Graph-Based Lexicon Expansion with Sparsity-Inducing Penalties

Dipanjan Das and **Noah A. Smith**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{dipanjan,nasmith}@cs.cmu.edu

Abstract

We present novel methods to construct compact natural language lexicons within a graph-based semi-supervised learning framework, an attractive platform suited for propagating soft labels onto new natural language types from seed data. To achieve compactness, we induce sparse measures at graph vertices by incorporating sparsity-inducing penalties in Gaussian and entropic pairwise Markov networks constructed from labeled and unlabeled data. Sparse measures are desirable for high-dimensional multi-class learning problems such as the induction of labels on natural language types, which typically associate with only a few labels. Compared to standard graph-based learning methods, for two lexicon expansion problems, our approach produces significantly smaller lexicons and obtains better predictive performance.

1 Introduction

Semi-supervised learning (SSL) is attractive for the learning of complex phenomena, for example, linguistic structure, where data annotation is expensive. Natural language processing applications have benefited from various SSL techniques, such as distributional word representations (Huang and Yates, 2009; Turian et al., 2010; Dhillon et al., 2011), self-training (McClosky et al., 2006), and entropy regularization (Jiao et al., 2006; Smith and Eisner, 2007). In this paper, we focus on semi-supervised learning that uses a graph constructed from labeled and unlabeled data. This framework, **graph-based SSL**—see Bengio et al. (2006) and Zhu (2008) for introductory material on this topic—has been widely used and has been shown to perform better than several other semi-supervised algorithms on benchmark datasets (Chapelle et al., 2006, ch. 21). The method constructs a graph where a small portion of vertices correspond to labeled instances, and the rest are unlabeled. Pairs of vertices are connected by weighted edges denoting the similarity between the pair. Traditionally, Markov random walks (Szummer and Jaakkola, 2001; Baluja et al., 2008) or optimization of a loss function based on smoothness properties of the graph (Corduneanu and Jaakkola, 2003; Zhu et al., 2003; Subramanya and Bilmes, 2008, *inter alia*) are performed to propagate labels from the labeled vertices to the unlabeled ones.

In this work, we are interested in multi-class generalizations of graph-propagation algorithms suitable for NLP applications, where each graph vertex can assume one *or more* out of many possible labels (Talukdar and Crammer, 2009; Subramanya and Bilmes, 2008, 2009). For us, graph vertices correspond to natural language *types* (not tokens) and undirected edges between them are weighted using a similarity metric. Recently, this setup has been used to learn soft labels on natural language types (say, word *n*-grams or syntactically disambiguated predicates) from seed data, resulting in large but noisy *lexicons*, which are used to constrain structured prediction models. Applications have ranged from domain adaptation of part-of-speech (POS) taggers (Subramanya et al., 2010), unsupervised learning of POS taggers by using bilingual graph-based projections (Das and Petrov, 2011), and shallow semantic parsing for unknown predicates (Das and Smith, 2011). However, none of the above captured the empirical fact that only a few categories typically associate with a given type (vertex). Take the case of POS tagging: Subramanya et al. (2010) construct a graph over trigram types as vertices, with 45 possible tags for the middle word of a trigram as the label set for each vertex. It is empirically observed that contextualized word types can assume very few (most often, one) POS tags. However, along with graph smoothness terms, they apply a penalty that encourages distributions to be close to uniform, the premise being that it would maximize the entropy of the distribution for a vertex that is far away or disconnected from a labeled vertex. To prefer maximum entropy solutions in low confidence regions of graphs, a similar entropic penalty is applied by Subramanya and Bilmes (2008, 2009).

In this paper, we make two major algorithmic contributions. First, we relax the assumption made by most previous work (Zhu and Ghahramani, 2002; Baluja et al., 2008; Subramanya and Bilmes, 2008; Subramanya and Bilmes, 2009; Subramanya et al., 2010; Das and Petrov, 2011; Das and Smith, 2011) that the ℓ_1 norm of the masses assigned to the labels for a given vertex must be 1. In other words, in our framework, the label distribution at each vertex is *unnormalized*—the only constraint we put on the vertices' vectors is that they must be nonnegative.[1] This relaxation simplifies optimization: since only a nonnegativity constraint for each label's mass at each vertex needs to be imposed, we can apply a generic quasi-Newton method (Zhu et al., 1997).

Second, we replace the penalties that prefer maximum entropy, used in prior work, with penalties that aim to identify *sparse* unnormalized measures at each graph vertex. We achieve this by penalizing the graph propagation objective with the ℓ_1 norm or the mixed $\ell_{1,2}$ norm (Kowalski and Torrésani, 2009) of the measures at each vertex, aiming for global and vertex-level sparsity, respectively. Importantly, the proposed graph objective functions are convex, so we avoid degenerate solutions and local minima.

We present experiments on two natural language lexicon expansion problems in a semi-supervised setting: (i) inducing distributions of POS tags over n-gram types in the *Wall Street Journal* section of the Penn Treebank corpus (Marcus et al., 1993) and (ii) inducing distributions of semantic frames (Fillmore, 1982) over predicates unseen in annotated data. Our methods produce sparse measures at graph vertices resulting in compact lexicons, and also result in better performance with respect to label propagation using Gaussian penalties (Zhu and Ghahramani, 2002) and entropic measure propagation (Subramanya and Bilmes, 2009), two state-of-the-art graph propagation algorithms.

