A Graph-Based Semi-Supervised Learning for Question Semantic Labeling

Asli Celikyilmaz
Computer Science Division
University of California, Berkeley
asli@berkeley.edu

Dilek Hakkani-Tur
International Computer Science Institute
Berkeley, CA
dilek@icsi.berkeley.edu

Abstract

We investigate a graph-based semi-supervised learning approach for labeling semantic components of questions such as *topic, focus, event, etc.*, for question understanding task. We focus on graph construction to handle learning with dense/sparse graphs and present *Relaxed Linear Neighborhoods* method, in which each node is linearly constructed from varying sizes of its neighbors based on the density/sparsity of its surrounding. With the new graph representation, we show performance improvements on syntactic and real datasets, primarily due to the use of unlabeled data and relaxed graph construction.

1 Introduction

One of the important steps in Question Answering (QA) is *question understanding* to identify semantic components of questions. In this paper, we investigate question understanding based on a machine learning approach to discover semantic components (Table 1).

An important issue in information extraction from text is that one often deals with insufficient labeled data and large number of unlabeled data, which have led to improvements in semi-supervised learning (SSL) methods, e.g., (Belkin and Niyogi., 2002b), (Zhou et al., 2004). Recently, graph based SSL methods have gained interest (Alexandrescu and Kirchhoff, 2007), (Goldberg and Zhu, 2009). These methods create graphs whose vertices correspond to labeled and unlabeled data, while the edge weights encode the similarity between each pair of data points. Classification is performed using these graphs by scoring unlabeled points in such a way that instances connected by large weights are given similar labels. Such methods can perform well when no parametric information about distribution of data is available and when data is characterized by an underlying manifold structure.

$\underbrace{What}_{other}\ \underbrace{film}_{focus}\ \underbrace{introduced}_{event}\ \underbrace{Jar\ Jar\ Binks}_{topic}?$
Semantic Components & Named-Entitiy Types
topic: *'Jar' (Begin-Topic); 'Jar' (In-Topic) ; 'Binks' (In-Topic)(HUMAN:Individual)*
focus: *'film' (Begin-Focus) (DESCRIPTION:Definition)*
action / event: *'introduced' (Begin-Event)*
expected answer-type: *ENTITY:creative*

Table 1: Question Analysis - Semantic Components of a sample question from TREC QA task.

In this paper, we present a semantic component labeling module for our QA system using a new graph-based SSL to benefit from unlabeled questions. One of the issues affecting the performance of graph-based methods (Maier and Luxburg, 2008) is that there is no reliable approach for model selection when there are too few labeled points (Zhou et al., 2004). Such issues have only recently came into focus (Wang and Zhang, 2006). This is somewhat surprising because *graph construction* is a fundamental step. Rather than proposing yet another learning algorithm, we focus on graph construction for our labeling task, which suffers from insufficient graph sparsification methods. Such problems are caused by fixed neighborhood assignments in k-nearest neighbor approaches, treating sparse and denser regions of data equally or using improper threshold assumptions in ϵ-neighborhood

Proceedings of the NAACL HLT 2010 Workshop on Semantic Search, pages 27–35,
Los Angeles, California, June 2010.

graphs, yielding disconnected components or subgraphs or isolated singleton vertices. We propose a *Relaxed Linear Neighborhood* (RLN) method to overcome fixed k or ϵ assumptions. RLN approximates the entire graph by a series of overlapped linear neighborhood patches, where neighborhood $\mathcal{N}(x_i)$ of any node x_i is captured dynamically based on the density/sparsity of its surrounding. Moreover, RLN exploits *degree of neighborhood* during reconstruction method rather than fixed assignments, which does not get affected by outliers, producing a more robust graph, demonstrated in Experiment #1.

We present our question semantic component model in section 3 with the following contributions:
(1) a new *graph construction* method for SSL, which relaxes neighborhood assumptions yielding robust graphs as defined in section 5,
(2) a new *inference approach* to enable learning from unlabeled data as defined in section 6.
The experiments in section 7 yield performance improvement in comparison to other labeling methods on different datasets. Finally we draw conclusions.

2 Related Work on Question Analysis

An important step in question analysis is extracting semantic components like *answer type, focus, event*, etc. The 'answer-type' is a quantity that a question is seeking. A question *'topic'* usually represents major context/constraint of a question ("Jar Jar Binks" in Table 1). A question *'focus'* (e.g., film) denotes a certain aspect (or descriptive feature) of a question *'topic'*. To extract topic-focus from questions, (Hajicova et al., 1993) used rule-based approaches via dependency parser structures. (Burger, 2006) implemented parsers and a mixture of rule-based and learning methods to extract different salient features such as question type, event, entities, etc. (Chai and Jin, 2004) explored semantic units based on their discourse relations via rule-based systems.

