems.

(b) We extensively evaluate StatSnowball and empirically show that StatSnowball can achieve a significantly higher precision and recall on large scale Web data.

(c) StatSnowball is efficient and we have developed a working Entity Relation Search Engine based on it.

The rest of the paper is structured as follows. Section 2 briefly overviews the StatSnowball system. Section 3 presents the statistical model (i.e., Markov logic networks), including some speeding up techniques. Section 4 presents how to generate and select good patterns in StatSnowball. Section 5 presents our empirical results. Section 6 discusses some related work, and Section 7 concludes this paper, with some future work discussed.

2. OVERVIEW OF STATSNOWBALL

In this section, we briefly overview the framework of StatSnowball. Different components will be explained in incoming sections.

The task of StatSnowball is to identify relation tuples. Each relation tuple can be among several entities. Here, we only focus on the binary relationship, and an extraction is a tuple (e_i, e_j, key) $i \neq j$, where e_i and e_j are two entities and key is a set of keywords that indicate the relationship. Like many other relation extraction systems [1, 8, 9], we assume that the entities are given and focus on how to detect (i.e., decide whether a relationship exists between two entities) and categorize (i.e., assign relation keywords to a detected relationship) the relationships.

Figure 2 shows the architecture of StatSnowball. Generally, StatSnowball has three parts. The first part P$_1$ is the input, which contains a set of seeds and an initial model. The seeds are not required to contain relation keywords that

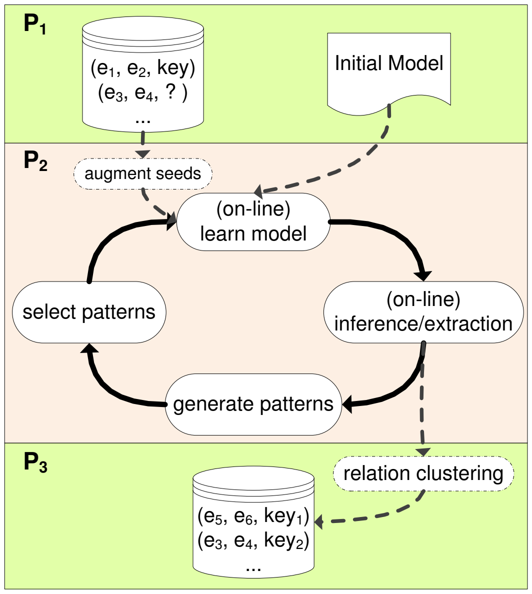

Figure 2: The framework of StatSnowball, with three parts – P$_1$ (input), P$_2$ (statistical extraction model), and P$_3$ (output).

indicate the relationship. Thus, we have two types of seeds, i.e., seeds with relation keywords like (e_1, e_2, key) or seeds without relation keywords like $(e_3, e_4, ?)$. If the initial model is empty, we will first use the seeds to generate extraction patterns in order to start the process.

The second part P$_2$ is the statistical extraction model. To start the iterative extraction process, StatSnowball takes the input seeds and the initial model (can be empty) in P$_1$ to learn an extractor. We apply the ℓ_2-norm regularized maximum likelihood estimation (MLE) at this step. Online learning is an alternative if batch learning is expensive. Then, StatSnowball uses the learned model to extract new relation tuples on the data corpus. The third step in P$_2$ is to generate extraction patterns with the newly identified relation tuples. These patterns are used to compose formulae of MLN. Finally, it selects good formulae to add to the probabilistic model and re-train the model. In this step, we first do ℓ_1-norm regularized MLE, which will set some formulae's weights to zeros. Then, we remove these zero-weighted formulae and send the resultant model to the next step for re-training. StatSnowball iteratively performs these four steps until no new extraction tuples are identified or no new patterns are generated. In this part, an optional component is the augmenting seeds, which can be used to find more seeds to start the process. In order to get high quality training seeds, this component applies strict keyword matching rules. We do not use it in the current system.

The third part P$_3$ is the output, which is necessary only when StatSnowball is configured to do Open IE [3]. When StatSnowball performs Open IE, the extraction results in P$_2$ are general relation tuples. To make the results more readable, we can apply clustering methods to group the relation tuples and assign relation keywords to them. The missing keywords of the seeds can be filled in this part. Currently, we treat it as a post-processing step. Recent work on relational clustering with MLN in [14] suggests that we can move P_3 into P_2 to do joint Open IE. We will discuss this more later.

Before going into the full exposition of the system, let's end this section with a strict mathematical formulation of StatSnowball. Formally, StatSnowball iteratively solves an ℓ_1-norm regularized optimization problem:

$$\text{P}: \quad \mathbf{w}^\star = \arg\min_{\mathbf{w}} LL(\mathcal{D}, \mathcal{R}, \mathbf{w}) + \lambda\|\mathbf{w}\|_1.$$

where $LL(\mathcal{D}, \mathcal{R}, \mathbf{w})$ is the loss defined on the corpus $\mathcal{D}$ given a set of patterns (which are represented as formulae in the probabilistic model as we shall see) $\mathcal{R}$ and the model weights $\mathbf{w}$; and $\|.\|_1$ is the ℓ_1-norm. The data corpus $\mathcal{D}$ and the pattern set $\mathcal{R}$ are updated at each iteration. For $\mathcal{D}$, by changing, we mean that new relation tuples are identified. For $\mathcal{R}$, the change is in the sense that new patterns are added. In the problem P, the loss can be the log-loss as used in probabilistic models or the hinge loss as used in support vector machines [6]. In this paper, we focus on the log-loss. This ℓ_1-norm regularized MLE problem yields a sparse estimate by setting some components of $\mathbf{w}$ to exact zeros [24, 11] and has efficient solvers, such as the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) method [2].