2 Model

2.1 Graph-Based SSL as MAP Inference

Let $\mathcal{D}_l = \{(\mathbf{x}_j, r_j)\}_{j=1}^{l}$ denote l annotated data *types*;[2] $\mathbf{x}_j$'s empirical label distribution is r_j. Let the unlabeled data types be denoted by $\mathcal{D}_u = \{\mathbf{x}_i\}_{i=l+1}^{m}$. Usually, $l \ll m$. Thus, the entire dataset can be called $\mathcal{D} \triangleq \mathcal{D}_l \cup \mathcal{D}_u$. Traditionally, the graph-based SSL problem has been set up as follows. Let $\mathcal{G} = (V, E)$ correspond to an undirected graph with vertices V and edges E. $\mathcal{G}$ is constructed by transforming each data type $\mathbf{x}_i \in \mathcal{D}$ to a vertex; thus $V = \{1, 2, \ldots, m\}$, and $E \subseteq V \times V$. Let V_l (V_u) denote the labeled (unlabeled) vertices. Moreover, we assume a symmetric weight matrix $\mathbf{W}$ that defines the similarity between a pair of vertices $i, k \in V$. We first define a component of this matrix as $w_{ij} \triangleq [\mathbf{W}]_{ik} = \mathrm{sim}(\mathbf{x}_i, \mathbf{x}_k)$. We also fix $w_{ii} = 0$ and set $w_{ik} = w_{ki} = 0$ if $k \notin \mathcal{N}(i)$ and $i \notin \mathcal{N}(k)$, where $\mathcal{N}(j)$ denotes the K-nearest neighbors of vertex j, to reduce the density of the graph. We next define an unnormalized measure q_i for every vertex $i \in V$. As mentioned before, we have r_j, a probability distribution estimated from annotated data for a labeled vertex $j \in V_l$. q_i and r_j are $|Y|$-dimensional measures, where Y is the possible set of labels; while r_j lies within the $|Y|$-dimensional probability simplex,[3] q_i are unnormalized with each component $q_i(y) \geq 0$. For most NLP problems, r_j are expected to be sparse, with very few components active, the rest being zero.

Graph-based SSL aims at finding the best $\boldsymbol{q} = \{q_i : 1 \leq i \leq m\}$ given the empirical distributions r_j, and the weight matrix $\mathbf{W}$, which provides

[1] Moreover, we also assume the edge weights in a given graph are unconstrained, consistent with prior work on graph-based SSL (Das and Petrov, 2011; Das and Smith, 2011; Subramanya and Bilmes, 2008; Subramanya and Bilmes, 2009; Subramanya et al., 2010; Zhu and Ghahramani, 2002).

[2] As explained in more detail in §4, these types are entities like n-grams or individual predicates, not tokens in running text.

[3] Note that our framework does not necessitate that r_j be a normalized probability distribution; we could have unnormalized r_j to allow strongly evident types appearing in more data to have larger influence than types that appear infrequently. We leave this extension to future work.

the geometry of all the vertices. We visualize this problem using a pairwise Markov network (MN). For every vertex (including labeled ones) $i \in V$, we create a variable X_i. Additionally, for labeled vertices $j \in V_l$, we create variables $\hat{X}_j$. All variables in the MN are defined to be *vector-valued*; specifically, variables X_i, $\forall i \in V$, take value q_i, and variables $\hat{X}_j$ corresponding to the labeled vertices in $\mathcal{G}$ are observed with values r_j. An example factor graph for this MN, with only four vertices, is shown in Figure 1. In the figure, the variables indexed by 1 and 4 correspond to labeled vertices. Factor ϕ_j with scope $\{X_j, \hat{X}_j\}$ encourages q_j to be close to r_j. For every edge $i - k \in E$, factor φ_{i-k} encourages similarity between q_i and q_k, making use of the weight matrix $\mathbf{W}$ (i.e., when w_{ik} is larger, the two measures are more strongly encouraged to be close). These factors are white squares with solid boundaries in the figure. Finally, we define unary factors on all variables $X_i, i \in V$, named $\psi_i(X_i)$, that can incorporate prior information. In Figure 1, these factors are represented by white squares with dashed boundaries.

According to the factor graph, the joint probability for all the measures q_i, $\forall i \in V$ that we want to induce, is defined as: $P(\boldsymbol{X}; \Phi) =$

$$\frac{1}{Z} \prod_{j=1}^{l} \phi_j(X_j, \hat{X}_j) \cdot \prod_{i-k \in E} \varphi_{i-k}(X_i, X_k) \cdot \prod_{i=1}^{m} \psi_i(X_i)$$

where Φ is the set of all factors in the factor graph, and Z is a partition function that normalizes the factor products for a given configuration of $\boldsymbol{q}$. Since the graph-based SSL problem aims at finding the best $\boldsymbol{q}$, we optimize $\ln P(\boldsymbol{X}; \Phi)$; equivalently,

$$\underset{\boldsymbol{q} \text{ s.t. } \boldsymbol{q} \geq \mathbf{0}}{\arg\max} \sum_{j=1}^{l} \ln \phi_j(X_j, \hat{X}_j) + \sum_{i-k \in E} \ln \varphi_{i-k}(X_i, X_k) + \sum_{i=1}^{m} \ln \psi_i(X_i) \qquad (1)$$

The above denotes an optimization problem with only non-negativity constraints. It equates to maximum *a posteriori* (MAP) inference; hence, the partition function Z can be ignored. We next discuss the nature of the three different factors in Eq. 1.

2.2 Log-Factors as Penalties

The nature of the three types of factors in Eq. 1 governs the behavior of a graph-based SSL algorithm. Hence, the equation specifies a *family* of graph-based methods that generalize prior research.

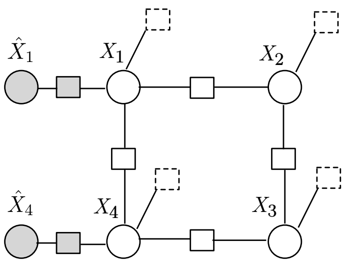

Figure 1: An example factor graph for the graph-based SSL problem. See text for the significance of the shaded and dotted factors, and the shaded variables.

