Extracting Semantic Networks from Text Via Relational Clustering

Stanley Kok and Pedro Domingos

Department of Computer Science and Engineering
University of Washington, Seattle WA 98195-2350, USA
{koks,pedrod}@cs.washington.edu

Abstract. Extracting knowledge from text has long been a goal of AI. Initial approaches were purely logical and brittle. More recently, the availability of large quantities of text on the Web has led to the development of machine learning approaches. However, to date these have mainly extracted ground facts, as opposed to general knowledge. Other learning approaches can extract logical forms, but require supervision and do not scale. In this paper we present an unsupervised approach to extracting semantic networks from large volumes of text. We use the TextRunner system [1] to extract tuples from text, and then induce general concepts and relations from them by jointly clustering the objects and relational strings in the tuples. Our approach is defined in Markov logic using four simple rules. Experiments on a dataset of two million tuples show that it outperforms three other relational clustering approaches, and extracts meaningful semantic networks.

1 Introduction

A long-standing goal of AI is to build an autonomous agent that can read and understand text. The natural language processing (NLP) community attempted to achieve this goal in the 1970's and 1980's by building systems for understanding and answering questions about simple stories [3, 13, 23, 6]. These systems parsed text into a network of predefined concepts, and created a knowledge base from which inferences can be made. However, they required a large amount of manual engineering, only worked on small text sizes, and were not robust enough to perform well on unrestricted naturally occurring text. Gradually, research in this direction petered out.

Interest in the goal has been recently rekindled [16][7] by the abundance of easily accessible Web text, and by the substantial progress over the last few years in machine learning and NLP. The confluence of these three developments led to efforts to extract facts and knowledge bases from the Web [4]. Two recent steps in this direction are a system by Pasca et. al [18] and TextRunner [1]. Both systems extract facts on a large scale from Web corpora in an unsupervised manner. Pasca et. al's system derives relation-specific extraction patterns from a starting set of seed facts, acquires candidate facts using the patterns,

adds high-scoring facts to the seeds, and iterates until some convergence criterion. TextRunner uses a domain-independent approach to extract a large set of relational tuples of the form $r(x, y)$ where x and y are strings denoting objects, and r is a string denoting a relation between the objects. It uses a lightweight noun phrase chunker to identify objects, and heuristically determines the text between objects as relations. These are good first steps, but they still fall short of the goal. While they can quickly acquire a large database of ground facts in an unsupervised manner, they are not able to learn general knowledge that is embedded in the facts.

Another line of recent research takes the opposite approach. Semantic parsing [26, 17, 29] is the task of mapping a natural language sentence into logical form. The logical statements constitute a knowledge base that can be used to perform some task like answering questions. Semantic parsing systems require a training corpus of sentences annotated with their associated logical forms (i.e., they are supervised). These systems are then trained to induce a parser that can convert novel sentences to their logical forms. Even though these systems can create knowledge bases directly, their need for annotated training data prevents them from scaling to large corpora like the Web.

In this paper, we present SNE, a scalable, unsupervised, and domain-independent system that simultaneously extracts high-level relations and concepts, and learns a semantic network [20] from text. It first uses TextRunner to extract ground facts as triples from text, and then extract knowledge from the triples. TextRunner's triples are noisy, sparse, and contain many co-referent objects and relations. Our system has to overcome these challenges in order to extract meaningful high-level relations and concepts from the triples in an unsupervised manner. It does so with a probabilistic model that clusters objects by the objects that they are related to, and that clusters relations by the objects they relate. This allows information to propagate between clusters of relations and clusters of objects as they are created. Each cluster represents a high-level relation or concept. A concept cluster can be viewed as a node in a graph, and a relation cluster can be viewed as links between the concept clusters that it relates. Together the concept clusters and relation clusters define a simple semantic network. Figure 1 illustrates part of a semantic network that our approach learns. SNE is short for Semantic Network Extractor.

SNE is based on Markov logic [22], and is related to the Multiple Relational Clusterings (MRC) model [12] we recently proposed. SNE is our first step towards creating a system that can extract an arbitrary semantic network directly from text. Ultimately, we want to tightly integrate the information extraction TextRunner component and the knowledge learning SNE component to form a self-contained *knowledge extraction* system. This tight integration will enable information to flow between both tasks, allowing them to be solved jointly for better performance [14].

We begin by briefly reviewing Markov logic in the next section. Then we describe our model in detail (Section 3). Next we describe related work (Section 4). After that, we report our experiments comparing our model with three

alternative approaches (Section 5). We conclude with a discussion of future work (Section 6).

2 Markov Logic

Markov logic combines first-order logic with Markov networks.

In *first-order logic* [9], formulas are constructed using four types of symbols: constants, variables, functions, and predicates. (In this paper we use only function-free logic.) Constants represent objects in the domain of discourse (e.g., people: `Anna`, `Bob`, etc.). Variables (e.g., `x`, `y`) range over the objects in the domain. Predicates represent relations among objects (e.g., `Friends`), or attributes of objects (e.g., `Student`). Variables and constants may be typed. An *atom* is a predicate symbol applied to a list of arguments, which may be variables or constants (e.g., `Friends(Anna, x)`). A *ground atom* is an atom all of whose arguments are constants (e.g., `Friends(Anna, Bob)`). A *world* is an assignment of truth values to all possible ground atoms. A database is a partial specification of a world; each atom in it is true, false or (implicitly) unknown.