3. THE STATISTICAL MODEL

In this section, we define the task of entity relationship identification and present the probabilistic models we applied in StatSnowball, including the training and inference (extraction) parts in P_2.

3.1 Extraction Task

The task of StatSnowball is to identify related entity pairs and detect the keywords that indicate the relationships. Like many other relation extraction systems [1, 8, 9], we assume that entities are given. In probabilistic models, the task of relation extraction is to predict whether two entities e_i and e_j have a relation R based on the probability $p(R(e_i, e_j)|O)$. For relation keyword detection, the task is to predict whether a token is a relation keyword. In StatSnowball, we define three fields (labels) that a token can belong to, namely, REL-S: the start of a relation; REL-C: a continuation of a relation; and NULL: not a relation keyword. We assume that each token can belong to one field. Then, relation keyword detection is to predict in which field (label) f the token t is most likely to be based on the probability $p(InField(t, f)|O)$, where $f \in$ {REL-S, RES-C, NULL}, and O denotes the observations that are available to make the prediction. In discriminative models, O can be arbitrary features of the inputs, e.g., the text content of a token or its neighboring tokens.

Based on the power of available probabilistic models, we can define the task at three different levels:

a. **Entity-Level:** this is the simplest extraction model with a strong independence assumption that whether two entities have some relationship is independent of other entities and is also independent of relation keyword detection. Logistic regression (LR) model is for this task.

b. **Sentence-Level:** since in human languages, the words in a sentence are not independent of each other to express a specific meaning, the independence assumption of the Entity-Level model is too strong. A Sentence-Level model relaxes this assumption and treats a sentence as a whole input and jointly detects whether a pair of entities (if any) in that sentence have some relationship and whether the tokens around the entities indicate the relation type. One example is presented in [4], where entities are assumed to be at the ends of a sentence and the tokens in-between are classified to be relation keywords or not by a linear-chain CRF [15].

c. **Page-Level or Corpus-Level:** since the sentences in a webpage or a text document are not completely independent, it may be desirable to jointly extract these related sentences. Joint inference has been shown to be effective to get globally consistent extraction results, such as [17, 28, 20]. Markov logic network (MLN) has the full power to jointly model correlated data. We provide this alternative and will empirically demonstrate the advantages of joint inference in StatSnowball.

As we have stated, we can use any probabilistic model in StatSnowball. In order to accommodate the above three levels of relationship extraction, StatSnowball adopts the most general MLN model, which can be configured to do the LR-based Entity-Level and the CRF-based Sentence-Level relation extraction, as we shall see.

3.2 Markov Logic Networks

A first-order knowledge base contains a set of formulae, which are constructed using constants, variables, functions, and predicates. Constants are the objects (e.g., entities and tokens) in the interested domain and variables range over the objects. For example, "Bob" and "Jim" are two people entities, and "killed" is a token. e and t are variables which denote an entity and a token, respectively. A function is a mapping from a set of objects to objects (e.g., MotherOf(e_i)) and a predicate represents a relation among objects (e.g., HasRelation(e_i, e_j) or some attributes (e.g., IsPeople(e_i)). An atom is a predicate applied to a set of arguments, which are constants or variables. If an atom's arguments are all constants, it is a ground atom. A world is an assignment of truth values to all possible ground atoms.

If a world violates one formula, it is impossible. Thus, the formulae in a first-order logic can be viewed as a set of hard constraints on the possible worlds. Markov logic is a probabilistic extension and softens the hard constraints by assigning a weight to each formula. The weight indicates how strong the corresponding formula is. When a world violates some formulae it is less impossible, but not impossible. For the task of entity relation extraction, we know the query predicates and the evidence predicates *a prior*. Thus, we partition the ground atoms into two sets—the set of evidence atoms X and the set of query atoms Q, and define a discriminative MLN [23]. Discriminative models have shown great promise as compared to generative models in many applications [15, 23]. In StatSnowball, X can be all the possible features we can extract from the inputs, and Q can be all the relationship queries $R(e_i, e_j)$, $\forall i \neq j$ and keyword detection queries $InField(t, f)$ $\forall t, f$. Given an input $\mathbf{x}$ (e.g., a sentence and its features), the discriminative MLN defines a conditional distribution $p(\mathbf{q}|\mathbf{x})$ as follows:

$$p(\mathbf{q}|\mathbf{x}) = \frac{1}{Z(\mathbf{w}, \mathbf{x})} \exp\Big(\sum_{i \in F_Q} \sum_{j \in G_i} w_i g_j(\mathbf{q}, \mathbf{x})\Big), \qquad (1)$$

where F_Q is the set of formulae with at least one grounding involving a query atom, G_i is the set of ground formulae of the ith first-order formula, and $Z(\mathbf{w}, \mathbf{x})$ is a normalization factor, also known as partition function in physics. $g_j(\mathbf{q}, \mathbf{x})$ is a binary function and equals to 1 if the jth ground formula is true and 0 otherwise.

Markov Logic Networks have the power of first-order logic to model complex relational databases. In the experiments, we will show examples of using MLN to do joint inference in StatSnowball. In StatSnowball, we apply the discriminative learning algorithm [23, 10] to learn the model weights with a sphere Gaussian prior, or equivalently the ℓ_2-norm penalized MLE, to avoid over-fitting.