We desire the following properties to be satisfied in the factors: (i) convexity of Eq. 1, (ii) amenability to scalable optimization algorithms, and (iii) sparse solutions as expected in natural language lexicons.

Pairwise factors: In our work, for the pairwise factors $\phi_j(X_j, \hat{X}_j)$ and $\varphi_{i-k}(X_i, X_k)$, we examine two functions that penalize inconsistencies between neighboring vertices: the squared ℓ_2 norm and the Jensen-Shannon (JS) divergence (Burbea and Rao, 1982; Lin, 1991), which is a symmetrized generalization of the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951; Cover and Thomas, 1991). These two divergences are symmetric. Both are inspired by previous work; however, the use of the JS divergence is a novel extension to Subramanya and Bilmes (2008). Specifically, the factors are:

$$\begin{aligned} \ln \phi_j(X_j, \hat{X}_j) &= -\delta(q_j, r_j) & (2) \\ \ln \varphi_{i-k}(X_i, X_k) &= -2 \cdot \mu \cdot w_{ik} \cdot \delta(q_i, q_k) & (3) \end{aligned}$$

where μ is a hyperparameter whose choice we discuss in §4. The function $\delta(u, v)$ for two vectors u and v is defined in two ways:

$$\underset{\text{Gaussian}}{\delta(u, v)} = \|u - v\|_2^2 \qquad (4)$$

$$\underset{\text{Entropic}}{\delta(u, v)} = \tfrac{1}{2} \sum_{y \in Y} \left(u(y) \cdot \ln \frac{2 \cdot u(y)}{u(y) + v(y)} + v(y) \cdot \ln \frac{2 \cdot v(y)}{u(y) + v(y)} \right) \qquad (5)$$

We call the version of $\delta(u, v)$ that uses the squared ℓ_2 distance (Eq. 4) *Gaussian*, as it represents the idea of label propagation via Gaussian fields proposed by Zhu et al. (2003). A minor difference lies in the fact that we include variables $X_j, j \in V_l$ for labeled vertices too, and allow them to change, but penalize them if they go too far away from the observed labeled distributions r_j. The other $\delta(u,v)$ shown in Eq. 5 uses the generalized JS-divergence defined in terms of the generalized KL-divergence for unnormalized measures (O'Sullivan, 1998).[4]

Eq. 5 improves prior work by replacing the asymmetric KL-divergence used to bring the distributions at labeled vertices close to the corresponding observed distributions, as well as replacing the KL-based graph smoothness term with the symmetric JS-divergence (Subramanya and Bilmes, 2008, see first two terms in Eq. 1). Empirical evidence shows that entropic divergences help in multiclass problems where a vertex can assume multiple labels, and may perform better than objectives with quadratic penalties (Subramanya and Bilmes, 2008, 2009).

A major departure from prior work is the use of unnormalized measures in Eq. 4-5, which simplifies optimization even with the complex JS-divergence in the objective function (see §3), and, we will see, produces comparable and often better results than baselines using normalized distributions (see §4).

Unary factors: The unary factors in our factor graph $\psi_i(X_i)$ can incorporate prior information specific to a particular vertex $\mathbf{x}_i$ embodied by the variable X_i. Herein, we examine three straightforward penalties, which can be thought of as penalties that encourage either uniformity or sparsity:

$$\text{Uniform squared } \ell_2\text{: } \ln \psi_i(X_i) = -\lambda \cdot \left\| q_i - \tfrac{1}{|Y|} \right\|_2^2 \quad (6)$$

$$\text{Sparse } \ell_1\text{:} \quad \ln \psi_i(X_i) = -\lambda \cdot \|q_i\|_1 \quad (7)$$

$$\text{Sparse } \ell_{1,2}\text{:} \quad \ln \psi_i(X_i) = -\lambda \cdot \|q_i\|_1^2 \quad (8)$$

where λ is a hyperparameter whose choice we discuss in §4. The penalty expressed in Eq. 6 penalizes q_i if it is far away from the uniform distribution. This penalty has been used previously (Das and Petrov, 2011; Das and Smith, 2011; Subramanya et al., 2010), and is similar to the maximum entropy penalty of Subramanya and Bilmes (2008, 2009). The intuition behind its use is that for low confidence or disconnected regions, one would prefer to have a uniform measure on a graph vertex. The penalties in equations 7–8, on the other hand, encourage sparsity in the measure q_i; these are related to regularizers for generalized linear models: the lasso (Tibshirani, 1996) and the elitist lasso (Kowalski and Torrésani, 2009). The former encourages global sparsity, the latter sparsity per vertex.[5] For each vertex, the $\ell_{1,2}$ penalty takes the form:

$$\|q_i\|_1^2 = \left(\sum_{y \in Y} |q_i(y)| \right)^2 \quad (9)$$

The ℓ_1 norm encourages its argument to be sparse, while the usual observed effect of an ℓ_2 norm is a dense vector without many extreme values. The $\ell_{1,2}$ penalty is the squared ℓ_2 norm of the ℓ_1 norms of every q_i, hence it promotes sparsity within each vertex, but we observe density over the vertices that are selected.

Talukdar (2010) enforced label sparsity for information extraction by discarding poorly scored labels during graph propagation updates, but did not use a principled mechanism to arrive at sparse measures at graph vertices. Unlike the uniform penalty (Eq. 6), sparsity corresponds to the idea of entropy *minimization* (Grandvalet and Bengio, 2004). Since we use unnormalized measures at each variable X_i, for low confidence graph regions or disconnected vertices, sparse penalties will result in all zero components in q_i, which conveys that the graph propagation algorithm is not confident on any potential label, a condition that is perfectly acceptable.