A *Markov network* or *Markov random field* [19] is a model for the joint distribution of a set of variables $X = (X_1, X_2, \ldots, X_n) \in \mathcal{X}$. It is composed of an undirected graph G and a set of potential functions ϕ_k. The graph has a node for each variable, and the model has a potential function for each clique in the graph. A potential function is a non-negative real-valued function of the state of the corresponding clique. The joint distribution represented by a Markov network is given by $P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$ where $x_{\{k\}}$ is the state of the kth clique (i.e., the state of the variables that appear in that clique). Z, known as the *partition function*, is given by $Z = \sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$. Markov networks are often conveniently represented as *log-linear models*, with each clique potential replaced by an exponentiated weighted sum of features of the state, leading to $P(X = x) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x)\right)$. A feature may be any real-valued function of the state. This paper will focus on binary features, $f_j(x) \in \{0, 1\}$. In the most direct translation from the potential-function form, there is one feature corresponding to each possible state $x_{\{k\}}$ of each clique, with its weight being $\log \phi_k(x_{\{k\}})$. This representation is exponential in the size of the cliques. However, we are free to specify a much smaller number of features (e.g., logical functions of the state of the clique), allowing for a more compact representation than the potential-function form, particularly when large cliques are present. Markov logic takes advantage of this.

A *Markov logic network (MLN)* is a set of weighted first-order formulas. Together with a set of constants representing objects in the domain, it defines a Markov network with one node per ground atom and one feature per ground formula. The weight of a feature is the weight of the first-order formula that originated it. The probability distribution over possible worlds x specified by the ground Markov network is given by $P(X = x) = \frac{1}{Z} \exp\left(\sum_{i \in F} \sum_{j \in G_i} w_i g_j(x)\right)$, where Z is the partition function, F is the set of all first-order formulas in the

MLN, G_i is the set of groundings of the ith first-order formula, and $g_j(x) = 1$ if the jth ground formula is true and $g_j(x) = 0$ otherwise. Markov logic enables us to compactly represent complex models in non-i.i.d. domains. General algorithms for inference and learning in Markov logic are discussed in [22].

3 Semantic Network Extraction

We call our model SNE, for Semantic Network Extractor. SNE simultaneously clusters objects and relations in an unsupervised manner, without requiring the number of clusters to be specified in advance. The object clusters and relation clusters respectively form the nodes and links of a semantic network. A link exists between two nodes if and only if a true ground fact can be formed from the symbols in the corresponding relation and object clusters. SNE can cluster objects of different types, and relations of any arity.

When faced with the task of extracting knowledge from noisy and sparse data like that used in our experiments, we have to glean every bit of useful information from the data to form coherent clusters. SNE does this by jointly clustering objects and relations. In its algorithm, SNE allows information from object clusters it has created at each step to be used in forming relation clusters, and vice versa. As we shall see later in our experimental results, this joint clustering approach does better than clustering objects and relations separately.

SNE is defined using a form of finite second-order Markov logic in which variables can range over relations (predicates) as well as objects (constants). Extending Markov logic to second order involves simply grounding atoms with all possible predicate symbols as well as all constant symbols, and allows us to represent some models much more compactly than first-order Markov logic.

For simplicity, we assume that relations are binary in our definition of SNE, i.e., relations are of the form $r(x, y)$ where r is a relation symbol, and x and y are object symbols. (Extending the definition to an arbitrary number of n-ary relations is straightforward.) We use γ_i and Γ_i to respectively denote a cluster and clustering (i.e., a partitioning) of symbols of type i. If r, x, and y are respectively in cluster γ_r, γ_x, and γ_y, we say that $r(x, y)$ is in the *cluster combination* $(\gamma_r, \gamma_x, \gamma_y)$.

The learning problem in SNE consists of finding the cluster assignment $\Gamma = (\Gamma_r, \Gamma_x, \Gamma_y)$ that maximizes the posterior probability $P(\Gamma|R) \propto P(\Gamma, R) = P(\Gamma)P(R|\Gamma)$, where R is a vector of truth assignments to the observable $r(x, y)$ ground atoms.

We define one MLN for the likelihood $P(R|\Gamma)$ component, and one MLN for the prior $P(\Gamma)$ component of the posterior probability with just four simple rules.

The MLN for the likelihood component only contains one rule stating that the truth value of an atom is determined by the cluster combination it belongs to:

$$\forall r, x, y, +\gamma_r, +\gamma_x, +\gamma_y \quad r \in \gamma_r \wedge x \in \gamma_x \wedge y \in \gamma_y \Rightarrow r(x, y)$$

This rule is soft. The "+" notation is syntactic sugar that signifies that there is an instance of this rule *with a separate weight* for each cluster combination $(\gamma_r, \gamma_x, \gamma_y)$. This rule predicts the probability of query atoms given the cluster memberships of the symbols in them. This is known as the *atom prediction* rule. As shown in [12], given a cluster assignment, the MAP weight w_k of an instance of the atom prediction rule is given by $\log(t_k/f_k)$, where t_k is the empirical number of true atoms in cluster combination k, and f_k is the number of false atoms. Adding smoothing parameters α and β, we estimate the MAP weight as $\log((t_k + \alpha)/(f_k + \beta))$.

Three rules are defined in the MLN for the prior component. The first rule states that each symbol belongs to exactly one cluster:

$$\forall x \; \exists^1 \gamma \; x \in \gamma$$

This rule is hard, i.e., it has infinite weight and cannot be violated.