3.2.1 Special Case 1: Logistic Regression

As we have stated, logistic regression (LR) is the simplest Entity-Level extraction model. It makes a strong independence assumption that entity pairs are independent of each other to have some relationship. Also, the existence of a relationship is independent of relation keyword detection. By restricting all the formulae in MLN to be the ones in which the query predicates can appear *ONLY ONCE*, the resultant MLN reduces to an LR model, and the distribution in Eq. (1) has the factorized form: $p(\mathbf{q}|\mathbf{x}) = \prod_{ij} p(R(e_i, e_j)|\mathbf{x}_{ij}) \prod_t p(\mathit{InField}(t, f_t)|\mathbf{x}_t)$, of which each component is an exponential family distribution. For example, $p(R(e_i, e_j)|\mathbf{x}_{ij})$ has the exponential form: $p(R(e_i, e_j)|\mathbf{x}_{ij}) \propto \exp\big(\sum_{i \in F_R} \sum_{j \in G_i} w_i g_j(R(e_i, e_j), \mathbf{x}_{ij})\big)$, where F_R are the formulas in which the query predicate R appears. The observations are the inputs $\mathbf{x}_{ij}$ related to the entities e_i and e_j. For relation keyword detection, $p(\mathit{InField}(t, f)|\mathbf{x}_t)$ has the similar distribution form.

3.2.2 Special Case 2: CRFs

In a Sentence-Level extraction model, entities and the tokens in the same sentence are not independent. Without context tokens, we cannot decide whether two entities in a sentence have some relationship. On the other hand, whether a token is a relation keyword is dependent on its surrounding tokens. For example, for the sentence “Google forced to buy YouTube.”, which contains the entities “Google” and “YouTube”, the verb “buy” indicates an acquirement relationship between the two entities and the verb “forced” is not a relation keyword because the following “buy” is more likely to be. The LR model cannot consider this mutual dependence information. Instead, for Sentence-Level extraction, we need to apply the linear-chain conditional random fields (CRFs) [15]. The MLN reduces to a linear-chain CRF by defining the following first-order formulae:

$$\mathit{InField}(t_i, \text{REL-S}) \wedge \mathit{Verb}(t_{i+1}) => \mathit{InField}(t_{i+1}, \text{REL-C}),$$

which means when a token is the start of a relation (REL-S), then the following verb is more likely to be a continuation of the relation (REL-C). One application of CRFs to identify relation keywords is presented in [4].

3.3 Formulating Extraction Queries

The second step of the part P_2 is to extract new relation tuples, which is an inference problem in probabilistic models. Suppose the current MLN model is $\mathcal{M}$. For each pair of entities (e_i, e_j), we use $\mathcal{M}$ to predict whether a relationship exists between e_i and e_j with the probability $p(R(e_i, e_j)|\mathbf{x}_{ij}, \mathcal{M})$. For each token t, we use $\mathcal{M}$ to predict whether t is a relation keyword. Here, the query $R(e_i, e_j)$ is a binary predicate and equals to 1 if a relationship exists between e_i and e_j and 0 otherwise. Thus, it is natural to use the probability $p(R(e_i, e_j)|\mathbf{x}_{ij}, \mathcal{M})$ as a confidence measure of the identified new tuple. We can choose a threshold c and keep the candidate extraction (e_i, e_j) only if $p(R(e_i, e_j)|\mathbf{x}_{ij}, \mathcal{M}) > c$. The higher the c, the stricter the decision rule is. If a high c is chosen during the iterations of StatSnowball, only high-quality relation tuples are selected and passed to the next step for generating patterns. For relation keyword detection, the query $\mathit{InField}(t, f)$ is ternary. We predict each token t to the label f which has the highest probability, that is, $f^\star = \arg\max_f p(\mathit{InField}(t, f)|\mathbf{x}_t, \mathcal{M})$. Similar to the use of $p(R(e_i, e_j)|\mathcal{M})$, we can use the probability as a confidence measure of the extraction.

3.3.1 Speeding Up Techniques

For the probabilistic models we are using in StatSnowball, the computational cost of both training and inference mainly depends on two factors—the number of queries and the complexity of the model. Since the number of relation keyword detection queries is linear to the number of tokens, we only consider the relation query R. For logistic regression, the model is very simple and the cost depends on how many relation queries are formulated. For the linear-chain structured CRF [15], it is also very efficient to do inference and learning by using dynamic programming methods. Moreover, as a Sentence-Level extraction model, the number of relation queries in CRFs is linear to the number of sentences. For the general MLN, its complexity depends on the number of formulae and their formulations. As we shall see, StatSnowball only generates very simple formulae. Thus, the cost mainly depends on the number of relation queries.

Naïvely formulating all possible relation queries for every combination of two entities, as in the current Alchemy[3], the whole number of queries is quadratic to the number of entities in a corpus for both LR and MLN. In our application of relation extraction, only when the entities appear within a short range, they could have enough evidence to indicate some relation between them. For two isolated entities, even human readers could have difficulty in deciding whether they have a relationship. Thus, we make some gentle assumptions in StatSnowball. For example, in Page-Level extraction models, we can assume that only the entities appearing in the same webpage/document are likely to have some relationship. When formulating the relation queries, entities in different pages/documents are not considered as a candidate. Similarly, for Sentence-Level extraction models, we can assume that only entities appearing in the same sentences are likely to have relations; otherwise, they do not. These assumptions can significantly reduce the number of queries. For example, suppose we have 200 entities in a corpus of 20 webpages and uniformly each page has 10 entities. Without any assumptions, the number of possible queries will be 200×200 (4×10^4) for all the combinations of two entities. If we make the assumption that only the entities in the same page are possible to be related, then only $20 \times 10 \times 10$ (2×10^3 or 5 percent) queries are possible to be relation tuples. All the other queries are discarded and consume no resources.