Model variants: We compare six objective functions: we combine factor representations from each of Eqs. 4–5 with those from each of Eqs. 6–8, replacing them in the generic graph objective function of Eq. 1. The nature of these six models is succinctly summarized in Table 1. For each model, we find the best set of measures $\boldsymbol{q}$ that maximize the corresponding graph objective functions, such that $\boldsymbol{q} \geq \mathbf{0}$. Note that in each of the graph objectives, we have two hyperparameters μ and λ that control the influence of the second and the third terms of Eq. 1 re-

[4] The generalized KL divergence is defined as $D_{KL}(u\|v) = \sum_y \left(u(y) \ln \frac{u(y)}{v(y)} - u(y) + v(y) \right)$.

[5] One could additionally consider a non-sparse penalty based on the squared ℓ_2 norm with zero mean: $\ln \psi_i(X_i) = -\lambda \cdot \|q_i\|_2^2$. We experimented with this unary penalty (along with the pairwise Gaussian penalty for binary factors) for the semantic frame lexicon expansion problem, and found that it performs exactly on par with the squared ℓ_2 penalty with uniform mean. To limit the number of non-sparse graph objectives, we omit detailed discussion of experiments with this unary penalty.

abbrev.	factors	
	pairwise	unary
UGF-ℓ_2	Gaussian	Uniform squared ℓ_2
UGF-ℓ_1	Gaussian	Sparse ℓ_1
UGF-$\ell_{1,2}$	Gaussian	Sparse $\ell_{1,2}$
UJSF-ℓ_2	Entropic	Uniform squared ℓ_2
UJSF-ℓ_1	Entropic	Sparse ℓ_1
UJSF-$\ell_{1,2}$	Entropic	Sparse $\ell_{1,2}$

Table 1: Six variants of graph objective functions novel to this work. These variants combine the pairwise factor representations from Eqs. 4–5 with unary factor representations from each of Eqs. 6–8 (which either encourage uniform or sparse measures), to be used in the graph objective function expressed in Eq. 1.

spectively. We discuss how these hyperparameters are chosen in §4.

Baseline Models: We compare the performance of the six graph objectives of Table 1 with two strong baselines that have been used in previous work. These two models use the following two objective functions, and find $\boldsymbol{q}$ s.t. $\boldsymbol{q} \geq \mathbf{0}$ and $\forall i \in V, \sum_{y \in Y} q_i(y) = 1$. The first is a normalized Gaussian field with a squared uniform ℓ_2 penalty as the unary factor (**NGF**-ℓ_2):

$$\underset{\substack{\boldsymbol{q},\text{ s.t. } \boldsymbol{q}\geq\mathbf{0},\\ \forall i\in V, \|q_i\|_1=1}}{\arg\min} \sum_{j=1}^{l} \|q_j - r_j\|_2^2 + \sum_{i=1}^{m} \left(\mu \sum_{k \in \mathcal{N}(i)} w_{ik} \|q_i - q_k\|_2^2 + \lambda \left\| q_i - \frac{1}{|Y|} \right\|_2^2 \right) \quad (10)$$

The second is a normalized KL field with an entropy penalty as the unary factor (**NKLF**-ME):

$$\underset{\substack{\boldsymbol{q},\text{ s.t. } \boldsymbol{q}\geq\mathbf{0},\\ \forall i\in V, \|q_i\|_1=1}}{\arg\min} \sum_{j=1}^{l} D_{KL}(r_j \parallel q_j) + \sum_{i=1}^{m} \left(\mu \sum_{k \in \mathcal{N}(i)} w_{ik} D_{KL}(q_i \parallel q_k) - \lambda \cdot H(q_i) \right) \quad (11)$$

where $H(q_i)$ denotes the Shannon entropy of the distribution q_i. Both these objectives are constrained by the fact that every q_i must be within the $|Y|$-dimensional probability simplex. The objective function in 10 has been used previously (Das and Smith, 2011; Subramanya et al., 2010) and serves as a generalization of Zhu et al. (2003). The entropic objective function in 11, originally called *measure propagation*, performed better at multiclass problems when compared to graph objectives using the quadratic criterion (Subramanya and Bilmes, 2008).

3 Optimization

The six variants of Eq. 1 in Table 1 are convex in $\boldsymbol{q}$. This is because the ℓ_1, squared ℓ_2 and the $\ell_{1,2}$ penalties are convex. Moreover, the generalized JS-divergence term, which is a sum of two KL-divergence terms, is convex (Cover and Thomas, 1991). Since we choose μ, λ and w_{ik} to be nonnegative, these terms' sums are also convex. The graph objectives of the two baselines noted in expressions 10–11 are also convex because negative entropy in expression 11 is convex, and rest of the penalties are the same as our six objectives. In our work, to optimize the objectives of Table 1, we use a generic quasi-Newton gradient-based optimizer that can handle bound-inequality constraints, called L-BFGS-B (Zhu et al., 1997). Partial derivatives of the graph objectives are computed with respect to each parameter $\forall i, y, q_i(y)$ of $\boldsymbol{q}$ and passed on to the optimizer which updates them such that the objective function of Eq. 1 is maximized. Note that since the ℓ_1 and $\ell_{1,2}$ penalties are non-differentiable at 0, special techniques are usually used to compute updates for unconstrained parameters (Andrew and Gao, 2007). However, since $\boldsymbol{q} \geq \mathbf{0}$, their absolute value can be assumed to be right-continuous, making the function differentiable. Thus,

$$\frac{\partial}{\partial q_i(y)} \|q_i\|_1 = 1 \qquad \frac{\partial}{\partial q_i(y)} \|q_i\|_1^2 = 2 \cdot \|q_i\|_1$$

(We omit the form of the derivatives of the other penalties for space.) There are several advantages to taking this route towards optimization. The ℓ_2 and the JS-divergence penalties for the pairwise terms can be replaced with more interesting convex divergences if required, and still optimization will be straightforward. Moreover, the nonnegative constraints make optimization with sparsity inducing penalties easy. Finally, computing the objective function and the partial derivatives is easily parallelizable on MPI (Gropp et al., 1994) or MapReduce (Dean and Ghemawat, 2008) architectures, by dividing up the computation across graph vertices.