The second rule imposes an exponential prior on the number of cluster combinations. This rule combats the proliferation of cluster combinations and consequent overfitting, and is represented by the formula

$$\forall \gamma_r, \gamma_x, \gamma_y \; \exists r, x, y \quad r \in \gamma_r \wedge x \in \gamma_x \wedge y \in \gamma_y$$

with negative weight $-\lambda$. The parameter λ is fixed during learning, and is the penalty in log-posterior incurred by adding a cluster combination to the model. Thus larger λs lead to fewer cluster combinations being formed. This rule represents the complexity of the model in terms of the number of instances of the atom prediction rule (which is equal to the number of cluster combinations).

The last rule encodes the belief that most symbols tend to be in different clusters. It is represented by the formula

$$\forall x, x', \gamma_x, \gamma_{x'} \;\; x \in \gamma_x \wedge x' \in \gamma_{x'} \wedge x \neq x' \Rightarrow \gamma_x \neq \gamma_{x'}$$

with positive weight μ. The parameter μ is also fixed during learning. We expect there to be many concepts and high-level relations in a large heterogenous body of text. The tuple extraction process samples instances of these concepts and relations sparsely, and we expect each concept or relation to have only a few instances sampled, in many cases only one. Thus we expect most pairs of symbols to be in different concept and relation clusters.

The equation for the log-posterior, as defined by the two MLNs, can be written in closed form as [1]

$$\log P(\Gamma|R) = \sum_{k \in K} \left[t_k \log \left(\frac{t_k + \alpha}{t_k + f_k + \alpha + \beta} \right) + f_k \log \left(\frac{f_k + \beta}{t_k + f_k + \alpha + \beta} \right) \right] - \lambda m_{cc} + \mu d + \mathcal{C} \quad (1)$$

[1] The derivation of the log-posterior is given in an online appendix at http://alchemy.cs.washington.edu/papers/kok08.

where K is the set of cluster combinations, m_{cc} is the number of cluster combinations, d is the number of pairs of symbols that belong to different clusters, and $\mathcal{C}$ is a constant.

Rewriting the equation, the log-posterior can be expressed as

$$\log P(\Gamma|R) = \sum_{k \in K^+} \left[t_k \log \left(\frac{t_k + \alpha}{t_k + f_k + \alpha + \beta} \right) + f_k \log \left(\frac{f_k + \beta}{t_k + f_k + \alpha + \beta} \right) \right] + \sum_{k \in K^-} \left[f_k \log \left(\frac{f_k + \beta}{t_k + f_k + \alpha + \beta} \right) \right] - \lambda m_{cc} + \mu d + \mathcal{C} \quad (2)$$

where K^+ is the set of cluster combinations that contains at least one true ground atom, and K^- is the set of cluster combinations that does not contain any true ground atoms. Observe that $|K^+| + |K^-| = |\Gamma_r||\Gamma_x||\Gamma_y|$. Even though it is tractable to compute the first summation over $|K^+|$ (which is at most the number of true ground atoms), it may not be feasible to compute the second summation over $|K^-|$ for large $|\Gamma_i|$s. Hence, for tractability, we assume that all tuples in K^- belong to a single 'default' cluster combination with the same probability p_{false} of being false. The log-posterior is simplified as

$$\log P(\Gamma|R) = \sum_{k \in K^+} \left[t_k \log \left(\frac{t_k + \alpha}{t_k + f_k + \alpha + \beta} \right) + f_k \log \left(\frac{f_k + \beta}{t_k + f_k + \alpha + \beta} \right) \right] + \left(|S_r||S_x||S_y| - \sum_{k \in K^+} (t_k + f_k) \right) \log(p_{false}) - \lambda m_{cc}^+ + \mu d + \mathcal{C}' \quad (3)$$

where S_i is the set of symbols of type i, $(|S_r||S_x||S_y| - \sum_{k \in K^+}(t_k + f_k))$ is the number of (false) tuples in K^-, m_{cc}^+ is the number of cluster combinations containing at least one true ground atom, and $\mathcal{C}' = \mathcal{C} - \lambda$.

SNE simplifies the learning problem by performing hard assignment of symbols to clusters (i.e., instead of computing probabilities of cluster membership, a symbol is simply assigned to its most likely cluster). Since, given a cluster assignment, the MAP weights can be computed in closed form, SNE simply searches over cluster assignments, evaluating each assignment by its posterior probability.

SNE uses a bottom-up agglomerative clustering algorithm to find the MAP clustering (Table 1). The algorithm begins by assigning each symbol to its own unit cluster. Next we try to merge pairs of clusters of each type. We create candidate pairs of clusters, and for each of them, we evaluate the change in posterior probability (Equation 3) if the pair is merged. If the candidate pair improves posterior probability, we store it in a sorted list. We then iterate through the list, performing the best merges first, and ignoring those containing clusters that have already been merged. In this manner, we incrementally merge clusters until no merges can be performed to improve posterior probability.