With the above independence assumption, an on-line learning method can be used for the learning of MLN. We implemented this technique for the learning of MLN too.

[3]http://alchemy.cs.washington.edu.

4. GENERATE AND SELECT PATTERNS

In this section, we present how to generate and select good extraction patterns in StatSnowball.

4.1 Pattern Generation

Generating new extraction patterns, which are used to compose the formulae of MLNs, is a key component of StatSnowball. A good pattern should achieve a good balance between two competitive criteria—specificity and coverage. Specificity means the pattern is able to identify high-quality relation tuples; while coverage means the pattern can identify a statistically non-trivial number of good relation tuples. In Snowball, the generated patterns are mainly based on keyword matching. These strict patterns can have very high precision, but the coverage (i.e., recall) is very low. As we have stated, there are two reasons why Snowball cannot effectively incorporate general patterns, which can significantly improve the recall as we shall see. First, Snowball was originally proposed to extract a specific type of relationship, for example, companies and their headquarters [1]. In this case, general patterns can yield many invalid extractions, although these extractions could be other types of valid relation tuples. Second, Snowball defines some measures and criteria which are not applicable to general patterns. For pattern evaluation, Snowball defines a confidence measure as the ratio of positive extractions of that pattern. For general patterns, the confidence scores could be very small, and it is difficult to select these patterns by using a threshold.

Instead of defining some heuristic measurements, StatSnowball applies probabilistic models and renders the pattern selection as the ℓ_1-norm regularized optimization problem P. Under this framework, we can treat strict keyword matching patterns and general patterns identically. Also, by using general patterns, StatSnowball can be configured to perform open information extraction, as we have stated. In our experiments, we evaluate both the Open IE and Snowball-like extraction with StatSnowball.

4.1.1 *Keyword-matching Patterns*

In our system, the keywords are from two parts. The first part is from the initial seeds. As shown in Figure 2, users can provide seeds with some keywords to indicate the relationships. We take these keywords to define candidate patterns, e.g., a candidate pattern should contain at least one of these keywords. The second parts of the keywords are those that are automatically discovered during the StatSnowball extraction process. To make the patterns more informative, we only consider three types of keywords as in Table 1 according to the part-of-speech (POS) tags, where MIN_OCCUR is pre-specified number, e.g., 20. Here, we use the same naming system as the Penn Treebank Project [4].

4.1.2 *General Patterns*

Our general extraction patterns are all based on a shallow natural language processing (NLP) technique—part-of-speech tagging (POS). Much work has been done to investigate the usability of shallow or deep linguistic structures for various application tasks such as named entity extraction [7], and relationship identification [9, 8]. In contrast to deep natural language processing, shallow NLP techniques are more robust and more efficient. This is a very important factor for Web-scale relation extraction as in Renlifang. Thus, StatSnowball only uses the part-of-speech tagging results. All the sentences in our data sets are parsed using a part-of-speech tagger. Our general patterns are the POS-tag sequences appearing between entities.

Note that the above two types of patterns are not necessarily independent of each other. For example, the general pattern "POS+NP/NN" can be seen as a keyword matching pattern too because only one token is tagged as POS (i.e., possessive ending), that is, "'s".

4.2 Pattern Selection

Selecting patterns is a feature induction problem of Markov random fields or Markov networks [19, 16]. In MLN, the problem is called structure learning [12]. Alchemy uses a generative approach to learning the structure of MLN by using beam search to generate candidate formulae and selecting good candidates according to the gain in (weighted) pseudo-likelihood. In StatSnowball, we apply the ℓ_1-norm regularized MLE as defined in the problem P and do discriminative structure learning [10]. As we have stated, the ℓ_1-norm penalty encourages a sparse estimate [25].

Specifically, we first use the generated patterns to formulate a set of candidate formulae of MLN. Then, we apply the algorithm [2] to optimize the ℓ_1-norm penalized conditional likelihood function as in the problem P, which yields a sparse model by setting some formulae's weights to zeros. The zero-weighted formulae are discarded and the resultant model is passed to the next step for re-training.

Our method can be viewed as a simplified variant of the discriminative structure learning [10]. Since our patterns (i.e., predicates) have been generated, we can generate the candidate formulae easily instead of applying an ILP system such as Aleph to do this from scratch. Here, we generate the formulae with the N-nary (e.g., binary or ternary) combination of the patterns. As in [10], we restrict the formulae to be non-recursive definite clauses, in which query predicates only appear once. Under this constraint, the model is equivalent to a logistic regression model with a set of automatically generated features (i.e., extraction patterns). For complex formulae, which contain multiple query predicates and can be used for joint inference in MLN, we do not automatically generate them because these formulae are very general and the number of these patterns is very small, as we shall see in the experiments. We manually design these formulae and add them to the model in each iteration of StatSnowball.