In comparison, constrained problems such as the one in Eq. 11 require a specialized alternating minimization technique (Subramanya and Bilmes, 2008, 2009), that performs two passes through the graph vertices during one iteration of updates, introduces an auxiliary set of probability distributions (thus, increasing memory requirements) and another hyperparameter α that is used to transform the weight matrix $\mathbf{W}$ to be suitable for the alternating minimization procedure. To optimize the baseline objectives, we borrow the gradient-free iterative updates described by Subramanya and Bilmes (2009) and Subramanya et al. (2010).

4 Experiments

In this section, we compare the six graph objective functions in Table 1 with the two baseline objectives on two lexicon expansion tasks.

4.1 POS Lexicon Expansion

We expand a POS lexicon for word types with a context word on each side, using distributional similarity in an unlabeled corpus and few labeled trigrams.

Data and task: We constructed a graph over *word trigram types* as vertices, using co-occurrence statistics. Following Das and Petrov (2011) and Subramanya et al. (2010), a similarity score between two trigram types was computed by measuring the cosine similarity between their empirical sentential context statistics. This similarity score resulted in the symmetric weight matrix $\mathbf{W}$, defining edge weights between pairs of graph vertices. Details of the similarity computation are given in those papers. $\mathbf{W}$ is thresholded so that only the K nearest neighbors for each vertex have similarity greater than zero, giving a sparse graph. We set $K = 8$ as it resulted in the sparsest graph which was fully connected.[6] For this task, Y is the set of 45 POS tags defined in the Penn Treebank (Marcus et al., 1993), and the measure q_i for vertex i (for trigram type $\mathbf{x}_i$) corresponds to the set of tags that can be associated with the middle word of $\mathbf{x}_i$. The trigram representation, as in earlier work, helps reduce the ambiguity of POS tags for the middle word, and helps in graph construction. The 690,705-vertex graph was constructed over all trigram types appearing in Sections 00–21 (union of the training and development sets used for POS tagging experiments in prior work) of the WSJ section of the Penn Treebank, but co-occurrence statistics for graph construction were gathered from a million sentences drawn from the English Gigaword corpus (Graff, 2003).

Given the graph $\mathcal{G}$ with m vertices, we assume that the tag distributions $\boldsymbol{r}$ for l labeled vertices are also provided. Our goal is to find the best set of measures $\boldsymbol{q}$ over the 45 tags for all vertices in the graph. Prior work used a similar lexicon for POS domain adaptation and POS induction for resource-poor languages (Das and Petrov, 2011; Subramanya et al., 2010); such applications of a POS lexicon are out of scope here; we consider only the lexicon expansion problem and do an intrinsic evaluation at a type-level to compare the different graph objectives.

Experimental details: To evaluate, we randomly chose 6,000 out of the 690,705 types for development. From the remaining types, we randomly chose 588,705 vertices for testing. This left us with 96,000 types from which we created sets of different sizes containing 3,000, 6,000, 12,000, 24,000, 48,000 and 96,000 *labeled* types, creating 6 increasingly easy transduction settings. The development and the test types were kept constant for direct performance comparison across the six settings and our eight models. After running inference, the measure q_i at vertex i was normalized to 1. Next, for all thresholds ranging from 0 to 1, with steps of 0.001, we measured the average POS tag precision and recall on the development data – this gave us the area under the precision-recall curve (prAUC), which is often used to measure performance on retrieval tasks. Given a transduction setting and the final $\boldsymbol{q}$ for an objective, hyperparameters μ and λ were tuned on the development set by performing a grid search, targeting prAUC.[7] We ran 100 rounds

[6] Our proposed methods can deal with graphs containing disconnected components perfectly well. Runtime is asymptotically linear in K for all objectives considered here.

[7] For the objectives using the uniform ℓ_2 and the maximum entropy penalties, namely **UGF**-ℓ_2, **UJSF**-ℓ_2, **NGF**-ℓ_2 and **NKLF**-ME, we chose λ from $\{0, 10^{-6}, 10^{-4}, 0.1\}$. For the rest of the models using sparsity inducing penalties, we chose λ from $\{10^{-6}, 10^{-4}, 0.1\}$. This suggests that for the former type of objectives, we allowed a zero unary penalty if that setting resulted in the best development performance, while for the latter type of models, we enforced a positive unary penalty. In fact, $\lambda = 0$ was chosen in several cases for the objectives with uniform penalties indicating that uniformity hurts performance. We chose μ from $\{0.1, 0.5, 1.0\}$.