To avoid creating all possible candidate pairs of clusters of each type (which is quadratic in the number of clusters), we make use of canopies [15]. A canopy for relation symbols is a set of clusters such that there exist object clusters γ_x and γ_y,

Table 1. The SNE algorithm.

function $SNE(S_r, S_x, S_y, R)$
  inputs: S_r, set of relation symbols
    S_x, set of object symbols that appear as first arguments
    S_y, set of object symbols that appear as second arguments
    R, ground $r(x, y)$ atoms formed from the symbols in S_r, S_x, and S_y
  output: a semantic network, $\{(\gamma_r, \gamma_x, \gamma_y) \in \Gamma_r \times \Gamma_x \times \Gamma_y : (\gamma_r, \gamma_x, \gamma_y)$ contains at least one true ground atom$\}$
for each $i \in \{r, x, y\}$
  $\Gamma_i \leftarrow unitClusters(S_i)$
$mergeOccurred \leftarrow true$
while $mergeOccurred$
  $mergeOccurred \leftarrow false$
  for each $i \in \{r, x, y\}$
    $CandidateMerges \leftarrow \emptyset$
    for each $(\gamma, \gamma') \in \Gamma_i \times \Gamma_i$
      $\Delta P \leftarrow$ change in $P(\{\Gamma_r, \Gamma_x, \Gamma_y\}|R)$ if γ, γ' are merged
      if $\Delta P > 0$, $CandidateMerges \leftarrow CandidateMerges \cup \{(\gamma, \gamma')\}$
    sort $CandidateMerges$ in descending order of ΔP
    $MergedClusters \leftarrow \emptyset$
    for each $(\gamma, \gamma') \in CandidateMerges$
      if $\gamma \notin MergedClusters$ and $\gamma' \notin MergedClusters$
        $\Gamma_i \leftarrow (\Gamma_i \setminus \{\gamma, \gamma'\}) \cup \{\gamma \cup \gamma'\}$
        $MergedClusters \leftarrow MergedClusters \cup \{\gamma\} \cup \{\gamma'\}$
        $mergedOccurred \leftarrow true$
return $\{(\gamma_r, \gamma_x, \gamma_y) \in \Gamma_r \times \Gamma_x \times \Gamma_y : (\gamma_r, \gamma_x, \gamma_y)$ contains at least one true ground atom$\}$

and for all clusters γ_r in the canopy, the cluster combination $(\gamma_r, \gamma_x, \gamma_y)$ contains at least one true ground atom $r(x, y)$. We say that the clusters in the canopy share the *property* (γ_x, γ_y). Canopies for object symbols x and y are similarly defined. We only try to merge clusters in a canopy that is no larger than a parameter $CanopyMax$. This parameter limits the number of candidate cluster pairs we consider for merges, making our algorithm more tractable. Furthermore, by using canopies, we only try 'good' merges, because symbols in clusters that share a property are more likely to belong to the same cluster than those in clusters with no property in common.

Note that we can efficiently compute the change in posterior probability (ΔP in Table 1) by only considering the cluster combinations with true ground atoms that contain the merged clusters γ and γ'. Below we give the equation for computing ΔP when we merge relation clusters γ_r and γ'_r to form γ''_r. The equations for merging object clusters are similar. Let TF_k be a shorthand for $t_k \log(\frac{t_k+\alpha}{t_k+f_k+\alpha+\beta}) + f_k \log(\frac{f_k+\beta}{t_k+f_k+\alpha+\beta})$.

$$\Delta P = \sum_{(\gamma''_r, \gamma_1, \gamma_2) \in K^+_{\gamma''_r \gamma'_r \gamma_r}} \left[TF_{(\gamma''_r, \gamma_1, \gamma_2)} - TF_{(\gamma'_r, \gamma_1, \gamma_2)} - TF_{(\gamma_r, \gamma_1, \gamma_2)} + \lambda \right]$$
$$+ \sum_{(\gamma''_r, \gamma_1, \gamma_2) \in K^+_{\gamma''_r \cdot \gamma_r}} \left[TF_{(\gamma''_r, \gamma_1, \gamma_2)} - f_{(\gamma'_r, \gamma_1, \gamma_2)} \log(p_{false}) - TF_{(\gamma_r, \gamma_1, \gamma_2)} \right]$$

$$+ \sum_{(\gamma_r^{''},\gamma_1,\gamma_2)\in K^{+}_{\gamma_r^{''}\gamma_{r\cdot}^{'}}} \left[TF_{(\gamma_r^{''},\gamma_1,\gamma_2)} - TF_{(\gamma_r^{'},\gamma_1,\gamma_2)} - f_{(\gamma_r,\gamma_1,\gamma_2)}\log(p_{false})\right]$$
$$-\mu|{\gamma_r}'||\gamma_r| \quad (4)$$

where $K^{+}_{\gamma_r^{''}\gamma_r^{'}\gamma_r}$ is the set of cluster combinations with true ground atoms such that each cluster combination $(\gamma_r^{''},\gamma_1,\gamma_2)$ in the set has the property that $(\gamma_r^{'},\gamma_1,\gamma_2)$ and $(\gamma_r,\gamma_1,\gamma_2)$ also contains true atoms. $K^{+}_{\gamma_r^{''}\cdot\gamma_r}$ is the set of cluster combinations with true ground atoms such that each cluster combination $(\gamma_r^{''},\gamma_1,\gamma_2)$ in the set has the property that $(\gamma_r,\gamma_1,\gamma_2)$, but not $(\gamma_r^{'},\gamma_1,\gamma_2)$, contains true ground atoms. $K^{+}_{\gamma_r^{''}\gamma_{r\cdot}^{'}}$ is similarly defined. Observe that we only sum over cluster combinations with true ground atoms that contains the affected clusters γ_r, $\gamma_r^{'}$ and $\gamma_r^{''}$, rather than over all cluster combinations with true ground atoms.