To compare with, we also use a heuristic-based method in our experiments to select patterns. Specifically, we use the generated patterns and formulate a set of candidate formulas. Then, we apply some heuristic rules to select these formulas and learn the resultant model's weights. For example, a formula will be selected if the number of its covered instances is above a certain threshold (e.g., 10).

5. EXPERIMENTS

In this section, we report some empirical results of StatSnowball with different configurations. We compare StatSnowball with O-CRFs [4] for Open IE and show the advantages of joint inference by using MLN in StatSnowball. We also compare StatSnowball with Snowball on a large Web data corpus for traditional relation extraction. We show that

[4]http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Table 1: Types of keywords, their requirements, and examples in pattern generation

Types	Requirements	Example
VB	not stop word and occur more than MIN_OCCUR times	(e_1, *killed*, e_2)
IN	if the token appear more than MIN_OCCUR times and the previous token is a noun phrase	(e_1, mother *of*, e_2)
POS	the following token is a noun phrase	(e_1, *'s* mother, e_2)

by elegantly incorporating general patterns, StatSnowball achieves significantly higher precision and recall. Finally, StatSnowball is efficient and has been applied to Renlifang.

5.1 Data Sets

Our experiments are performed on two data sets. The first one is the published corpus [4] and will be referred to as *Sent500*. This data set contains 500 sentences and each sentence has one pair of entities (noun phrases). In [4], CRFs are used to do sequence labeling and identify the words that represent a relationship between the two entities. The second corpus is built from the MSN news crawler. To remove noise, such as page heads, navigation bars, etc., we first partition the crawled webpages into blocks using a visual parser [28]. The blocks in the center of a webpage are selected to compose our data set. All the text sentences in the blocks are parsed using a part-of-speech tagger to get the POS tagging results. We collect 1 million such blocks and will refer to this data set as *Web1M*.

5.2 Methods and Results

We report the experiments and results on two data sets separately as follows.

On *Sent500*, we perform two types of experiments. The first experiment is to demonstrate the advantages of MLNs over CRFs. In [4], CRFs are used to label the words between two entities (noun phrases) as a sequence labeling problem. In this experiment, we apply the MLN to do the similar sequence labeling task but with joint inference and show its advantages. The second experiment is to use StatSnowball to do Open IE and identify the unknown entity relationships. We compare StatSnowball with O-CRFs [4]. We also compare two StatSnowball systems with different pattern selection methods, i.e., the ℓ_1-norm regularized pattern selection and heuristic-based pattern selection. We will refer to these two StatSnowball systems as *ℓ_1StatSnowball* and *heStatSnowball*, respectively. As in [4], we evaluate on four categories of relations, that is, Verb, Noun+Prep, Verb+Prep, and Infinitive.

On *Web1M*, we configure StatSnowball to do the traditional relation extraction and compare it with Snowball.

5.2.1 Joint Inference with MLNs on Sent500

We compare MLNs and CRFs on Sent500 to do sequence labeling on sentences and label tokens to be REL-B, REL-C, or NULL. The set of features used in this experiment are similar to the features as used in O-CRFs [4], including POS tags, token relative position and context features. In CRFs, sentences are labeled independently. To do joint inference in MLN, we first group the sentences according to the similarity between their tokens. In total, 76 groups are identified in Sent500. For similar tokens (i.e., tokens in the same group), we define the following formula to do collective labeling:

$$\mathit{SimiToken}(t_1, t_2) \wedge F_1(t_1) \wedge F_2(t_2) \wedge \mathit{InField}(t_1, +f) \Rightarrow \mathit{InField}(t_2, +f),$$

Table 2: Evaluation results of MLNs with joint inference and the basic CRFs on the Sent500 data set

Categories		verb	noun+prep	verb+prep	infinitive	overall
CRF	*P*	**0.978**	0.751	0.836	**0.962**	0.882
	R	**0.776**	0.719	0.670	0.855	0.755
	F1	**0.865**	0.734	0.744	0.905	0.813
MLN	*P*	0.963	**0.922**	**0.870**	0.960	**0.929**
	R	0.768	**0.901**	**0.684**	**0.911**	**0.816**
	F1	0.854	**0.911**	**0.766**	**0.935**	**0.869**

where the predicate $SimiToken(t_i, t_j)$ is true if t_i is similar to t_j, false otherwise. Basically, this formula says that similar tokens (grouped together) should have the same label if they also have some token-level features F_i.

We will refer to the above two methods as *CRF* and *MLN* respectively. Table 2 shows the average results over 10 runs. In each run of the basic CRF, we randomly select 50 percent of the sentences as training and test on the rest (different from the experiments in [4]). For the joint MLN, we randomly select a half of the groups as training data and test on the rest. From the results, we can see that the joint inference in MLN performs much better, almost on all the four categories, than the CRF model, which treats the sentences independently. The performance of MLN on Verb is slightly worse than that of CRF.

5.2.2 StatSnowball on Sent500 for Open IE

As we have stated, StatSnowball can be configured to do Open IE. Here, we compare StatSnowball on the Sent500 with the state-of-the-art Open IE system, that is, O-CRFs [4]. All the patterns used in StatSnowball are the general POS tag sequences, except that we use the preposition keywords (like "in" and "of") to distinguish the tokens that are tagged as "IN" by the POS tagger. We randomly select 30 sentences as initial seeds for the two StatSnowball systems that use different pattern selection methods (i.e., ℓ_1-norm regularized and heuristic based methods). The results are shown in Table 3, where the results of O-CRFs are from the original paper [4], and the results of StatSnowball systems are the average results over 10 runs. For the ℓ_1StatSnowball, using different regularization constants (i.e., λ) during the iterations could improve the performance. In all the experiments, we do not tune the parameter but set λ at 0.5.