$\lvert\mathcal{D}_l\rvert$:	3K	6K	12K	24K	48K	96K
NGF-ℓ_2	0.208	0.219	0.272	0.335	0.430	0.544
NKLF-ME	0.223	0.227	0.276	0.338	0.411	0.506
UGF-ℓ_2	0.223	0.257	0.314	**0.406**	**0.483**	**0.564**
UGF-ℓ_1	0.223	0.257	0.309	**0.406**	**0.483**	0.556
UGF-$\ell_{1,2}$	0.223	0.256	0.313	0.403	0.478	0.557
UJSF-ℓ_2	**0.271**	0.250	0.310	0.364	0.409	0.481
UJSF-ℓ_1	0.227	0.257	**0.317**	0.369	0.410	0.481
UJSF-$\ell_{1,2}$	0.227	**0.258**	0.309	0.369	0.409	0.479

Table 2: Area under the precision recall curve for the two baseline objectives and our methods for POS tag lexicon induction. This is a measure of how well the type lexicon (for some types unlabeled during training) is recovered by each method. The test set contains 588,705 types.

of iterative updates for all 8 graph objectives.

Type-level evaluation: To measure the quality of the lexicons, we perform type level evaluation using area under the precision-recall curve (prAUC). The same measure (on development data) was used to tune the two hyperparameters. Table 2 shows the results measured on 588,705 test vertices (the same test set was used for all the transduction settings). The general pattern we observe is that our unnormalized approaches almost always perform better than the normalized baselines. (The exception is the 3,000 labeled example case, where most unnormalized models are on par with the better baseline.) In scenarios with fewer labeled types, pairwise entropic penalties perform better than Gaussian ones, and the pattern reverses as more labeled types come available. This trend is the same when we compare only the two baselines. In four out of the six transduction settings, one of the sparsity-inducing graph objectives achieves the best performance in terms of prAUC, which is encouraging given that they generally produce smaller models than the baselines.

Overall, though, using sparsity-inducing unary factors seems to have a weak negative effect on performance. Their practical advantage, however is apparent when we consider the size of the model. After the induction of the set of measures $\boldsymbol{q}$ for all transduction settings and all graph objectives, we noticed that our numerical optimizer (LBFGS-B) often assigns extremely small positive values rather than zero. This problem can be attributed to several artifacts, including our limit of 100 iterations of optimization. Hence, we use a global threshold of 10^{-6}, and treat any real value below this threshold to be zero. Figure 2 shows the number of non-zero components in $\boldsymbol{q}$ (or, the lexicon size) for the graph objectives that achieve sparsity (baselines **NGF**-ℓ_2 and **NKLF**-ME, plus our **UGF**-ℓ_2 are not expected to, and do not, achieve sparsity; surprisingly **UJSF**-ℓ_2 *does* and is shown). Even though the hyperparameters μ and λ in the graph objective functions were not tuned towards sparsity, we see that sparsity-inducing factors are able to achieve far more compact lexicons. Sparsity is desirable in settings where labeled development data for tuning thresholds that select the most probable labels for a given type is unavailable (e.g., Das and Petrov, 2011).

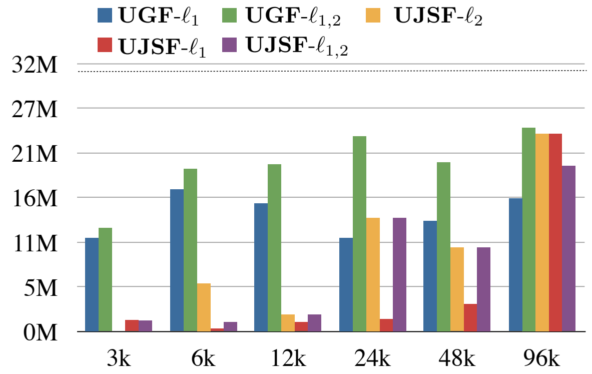

Figure 2: The number of non-zero components in $\boldsymbol{q}$ for five graph objective functions proposed in this work, plotted against various numbers of labeled datapoints. Note that **NGF**-ℓ_2, **NKLF**-ME and **UGF**-ℓ_2 produce non-zero components for virtually all $\boldsymbol{q}$, and are therefore not shown (the dotted line marks the maximally non-sparse solution, with 31,081,725 components). All of these five objectives result in sparsity. On average, the objectives employing entropic pairwise penalties with sparse unary penalties **UJSF**-ℓ_1 and **UJSF**-$\ell_{1,2}$ produce very sparse lexicons. Although **UGF**-ℓ_2 produces no sparsity at all, its entropic counterpart **UJSF**-ℓ_2 produces considerable sparsity, which we attribute to JS-divergence as a pairwise penalty.

4.2 Expansion of a Semantic Frame Lexicon

In a second set of experiments, we follow Das and Smith (2011, D&S11 henceforth) in expanding a lexicon that associates lexical predicates (*targets*) with semantic *frames* (abstract events or scenarios that a predicate evokes when used in a sentential context) as labels. More concretely, each vertex in the graph corresponds to a lemmatized word type with its coarse part of speech, and the labels are frames from the FrameNet lexicon (Fillmore et al., 2003). Graph construction leverages distributional

	UNKNOWN		ALL		
	PREDICATES		PREDICATES		lexicon
	exact	**partial**	**exact**	**partial**	size
Supervised	23.08	46.62	82.97	90.51	-
***NGF**-ℓ_2	39.86	62.35	83.51	91.02	128,960
NKLF-ME	36.36	60.07	83.40	90.95	128,960
UGF-ℓ_2	37.76	60.81	83.44	90.97	128,960
UGF-ℓ_1	39.86	62.85	83.51	91.04	122,799
UGF-$\ell_{1,2}$	39.86	62.85	83.51	91.04	128,732
UJSF-ℓ_2	40.56	62.81	83.53	91.04	128,232
UJSF-ℓ_1	39.16	62.43	83.49	91.02	128,771
UJSF-$\ell_{1,2}$	**42.67**	**65.29**	**83.60**	**91.12**	**45,544**

Table 3: Exact and partial frame identification accuracy with lexicon size (non-zero frame components). The "unknown predicates" section of the test data contains 144 targets, while the entire test set contains 4,458 targets. Bold indicates best results. The **UJSF**-$\ell_{1,2}$ model produces statistically significant results ($p < 0.001$) for all metrics with respect to the supervised baseline used in D&S11. For both the unknown targets as well as the whole test set. However, it is weakly significant ($p < 0.1$) compared to the **NGF**-ℓ_2 model for the unseen portion of the test set, when partial frame matching is used. For rest of the settings, the two are statistically indistinguishable. * indicates the best results in D&S11.

similarity as well as linguistic annotations.