4 Related Work

Rajaraman and Tan [21] propose a system that learns a semantic network by clustering objects but not relations. While it anecdotally shows a snippet of its semantic network, an empirical evaluation of the network is not reported. Hasegawa et. al [10] propose an unsupervised approach to discover relations from text. They treat the short text segment between each pair of objects as a relation, and cluster pairs of objects using the similarity between their relation strings. Each cluster corresponds to a relation, and a pair of objects can appear in at most one cluster (relation). In contrast, SNE allows a pair of objects to participate in multiple relations (semantic statements). Shinyama and Sekine [25] form (possibly overlapping) clusters of tuples of objects (rather than just pairs of objects). They use the words surrounding the objects in the same sentence to form a pattern. Objects in sentences with the same pattern are deemed to be related in the same way, and are clustered together. All three previous systems are not domain-independent because they rely on name entity (NE) taggers to identify objects in text. The concepts and relations that they learn are restricted by the object types that can be identified with the NE taggers. All three systems also use ad-hoc techniques that do not give a probability distribution over possible worlds, which we need in order to perform inference and answer queries. By only forming clusters of (tuples of) objects, and not relations, they do not explicitly learn high-level relations like SNE.

ALICE [2] is a system for lifelong knowledge extraction from a Web corpus. Like SNE, it uses TextRunner's triples as input. However, unlike SNE, it requires background knowledge in the form of an existing domain-specific concept taxonomy, and does not cluster relations into higher level ones.

RESOLVER [28] is a system that takes TextRunner's triples as input, and resolves references to the same object and relations by clustering the references together (e.g., `Red_Planet` and `Mars` are clustered together). In contrast, SNE learns abstract concepts and relations (e.g., `Mars`, `Venus`, `Earth`, etc. are clus-

tered together to form the concept of 'planet'). Unlike SNE, RESOLVER's probabilistic model clusters objects and relations separately rather than jointly. To allow information to propagate between object clusters and relation clusters, RESOLVER uses an ad-hoc approach. In its experiments, RESOLVER gives similar results with or without the ad-hoc approach. In contrast, we show in our experiments that SNE gives better performance with joint rather than separate clustering (see Table 3). In a preliminary experiment where we adapt SNE to only use string similarities between objects (and relations), we find that SNE performs better than RESOLVER on an entity resolution task on the dataset described in Section 5.

5 Experiments

Our goal is to create a system that is capable of extracting semantic networks from what is arguably the largest and most accessible text resource — the Web. Thus in our experiments, we use a large Web corpus to evaluate the effectiveness of SNE's relational clustering approach in extracting a simple semantic network from it. Since to date, no other system could do the same, we had to modify three other relational clustering approaches so that they could run on our large Web-scale dataset, and compared SNE to them. The three approaches are Multiple Relational Clusterings [12], Information-Theoretic Co-clustering [5], and the Infinite Relational Model [11].

5.1 Multiple Relational Clusterings

Like SNE, MRC is a model that simultaneously clusters objects and relations without requiring the number of clusters to be specified in advance. However, unlike SNE, MRC is able to find multiple clusterings, rather than just one. MRC is also defined using finite second-order Markov logic. The main difference between SNE and MRC is in the search algorithm used. MRC also differs from SNE in having an exponential prior on the number of clusters rather than on the number of cluster combinations with true ground atoms. MRC calls itself recursively to find multiple clusterings. We can view MRC as growing a tree of clusterings, and it returns the finest clusterings at the leaves. In each recursive call, MRC uses a top-down generate-and-test greedy algorithm with restarts to find the MAP clustering of the subset of relation and constant symbols it received. While this 'blind' generate-and-test approach may work well for small datasets, it will not be feasible for large Web-scale datasets like the one used in our experiments. For such large datasets, the search space will be so enormous that the top-down algorithm will generate too many candidate moves to be tractable. In our experiments, we replaced MRC's search algorithm with the algorithm in Table 1. We use MRC1 to denote an MRC model that is restricted to find a single clustering.

5.2 Information-Theoretic Co-clustering

The ITC model [5] clusters discrete data in a two-dimensional matrix along both dimensions simultaneously. It greedily searches for the hard clusterings that optimize the mutual information between the row and column clusters. The model has been shown to perform well on noisy and sparse data. ITC's top-down search algorithm has the flavor of K-means, and requires the number of row and column clusters to be specified in advance. At every step, ITC finds the best cluster for each row or column by iterating through all clusters. This will not be tractable for large datasets like our Web dataset, which can contain many clusters. Thus, we instead use the algorithm in Table 1 (ΔP in Table 1 is set to the change in mutual information rather than the change in log-posterior probability). Notice that, even if ITC's search algorithm were tractable, we would not be able to apply it to our problem because it only works on two-dimensional data. We extend ITC to three dimensions by optimizing the mutual information among the clusters of three dimensions. Furthermore, since we do not know the exact number of clusters in our Web dataset a priori, we follow [5]'s suggestion of using an information-theoretic prior (BIC [24]) to select the appropriate number of clusters. We use ITC-C and ITC-CC to respectively denote the model with a BIC prior on clusters, and the model with a BIC prior on cluster combinations. Note that, unlike SNE, ITC does not give a probability distribution over possible worlds, which we need in order to do inference and answer queries (although that is not the focus of this paper).

5.3 Infinite Relational Model

Like SNE, the IRM [11] is a model that simultaneously clusters objects and relations without requiring the number of clusters to be specified in advance. It defines a generative model for the predicates and cluster assignments. It assumes that the predicates are conditionally independent given the cluster assignments, and the cluster assignments for each type are independent. IRM uses a Chinese restaurant process (CRP) prior on the cluster assignments. Under the CRP, each new object is assigned to an existing cluster with probability proportional to the cluster size. IRM assumes that the probability p of an atom being true conditioned on cluster membership is generated according to a Beta distribution, and that the truth values of atoms are then generated according to a Bernoulli distribution with parameter p. IRM finds the MAP cluster assignment using a top-down search similar to MRC, except that it also searches for the optimal values of its CRP and Beta parameters. As mentioned earlier, top-down search is not feasible for large Web-scale data, so we replace IRM's search algorithm with the one in Table 1. We also fixed the values of the CRP and Beta parameters. As in SNE, we assumed that the atoms in cluster combinations with only false atoms belonged to a default cluster combination, and they had the same probability p_{false} of being false. We also experimented with a CRP prior on cluster combinations. We use IRM-C and IRM-CC to respectively denote the IRM with a CRP prior on clusters, and the IRM with a CRP prior on cluster combinations. Xu et al. [27] proposed a model closely related to the IRM.