From the results, we can see that the O-CRFs generally achieve higher precisions, especially on "Verb+Prep" and "Infinitive". In contrast, StatSnowball systems achieve a better balance between recall and precision. Thus, the overall F1 of both StatSnowball systems are significantly better than that of the O-CRFs. Also, we can see that the heuristic-based pattern selection method works much worse than the ℓ_1-norm regularized pattern selection method in StatSnowball.

Figure 3 shows the performance of the two StatSnowball systems with respect to the number of iterations. The results are from one run. We do not take the average over 10

Table 3: Evaluation results of different models on the Sent500 data set

Categories		Verb	Noun+Prep	Verb+Prep	Infinitive	Overall
O-CRFs [4]	*Precision*	**0.939**	**0.891**	**0.952**	**0.957**	**0.883**
	Recall	0.651	0.360	0.500	**0.468**	0.452
	F1	0.769	0.513	**0.656**	**0.629**	0.598
heStatSnowball	*Precision*	0.956	0.704	0.583	0.456	0.790
	Recall	0.810	0.495	0.344	0.192	0.581
	F1	0.875	0.538	0.397	0.233	0.669
ℓ_1StatSnowball	*Precision*	0.929	0.815	0.617	0.418	0.798
	Recall	**0.846**	**0.782**	**0.575**	0.320	**0.733**
	F1	**0.886**	**0.799**	0.595	0.362	**0.764**

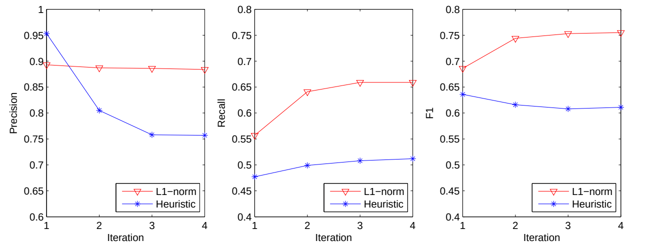

Figure 3: The performance (precision, recall, and F1) of the two StatSnowball systems with different pattern selection methods during the iteration.

runs because different runs can have various iteration numbers, e.g., from 4 to 10. We can see that during the iteration, both systems can find more relation tuples. Moreover, the precision of heStatSnowball decreases a lot, while the precision of ℓ_1StatSnowball does not change much, although with a slight decrease. Overall, the F1 curve of ℓ_1StatSnowball is consistently above that of heStatSnowball. Also, the former is increasing and the latter decreases a little.

5.2.3 Joint Inference in StatSnowball

Similar to the first experiment using MLN to do joint inference, we incorporate the joint inference to StatSnowball. We do not automatically generate these joint inference formulae. This is because the joint inference formulae are general and the number is very small (only 1 as shown in Section 5.2.1). Thus, it would be much harder to automatically generate these formulae and also it may cause difficulty in selecting them. Instead, we apply a simple strategy here. We manually define these joint formulae as in Section 5.2.1. During the iterations of StatSnowball, we automatically generate and select the simpler formulae, each of which contain only one query predicate, and add the joint formula to the resultant model to learn an MLN for joint inference.

For the joint StatSnowball, the sentences in Sent500 are grouped as in Section 5.2.1. We randomly select 10, 15, and 25 groups as starting seeds, and take the average over 10 runs as the final results. Table 4 shows the overall performance of the ℓ_1StatSnowball with joint inference, the heStatSnowball with joint inference, and the ℓ_1StatSnowball without joint inference.

From the results, we can see that the ℓ_1StatSnowball with joint inference always performs the best compared to the other two systems. Also, for all the three systems, the performance consistently gets better when more seeds are provided. Similar to Figure 3, Figure 4 shows the performance of the two StatSnowball systems with joint inference with

Table 4: Overall performance of different systems with different number of initial seeds.

# groups		10	15	25
ℓ_1StatSnowball (no joint inference)	*P*	0.735	0.806	0.806
	R	0.635	0.717	0.755
	F1	0.681	0.758	0.779
heStatSnowball (joint inference)	*P*	0.769	0.788	0.828
	R	0.691	0.696	0.748
	F1	0.728	0.739	0.786
ℓ_1StatSnowball (joint inference)	*P*	**0.822**	**0.816**	**0.873**
	R	**0.772**	**0.784**	**0.804**
	F1	**0.796**	**0.800**	**0.837**

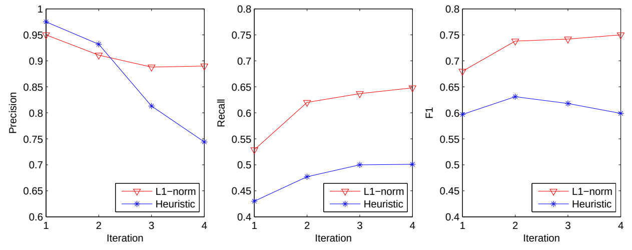

Figure 4: The performance (precision, recall, and F1) of the two StatSnowball systems with joint inference during the iteration.

respect to the number of iterations, and the results are from one of the ten runs. We can see that the ℓ_1-norm regularized pattern selection method consistently outperforms the heuristic-based method, especially on recall and F1.