Data: We borrow the graph-based SSL process of D&S11 in its entirety. The constructed graph contains 64,480 vertices, each corresponding to a target, out of which 9,263 were drawn from the labeled data. The possible set of labels Y is the set of 877 frames defined in FrameNet; the measure q_i corresponds to the set of frames that a target can evoke. The targets drawn from FrameNet annotated data (l = 9,263) have frame distributions r_i with which the graph objectives are seeded.[8]

Evaluation: The evaluation metric used for this task is frame disambiguation accuracy on a blind test set containing marked targets in free text. A section of this test set contained 144 targets, previously unseen in annotated FrameNet data; this section is of interest to us and we present separate accuracy results on it. Given the measure q_i over frames induced using graph-based SSL for target i, we truncate it to keep at most the top M frames that get the highest mass under q_i, only retaining those with non-zero values. If all components of q_i are zero, we remove target i from the lexicon, which is often the case in the sparsity-inducing graph objectives. If a target is unseen in annotated data, a separate probabilistic model (which serves as a supervised baseline like in D&S11, row 1 in Table 3) disambiguates among the M filtered frames observing the sentential context of the target instance. This can be thought of as combining type- and token-level information for inference. If the target was previously seen, it is disambiguated using the supervised baseline. The test set and the probabilistic model are identical to the ones in D&S11. We fixed K, the number of nearest neighbors for each vertex, to be 10. For each graph objective, μ, λ and M were chosen by five-fold cross-validation. The cross-validation sets were the same as the ones described in §6.3 of D&S11.[9]

Results and discussion: Table 3 shows frame identification accuracy, both using exact match as well as partial match that assigns partial credit when a related frame is predicted (Baker et al., 2007). The final column presents lexicon size in terms of the set of truncated frame distributions (filtered according to the top M frames in q_i) for all the targets in a graph. All the graph-based models are better than the supervised baseline; for our objectives using pairwise Gaussian fields with sparse unary penalties, the accuracies are equal or better with respect to **NGF**-ℓ_2; however, the lexicon sizes are reduced by a few hundred to a few thousand entries. Massive reduction in lexicon sizes (as in the POS problem in §4.1) is not visible for these objectives because we throw out most of the components of the entire set of distributions $\boldsymbol{q}$ and keep only at most the top M (which is automatically chosen to be 2 for all objectives) frames per target. Although a significant number of components in the whole distribution $\boldsymbol{q}$ in the sparse objectives get zero mass, the M components for a target tend to be non-zero for a majority of the targets. Better results are observed for the objectives using entropic pairwise penalties; the ob-

[8] We refer the reader to D&S11 for the details of the graph construction method, the FrameNet dataset used, example semantic frames, and an excerpt of the graph over targets.

[9] We chose μ from $\{0.01, 0.1, 0.3, 0.5, 1.0\}$; λ was chosen from the same sets as the POS problem. The graph construction hyperparameter α described by D&S11 was fixed to 0.2. As in D&S11, M was chosen from $\{2, 3, 5, 10\}$.

(a)

$t =$ *discrepancy*.N	$t =$ *contribution*.N	$t =$ *print*.V	$t =$ *mislead*.V
∗Similarity	∗Giving	∗Text_creation	Experiencer_obj
Natural_features	Money	Sending	∗Prevarication
Prevarication	Commitment	Dispersal	Manipulate_into_doing
Quarreling	Assistance	Reading	Compliance
Duplication	Earnings_and_losses	Statement	Evidence

$t =$ *abused*.A	$t =$ *maker*.N	$t =$ *inspire*.V	$t =$ *failed*.A
Offenses	Commerce_scenario	Cause_to_start	Success_or_failure
Killing	∗Manufacturing	Experiencer_obj	∗Successful_action
Compliance	Businesses	∗Subjective_influence	Unattributed_information
Differentiation	Behind_the_scenes	Evoking	Piracy
Committing_crime	Supply	Attempt_suasion	Want_suspect

(b)

$t =$ *discrepancy*.N	$t =$ *contribution*.N	$t =$ *print*.V	$t =$ *mislead*.V
*Similarity	*Giving	*Text_creation	*Prevarication
Non-commutative_statement	Commerce_pay	State_of_entity	Experiencer_obj
Natural_features	Commitment	Dispersal	Manipulate_into_doing
	Assistance	Contacting	Reassuring
	Earnings_and_losses	Reading	Evidence

$t =$ *abused*.A	$t =$ *maker*.N	$t =$ *inspire*.V	$t =$ *failed*.A
	*Manufacturing	Cause_to_start	*Successful_action
	Businesses	*Subjective_influence	Successfully_communicate_message
	Commerce_scenario	Objective_influence	
	Supply	Experiencer_obj	
	Being_active	Setting_fire	

Table 4: Top 5 frames (if there are ≥ 5 frames with mass greater than zero) according to the graph posterior $q_t(f)$ for (a) **NGF**-ℓ_2 and (b) **UJSF**-$\ell_{1,2}$, given eight unseen predicates in annotated FrameNet data. * marks the correct frame, according to the predicate instances in test data (each of these predicates appear only once in test data). Note that **UJSF**-$\ell_{1,2}$ ranks the correct frame higher than **NGF**-ℓ_2 for several predicates, and produces sparsity quite often; for the predicate *abused*.A, the correct frame is not listed by **NGF**-ℓ_2, while **UJSF**-$\ell_{1,2}$ removes it altogether from the expanded lexicon, resulting in compactness.

jective **UJSF**-$\ell_{1,2}$ gives us the best absolute result by outperforming the baselines by strong margins, and also resulting in a tiny lexicon, less than half the size of the baseline lexicons. The size can be attributed to the removal of predicates for which all frame components were zero ($q_i = \mathbf{0}$). Table 4 contrasts the induced frames for several unseen predicates for the **NGF**-ℓ_2 and the **UJSF**-ℓ_2 objectives; the latter often ranks the correct frame higher, and produces a small set of frames per predicate.