5.4 Dataset

We compared the various models on a dataset of about 2.1 million triples[2] extracted in a Web crawl by TextRunner [1]. Each triple takes the form $r(x, y)$ where r is a relation symbol, and x and y are object symbols. Some example triples are: `named_after`(`Jupiter`, `Roman_god`) and `upheld`(`Court`, `ruling`). There are 15,872 distinct r symbols, 700,781 distinct x symbols, and 665,378 distinct y symbols. Two characteristics of TextRunner's extractions are that they are sparse and noisy. To reduce the noise in the dataset, our search algorithm (Table 1) only considered symbols that appeared at least 25 times. This leaves 10,214 r symbols, 8942 x symbols, and 7995 y symbols. There are 2,065,045 triples that contain at least one symbol that appears at least 25 times. In all experiments, we set the $CanopyMax$ parameter to 50. We make the closed-world assumption for all models (i.e., all triples not in the dataset are assumed false).

5.5 SNE vs. MRC

We compared the performances of SNE and MRC1 in learning a *single* clustering of symbols. We set the λ, μ and p_{false} parameters in SNE to 100, 100 and 0.9999 respectively based on preliminary experiments. We set SNE's α and β parameters to 2.81×10^{-9} and $10 - \alpha$ so that $\frac{\alpha}{\alpha+\beta}$ is equal to the fraction of true triples in the dataset. (A priori, we should predict the probability that a ground atom is true to be this value.) We evaluated the clusterings learned by each model against a gold standard manually created by the first author. The gold standard assigns 2688 r symbols, 2568 x symbols, and 3058 y symbols to 874, 511, and 700 non-unit clusters respectively. We measured the pairwise precision, recall and F1 of each model against the gold standard. Pairwise precision is the fraction of symbol pairs in learned clusters that appear in the same gold clusters. Pairwise recall is the fraction of symbol pairs in gold clusters that appear in the same learned clusters. F1 is the harmonic mean of precision and recall. For the weight of MRC1's exponential prior on clusters, we tried the following values and pick the best: 0, 1, 10–100 (in increments of 10), and 110–1000 (in increments of 100). We report the precision, recall and F1 scores that are obtained with the best value of 80. From Table 2, we see that SNE performs significantly better than MRC1.

We also ran MRC to find multiple clusterings. Since the gold standard only defines a single clustering, we cannot use it to evaluate the multiple clusterings. We provide a qualitative evaluation instead. MRC returns 23,151 leaves that contain non-unit clusters, and 99.8% of these only contain 3 or fewer clusters of size 2. In contrast, SNE finds many clusters of varying sizes (see Table 6). The poor performance of MRC in finding multiple clusterings is due to data sparsity. In each recursive call to MRC, it only receives a small subset of the relation and object symbols. Thus with each call the data becomes sparser, and there is not enough signal to cluster the symbols.

[2] Publicly available at http://knight.cis.temple.edu/~yates/data/resolver_data.tar.gz

Table 2. Comparison of SNE and MRC1 performances on gold standard. Object 1 and Object 2 respectively refer to the object symbols that appear as the first and second arguments of relations. The best F1s are shown in bold.

	Relation			**Object 1**			**Object 2**		
Model	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
SNE	0.452	0.187	**0.265**	0.460	0.061	**0.108**	0.558	0.062	**0.112**
MRC1	0.054	0.044	0.049	0.031	0.007	0.012	0.059	0.011	0.018

Table 3. Comparison of SNE performance when it clusters relation and object symbols jointly and separately. SNE-Sep clusters relation and object symbols separately. Object 1 and Object 2 respectively refer to the object symbols that appear as the first and second arguments of relations. The best F1s are shown in bold.

	Relation			**Object 1**			**Object 2**		
Model	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
SNE	0.452	0.187	**0.265**	0.460	0.061	**0.108**	0.558	0.062	**0.112**
SNE-Sep	0.597	0.116	0.194	0.519	0.045	0.083	0.551	0.047	0.086

5.6 Joint vs. Separate Clustering of Relations and Objects

We investigated the effect of having SNE only cluster relation symbols, first-argument object symbols, or second-argument object symbols, e.g., if SNE cluster relation symbols, then it does not cluster both kinds of object symbols. From Table 3, we see that SNE obtains a significantly higher F1 when it clusters relations and objects jointly than when it clusters them separately.

5.7 SNE vs. IRM and ITC

We compared IRM-C and IRM-CC with respect to the gold standard. We set IRM's Beta parameters to the values of SNE's α and β, and set p_{false} to the same value as SNE's. We tried the following values for the parameter of the CRP priors: 0.25, 0.5, 0.75, 1–10 (in increments of 1), 20–100 (in increments of 10). We found that the graphs showing how precision, recall, and F1 vary with the CRP value are essentially flat for both IRM-C and IRM-CC. Both system perform about the same. The slightly higher precision, recall, and F1 scores occur at the low end of the values we tried, and we use the best one of 0.25 for the slightly better-performing IRM-CC system. Henceforth, we denote this IRM as IRM-CC-0.25, and use it for other comparisons.