5.2.4 Results on *Web1M*

Since for the large data corpus *Web1M*, labeling all the relations is impractical, it is difficult to quantitatively evaluate the Open IE as in [4]. To give a quantitative analysis of StatSnwoball, we configure it to extract predefined relations, as in the traditional use of Snowball. Here, we focus on the "Wife" and "Husband" relationships. We want to identify all the entity pairs (e_i, e_j), where e_i is the wife or husband of e_j.

Similar to the previous evaluations, the relation extraction is a sequence labeling problem. But in this task, we predict the tokens between each entity pair as either REL-R (i.e., a relation keyword) or NULL. Our extraction patterns are based on both the general POS tags and the strict keyword matching. For example, we use the POS tag sequence between the entity pairs as a candidate extraction pattern.

To get a high precision, in all the systems we will evaluate we only generate the patterns that contain at least one of the relation keywords provided with the initial seeds.

As we have stated, the MLN model in StatSnowball has the great flexibility to do joint inference. Here, we provide another example of using StatSnowball to perform joint relation extraction. In this experiment, the two types of relationships we are concerned with are strongly correlated. For example, if e_1 is the husband of e_2, then e_2 is more likely to be the wife of e_1 if we ignore the issue that multiple entities can have the same name. We incorporate this *prior* knowledge of dependency into the MLN model in StatSnowball by defining the following formula:

$$IsHusband(e_1, e_2) \Rightarrow IsWife(e_2, e_1).$$

Using this formula, the MLN can jointly extract the two types of relations. We refer to this system as *StatSnowball (Joint)* and refer to the StatSnowball without the above formula as *StatSnowball (Basic)*. Similar to the previous use of StatSnowball for joint inference (see Section 5.2.3), we only generate formulae that contain only one query predicate and manually add the above formula during the iterations.

The original Snowball system [1] uses strict keyword matching patterns. To see how the performance changes with general patterns, we configure another Snowball system with additional general patterns based on POS tags. We refer to this Snowball system as *PosSnowball*. The evaluation criteria of patterns and extraction tuples in PosSnowball are the same as those in the original Snowball.

To start the iteration process, *StatSnowball (Joint)* uses 30 seeds (15 wife seeds and 15 husband seeds). All the other systems perform the extraction of "Wife" and "Husband" separately with the corresponding seeds. All the extracted tuples are sent to human readers to judge whether they are correct extractions. Figure 5 shows the number of correct tuples and the precision of the identified tuples with respect to the number of iterations. From the results, we can see that StatSnwoball systems identify much more correct relation tuples with a significantly higher precision on all the identified tuples than the Snowball systems, especially the Snowball using only keyword-matching patterns. It is interesting to note that by using the simple joint inference formula as defined above in StatSnowball, we can find more accurate tuples, but with a slightly lower precision compared to the StatSnowball without this formula. The results of PosSnowball show that using general POS tag-based patterns can significantly improve the recall. Finally, during the iterations, although the number of correct tuples grows in Snowball systems, the precision decreases very quickly. In StatSnowball, however, we can get more correct tuples without sacrificing the high precision.

The StatSnowball is efficient. It takes about 50 minutes to finish the experiments on Web1M with a standard single-core desktop computer. That means 1 billion Web blocks can be processed by StatSnowball within 1 day by using 9 quad-core desktop computers. We have built an entity relation search engine, as shown in the introduction, which indexes the entities (people, locations, and organizations) and their relationships extracted from 10M Web blocks.

6. RELATED WORK

Relation extraction has been promoted by the Message Understanding Conference and Automatic Content Extraction program. The task has been traditionally studied as to extract *predefined* semantic relations between pairs of entities in text. The supervised methods [27, 8, 9, 26] require a set of human-tagged examples of the predefined relations. Bootstrapping methods [5, 1, 7] significantly reduce the number of training examples by iteratively discovering new extraction patterns and identifying entity relations with a small set of seeds, either target relation tuples [1] or general extraction templates [7]. However, the system [1] only generates patterns that are mainly based on keyword matching and its evaluation criteria are also specific to these strict high-precision but low-recall patterns. Another bootstrapping system—KnowItAll [7] requires large numbers of search engine queries and webpage downloads.

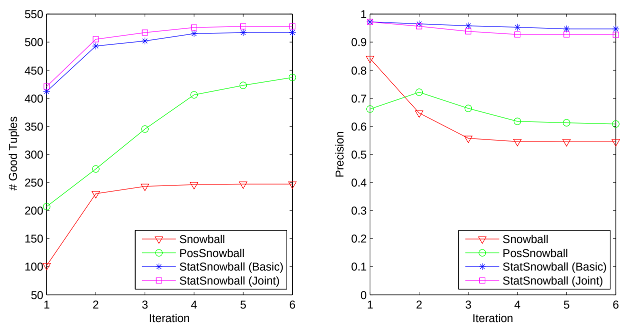

Figure 5: The number of correct relation tuples (left plot) and the precision of the extracted results by different methods.

Open information extraction (Open IE) [3] is a domain independent extraction paradigm and has been studied in both the natural language document corpus [22] and the Web environment [3] to extract relation tuples. Open IE can extract unknown relations from heterogeneous corpora. As we have stated, StatSnowball differs from the Open IE methods in several aspects. For example, Open IE systems require human-selected features to learn a good extractor, while StatSnowball automatically generates and selects the extraction patterns; Open IE systems require the use of deep linguistic parsing techniques to correctly label training samples, while StatSnwoball only uses cheaper and more robust shallow parsing techniques to generate its patterns; and the start-of-the-art Open IE system—O-CRFs [3] use CRFs to label each sentences independently, but StatSnowball can perform joint inference, which can yield better extraction results.