5 Conclusion

We have presented a family of graph-based SSL objective functions that incorporate penalties encouraging sparse measures at each graph vertex. Our methods relax the oft-used assumption that the measures at each vertex form a normalized probability distribution, making optimization and the use of complex penalties easier than prior work. Optimization is also easy when there are additional terms in a graph objective suited to a specific problem; our generic optimizer would simply require the computation of new partial derivatives, unlike prior work that required specialized techniques for a novel objective function. Finally, experiments on two natural language lexicon learning problems show that our methods produce better performance with respect to state-of-the-art graph-based SSL methods, and also result in much smaller lexicons.

Acknowledgments

We thank André Martins, Amar Subramanya, and Partha Talukdar for helpful discussion during the progress of this work and the three anonymous reviewers for their valuable feedback. This research was supported by Qatar National Research Foundation grant NPRP 08-485-1-083, Google's support of the Worldly Knowledge Project, and TeraGrid resources provided by the Pittsburgh Supercomputing Center under NSF grant number TG-DBS110003.

References

G. Andrew and J. Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proc. of ICML*.

C. Baker, M. Ellsworth, and K. Erk. 2007. Task 19: frame semantic structure extraction. In *Proc. of SemEval*.

S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. 2008. Video suggestion and discovery for Youtube: taking random walks through the view graph. In *Proc. of WWW*.

Y. Bengio, O. Delalleau, and N. Le Roux. 2006. Label propagation and quadratic criterion. In Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors, *Semi-Supervised Learning*, pages 193–216. MIT Press.

J. Burbea and C. R. Rao. 1982. On the convexity of some divergence measures based on entropy functions. *IEEE Transactions on Information Theory*, 28:489–495.

O. Chapelle, B. Schölkopf, and A. Zien, editors. 2006. *Semi-Supervised Learning*. MIT Press.

A. Corduneanu and T. Jaakkola. 2003. On information regularization. In *Proc. of UAI*.

T. M. Cover and J. A. Thomas. 1991. *Elements of information theory*. Wiley-Interscience.

D. Das and S. Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proc. of ACL*.

D. Das and N. A. Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *Proc. of ACL*.

J. Dean and S. Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51:107–113, January.

P. S. Dhillon, D. Foster, and L. Ungar. 2011. Multi-view learning of word embeddings via cca. In *Proc. of NIPS*.

C. J. Fillmore, C. R. Johnson, and M. R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3).

C. J. Fillmore. 1982. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea.

D. Graff. 2003. English Gigaword. Linguistic Data Consortium.

Y. Grandvalet and Y. Bengio. 2004. Semi-supervised learning by entropy minimization. In *Proc. of NIPS*.

W. Gropp, E. Lusk, and A. Skjellum. 1994. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press.

F. Huang and A. Yates. 2009. Distributional representations for handling sparsity in supervised sequence labeling. In *Proc. of ACL*.

F. Jiao, S. Wang, C.-H. Lee, R. Greiner, and D. Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proc. of ACL*.

M. Kowalski and B. Torrésani. 2009. Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients. *Signal, Image and Video Processing*, 3:251–264.

S. Kullback and R. A. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22.

J. Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37:145–151.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2).

D. McClosky, E. Charniak, and M. Johnson. 2006. Effective self-training for parsing. In *Proc. of HLT-NAACL*.

J. A. O'Sullivan. 1998. Alternating minimization algorithms: from Blahut-Arimoto to Expectation-Maximization. In A. Vardy, editor, *Codes, Curves, and Signals: Common Threads in Communications*, pages 173–192. Kluwer.

D. A. Smith and J. Eisner. 2007. Bootstrapping feature-rich dependency parsers with entropic priors. In *Proc. of EMNLP*.

A. Subramanya and J. Bilmes. 2008. Soft-supervised learning for text classification. In *Proc. of EMNLP*.

A. Subramanya and J. Bilmes. 2009. Entropic graph regularization in non-parametric semi-supervised classification. In *Proc. of NIPS*.

A. Subramanya, S. Petrov, and F. Pereira. 2010. Efficient Graph-based Semi-Supervised Learning of Structured Tagging Models. In *Proc. of EMNLP*.

M. Szummer and T. Jaakkola. 2001. Partially labeled classification with Markov random walks. In *Proc. of NIPS*. MIT Press.

P. P. Talukdar and K. Crammer. 2009. New regularized algorithms for transductive learning. In *Proc. of the ECML-PKDD*.

P. P. Talukdar. 2010. *Graph-Based Weakly-Supervised Methods for Information Extraction and Integration*. Ph.D. thesis, University of Pennsylvania.

R. Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288.

J. Turian, L. Ratinov, and Y. Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc. of ACL*.

X. Zhu and Z. Ghahramani. 2002. Learning from labeled and unlabeled data with Label Propagation. Technical report, Carnegie Mellon University.

C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23:550–560.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of ICML*.

X. Zhu. 2008. Semi-Supervised Learning Literature Survey. Online publication., July.