We also compared SNE, IRM-CC-0.25, ITC-C, and ITC-CC. From Table 4, we see that ITC performs better with a BIC prior on cluster combinations than a BIC prior on clusters. We also see that SNE performs the best in terms of F1.

We then evaluated SNE, IRM-CC-0.25 and ITC-CC in terms of the semantic statements that they learned. A cluster combination that contains a true ground atom corresponds to a semantic statement. SNE, IRM-CC-0.25 and ITC-CC

Table 4. Comparison of SNE, IRM-CC-0.25, ITC-CC, and ITC-C performances on gold standard. Object 1 and Object 2 respectively refer to the object symbols that appear as the first and second arguments of relations. The best F1s are shown in bold.

	Relation			**Object 1**			**Object 2**		
Model	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
SNE	0.452	0.187	**0.265**	0.461	0.061	**0.108**	0.558	0.062	**0.112**
IRM-CC-0.25	0.201	0.089	0.124	0.252	0.043	0.073	0.307	0.041	0.072
ITC-CC	0.773	0.003	0.006	0.470	0.047	0.085	0.764	0.002	0.004
ITC-C	0.000	0.000	0.000	0.571	0.000	0.000	0.333	0.000	0.000

Table 5. Evaluation of semantic statements learned by SNE, IRM-CC-0.25, and ITC-CC.

Model	**Total Statements**	**Num. Correct**	**Fract. Correct**
SNE	1241	965	0.778
IRM-CC-0.25	487	426	0.874
ITC-CC	310	259	0.835

respectively learned 1,464,965, 1,254,995 and 82,609 semantic statements. We manually inspected semantic statements containing 5 or more true ground atoms, and counted the number that were correct. Table 5 shows the results. Even though SNE's accuracy is smaller than IRM-CC-0.25's and ITC-CC's by 11% and 7% respectively, SNE more than compensates for the lower accuracy by learning 127% and 273% more correct statements respectively. Figure 1 shows examples of correct semantic statements learned by SNE.

SNE, IRM-CC-0.25 and ITC-CC respectively ran for about 5.5 hours, 9.5 hours, and 3 days on identically configured machines. ITC-CC spent most of its time computing the mutual information among three clusters. To compute the mutual information, given any two clusters, we have to retrieve the number of cluster combinations that contain the two clusters. Because of the large number of cluster pairs, we choose to use a data structure (red-black tree) that is space-efficient, but pays a time penalty when looking up the required values.

5.8 Comparison of SNE with WordNet

We also compared the object clusters that SNE learned with WordNet [8], a hand-built semantic lexicon for the English language. WordNet organizes 117,798 distinct nouns into a taxonomy of 82,115 concepts. There are respectively 4883 first-argument, and 5076 second-argument object symbols that appear at least 25 times in our dataset, and also in WordNet. We converted each node (synset) in WordNet's taxonomy into a cluster containing its original concepts, and all its children concepts. We then matched each SNE cluster to the WordNet cluster that gave the best F1 score. We measured F1 as the harmonic mean of precision

Table 6. Comparison of SNE object clusters with WordNet.

Cluster Size	Num. Clusters	Level	Prec.	Recall	F1
47	1	7.0±0.0	0.8±0.0	0.2±0.0	0.4±0.0
36	1	8.0±0.0	0.3±0.0	0.3±0.0	0.3±0.0
24	1	6.0±0.0	0.2±0.0	0.3±0.0	0.2±0.0
19	1	7.0±0.0	0.2±0.0	0.3±0.0	0.2±0.0
16	1	7.0±0.0	0.3±0.0	0.3±0.0	0.3±0.0
12	3	7.0±0.7	0.5±0.1	0.7±0.1	0.5±0.2
11	1	6.0±0.0	0.9±0.0	0.7±0.0	0.8±0.0
10	2	5.5±0.7	0.6±0.1	0.9±0.1	0.5±0.1
8	5	7.0±0.9	0.4±0.2	0.7±0.4	0.3±0.1
7	4	6.0±1.4	0.7±0.3	0.8±0.2	0.9±0.1
6	12	6.6±1.7	0.4±0.2	0.6±0.2	0.6±0.2
5	12	7.2±1.6	0.4±0.2	0.5±0.3	0.7±0.1
4	84	7.2±1.7	0.4±0.1	0.7±0.2	0.6±0.2
3	185	7.3±1.8	0.5±0.2	0.7±0.2	0.7±0.2
2	1419	7.2±1.8	0.6±0.1	0.7±0.1	0.8±0.1

and recall. Precision is the fraction of symbols in an SNE cluster that is also in the matched WordNet cluster. Recall is the fraction of symbols in a WordNet cluster that is also in the corresponding SNE cluster. Table 6 shows how precision, recall, and F1 vary with cluster sizes. (The scores are averaged over all object clusters of the same size). We see that the F1s are fairly good for object clusters of size 7 or less. The table also shows how the level of the matched cluster in WordNet's taxonomy vary with cluster size. The higher the level, the more specifc the concept represented by the matched WordNet cluster. For example, clusters at level 7 correspond to specific concepts like 'country', 'state', 'dwelling', and 'home', while the single cluster at level 0 (i.e., at the root of the taxonomy) corresponds to 'all entities'. We see that the object clusters correspond to fairly specifc concepts in WordNet. We did not compare the relation clusters to WordNet's verbs because the overlap between the relation symbols and the verbs are too small.