Pattern selection in StatSnowball is the problem of structure learning in Markov logic networks [12] or the problem of feature induction in Markov random fields [19, 16]. Similar to the feature induction of MRFs, where features are assumed to be composed from basic features, structure learning [12] is studied under the assumption that candidate formulas can be constructed from predicates via some operators like addition and flipping. Statistic predicate invention in Markov logic networks, also known as hidden variable discovery in statistical learning, is studied in [13], which can generate and select new predicates that are expressed in terms of existing ones via iterative clustering. Both structure learning and statistical predicate invention can be too expensive to be applied to a large data set. The discriminative structure learning of MLN [10] applies ℓ_1-norm regularized MLE to select candidate formulas generated by a first-order logic induction system. In StatSnowball, we generate the candidate formulas iteratively from extracted relation tuples.

7. CONCLUSIONS AND DISCUSSIONS

This paper presents a statistical entity relation extraction system called *StatSnowball*. By adopting a bootstrapping architecture, StatSnowball significantly reduces the number of human-tagged examples. By elegantly incorporating general extraction patterns, StatSnowball can significantly improve the recall and can be configured to perform open information extraction (Open IE). StatSnowball uses the general relational model—Markov logic networks (MLNs), which can be configured to perform different levels of relation extraction and can perform joint inference. Our empirical studies show that by using joint inference in MLN and StatSnowball, performance can be improved. Finally, by applying the ℓ_1-norm regularized maximum likelihood estimation, which enjoys well-founded theories and efficient solvers, StatSnowball is efficient and can perform well on large scale Web data. We have built an entity relationship search engine called *Renlifang* based on it.

The statistical StatSnowball opens broad ways for future improvements and extensions. Currently, we apply a named entity extraction method to process the data sets to get the entities. It is interesting to integrate StatSnowball with named entity extraction methods and build a self-contained extraction system. This tightly integrated approach allows information flow between two tasks and can obtain better performance by using joint inference [28, 17, 20]. Similarly, the optional P_3 part in StatSnowball can be integrated into P_2 as suggested by the relational clustering methods [14].

8. ACKNOWLEDGMENTS

The authors Jun and Bo are supported by Chinese Nature Science Foundation Grant 60621062 and 60605003; National Key Foundation R&D Projects 2003CB317007, 2004CB318108 and 2007CB311003; and Basic Research Foundation of Tsinghua National Lab for Info Sci & Tech. Jun is also supported by a Microsoft Fellowship.

9. REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *International Conference on Digital Libraries*, 2000.

[2] G. Andrew and J. Gao. Scalable training of l_1-regularized log-linear models. In *ICML*, 2007.

[3] M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[4] M. Banko and O. Etzioni. The tradeoffs between open and traditional relation extraction. In *ACL*, 2008.

[5] S. Brin. Extracting patterns and relations from the world wide web. In *International Workshop on the Web and Databases*, 1998.

[6] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learing*, 20:273–297, 1995.

[7] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

[8] C. Giuliano, A. Lavelli, and L. Romano. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *EACL*, 2006.

[9] A. Harabagiu, C. A. Bejan, and P. Morărescu. Shallow semantics for relation extraction. In *IJCAI*, 2005.

[10] T. N. Huynh and R. J. Mooney. Dsicriminative structure and parameter learning for markov logic networks. In *ICML*, 2008.

[11] A. Kaban. On Bayesian classification with laplace priors. *Pattern Recognition Letters*, 28(10):1271–1282, 2007.

[12] S. Kok and P. Domingos. Learning the structure of markov logic networks. In *ICML*, 2005.

[13] S. Kok and P. Domingos. Statistical predicate invention. In *ICML*, 2007.

[14] S. Kok and P. Domingos. Extracting semantic networks from text via relational clustering. In *ECML*, 2008.

[15] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[16] A. McCallum. Efficiently inducing features of conditional random fields. In *UAI*, 2003.

[17] A. McCallum and D. Jensen. A note on the unification of information extraction and data mining using conditional probability, relational models. In *IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, 2003.

[18] Z. Nie, J.-R. Wen, and W.-Y. Ma. Object-level vertical search. In *CIDR*, 2007.

[19] S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Trans. on PAMI*, 1997.

[20] H. Poon and P. Domingos. Joint inference in information extraction. In *AAAI*, 2007.

[21] M. Richardson and P. Domingos. Markov logic networks. *Machine Learing*, 62(1-2):107–136, 2006.

[22] Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *HLT/NAACL*, 2006.

[23] P. Singla and P. Domingos. Discriminative training of markov logic networks. In *AAAI*, 2005.

[24] C. H. Teo, Q. Le, A. Smola, and S. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *SIGKDD*, 2007.

[25] R. Tibshirani. Regression shrinkage and selection via the LASSO. *J. Royal. Statist. Soc.*, B(58):267–288, 1996.

[26] D. Zelenko, C. AoneE, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, (3):1083–1106, 2003.

[27] G. Zhou, M. Zhang, D. H. Ji, and Q. Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *EMNLP-CoNLL*, 2005.

[28] J. Zhu, Z. Nie, J.-R. Wen, B. Zhang, and W.-Y. Ma. Simultaneous record detection and attribute labeling in web data extraction. In *SIGKDD*, 2006.