6 Conclusion and Future Work

We presented SNE, a scalable, unsupervised, domain-independent system for extracting knowledge in the form of simple semantic networks from text. SNE is based on second-order Markov logic. It uses a bottom-up agglomerative clustering algorithm to jointly cluster relation symbols and object symbols, and allows information to propagate between the clusters as they are formed. Empirical comparisons with three systems on a large real-world Web dataset show the promise of our approach.

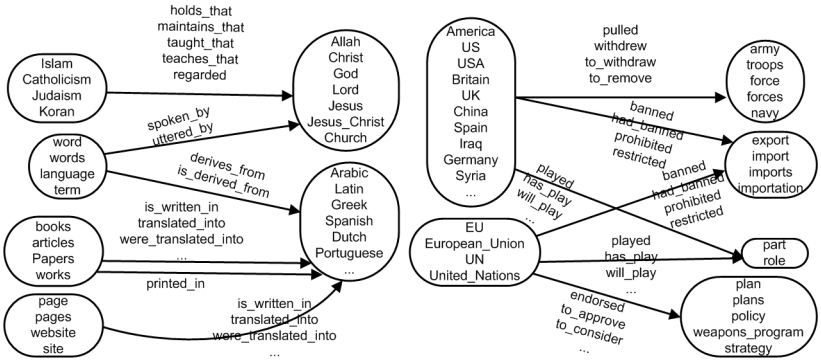

Fig. 1. Fragments of a semantic network learned by SNE. Nodes are concept clusters, and the labels of links are relation clusters. More fragments are available at http://alchemy.cs.washington.edu/papers/kok08.

Directions for future work include: integrating tuple extraction into SNE's Markov logic framework so that information can flow between semantic network learning and tuple extraction, potentially improving the performance of both; extending the learning mechanism so as to learn richer semantic networks as well as complex logical theories from text; etc.

Acknowledgments. This research was partly funded by DARPA contracts NBCH-D030010/02-000225, FA8750-07-D-0185, and HR0011-07-C-0060, DARPA grant FA8750-05-2-0283, NSF grant IIS-0534881, and ONR grant N-00014-05-1-0313. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, NSF, ONR, or the United States Government.

References

1. M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *Proc. IJCAI-2007*, Hyderabad, India, 2007. AAAI Press.
2. M. Banko and O. Etzioni. Strategies for lifelong knowledge extraction from the web. In *Proc. K-CAP-2007*, British Columbia, Canada, 2007.
3. E. Charniak. *Toward a Model of Children's Story Comprehension*. PhD thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Boston, MA, 1972.
4. M. W. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proc. AAAI-98*, pages 509–516, Madison, WI, 1998. AAAI Press.
5. I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proc. KDD-2003*, Washington, DC, 2003.

6. M. G. Dyer. *In-Depth Understanding*. MIT Press, Cambridge, MA, 1983.
7. O. Etzioni, M. Banko, and M. J. Cafarella. Machine reading. In *Proc. 2007 AAAI Spring Symposium on Machine Reading*, Palo Alto, CA, 2007. AAAI Press.
8. C. Gellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.
9. M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1987.
10. T. Hasegawa, S. Sekine, and R. Grishman. Discovering relations among named entities from large corpora. In *Proc. ACL-2004*, Barcelona, Spain, 2004.
11. C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N Ueda. Learning systems of concepts with an infinite relational model. In *Proc. AAAI-2006*, Boston, MA, 2006. AAAI Press.
12. S. Kok and P. Domingos. Statistical predicate invention. In *Proc. ICML-2007*, pages 443–440, Corvallis, Oregon, 2007. ACM Press.
13. W. G. Lehnert. *The Process of Question Answering*. Erlbaum, Hillsdale, NJ, 1978.
14. A. McCallum and D. Jensen. A note on the unification of information extraction and data mining using conditional-probability, relational models. In *Proc. IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, pages 79–86, Acapulco, Mexico, 2003. IJCAII.
15. A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. KDD-2000*, pages 169–178, 2000.
16. T. Mitchell. Reading the web: A breakthrough goal for AI. *AI Magazine*, 26(3):12–16, 2005.
17. R. J. Mooney. Learning for semantic parsing. In *Proc. CICLing-2007*, Mexico City, Mexico, 2007. Springer.
18. M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Names and similarities on the web: Fact extraction on the fast lane. In *Proc. ACL/COLING-2006*, 2006.
19. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.
20. M. R. Quillian. Semantic memory. In M. L. Minsky, editor, *Semantic Information Processing*, pages 216–270. MIT Press, Cambridge, MA, 1968.
21. K. Rajaraman and A-H. Tan. Mining semantic networks for knowledge discovery. In *Proc. ICMD-2003*, 2003.
22. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.
23. R. C. Schank and C. K. Riesbeck. *Inside Computer Understanding*. Erlbaum, Hillsdale, NJ, 1981.
24. G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.
25. Y. Shinyama and S. Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proc. HLT-NAACL-2006*, New York, New York, 2006.
26. Y. W. Wong and R. J. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proc. ACL-2007*, Prague, Czech Republic, 2007.
27. Z. Xu, V. Tresp, K. Yu, and H.-P. Kriegel. Infinite hidden relational models. In *Proc. UAI-2006*, Cambridge, MA, 2006.
28. A. Yates and O. Etzioni. Unsupervised resolution of objects and relations on the web. In *Proc. NAACL-HLT-2007*, Rochester, NY, 2007.
29. L. S. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammers. In *Proc. UAI-2005*, Edinburgh, Scotland, 2005.