In (Duan et al., 2008) a language model is presented to extract semantic components from questions. Similarly, (Fan et al., 2008)'s semantic chunk annotation uses conditional random fields (CRF) (Lafferty et al., 2001) to annotate semantic chunks of questions in Chinese. Our work aparts from these studies in that we use a graph-based SSL method to extract semantic components from unlabeled questions. Graph-based methods are suitable for labeling tasks because when two lexical units in different questions are close in the intrinsic geometry of question forms, their semantic components (labels) will be similar to each other. Labels vary smoothly along the geodesics, i.e., manifold assumption, which plays an essential role in SSL (Belkin et al., 2006).

This paper presents a new graph construction to improve performance of an important module of QA when labeled data is sparse. We compare our results with other graph construction methods. Next, we present the dataset construction for our semantic component labeling model before we introduce the new graph construction and inference for SSL.

3 Semantic Component Labeling

Each word (token) in a question is associated with a label among a pre-determined list of semantic tags. A question i is defined as a sequence of input units (words/tokens) $x_i = (x_{1i}, ..., x_{Ti}) \in \mathcal{X}^T$ which are tagged with a sequence of class labels, $y_i = (y_{1i}, ..., y_{Ti}) \in \mathcal{Y}^T$, semantic components. The task is to learn classifier $\mathcal{F}$ that, given a new sequence x^{new}, predicts a sequence of class labels $y^{new} = \mathcal{F}(x^{new})$. Among different semantic component types presented in previous studies, we give each token a MUC style semantic label from a list of 11 labels.
(1) O: $other$;
(2) BT:*begin-topic*;
(3) IT:*in-topic*
(4) BF:*begin-focus*;
(5) IF:*in-focus*
(6) BE:*begin-event*;
(7) IE:*in-event*
(8) BCL:*begin-clause*
(9) ICL:*in-clause*
(10) BC:*begin-complement*
(11) IC:*in-complement*
More labels can be appended if necessary. The first token of a component gets *'begin'* prefix and consecutive words are given *'in'* prefix, e.g., *Jar (begin-topic), Jar (in-topic), Binks (in-topic)* in Table 1.

In graph-based SSL methods, a graph is constructed $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V} = \mathcal{X}$ is a vertex set, $\mathcal{E}$ is an edge set, associated with each edge e_{ij} rep-

resents a relation between x_i and x_j. The task is to assign a label (out of 11 possible labels) to each token of a question i, x_{ti}, $t = 1, ..., T$, T is the max number of tokens in a given query. We introduce a set of nodes for each token (x_{ti}), each representing a binary relation between that token and one of possible tags (y_{ti}). A binary relation represents an agreement between a given token and assigned label, so our SSL classifier predicts the probability of true relation between token and assigned label. Thus, for each token, we introduce 11 different nodes using $y_k \in \{$*O,BT,IT,BF,IF,BC,IC,BE,IE,BCL,ICL*$\}$. There will be 11 label probability assignments obtained from each of the 11 corresponding nodes. For labeled questions, intuitively, only one node per token is introduced to the graph for known(true) token/label relations. We find the best question label sequence via Viterbi algorithm (Forney, 1973).

3.1 Feature Extraction For Labeling Task

The following pre-processing modules are built for feature extraction prior to graph construction.

3.1.1 Pre-Processing For Feature Extraction

Phrase Analysis(PA): Using basic syntactic analysis (shallow parsing), the PA module re-builds phrases from linguistic structures such as noun-phrases (NN), basic prepositional phrases (PP) or verb groups (VG). Using Stanford dependency parser (Klein and Manning, 2003), (Marneffe et al., 2006), which produces 48 different grammatical relations, PA module re-constructs the phrases. For example for the question in Table 1, dependency parser generates two relations:
– *nn(Binks-3, Jar-1)* and *nn(Binks-3, Jar-2)*,
PA reveals *"Jar Jar Binks"* as a noun phrase reconstructing the ***nn****:noun compound modifier*. We also extract part of speech tags of questions via dependency parser to be used for feature extraction.

Question Dependency Relations (QDR): Using shallow semantics, we decode underlying Stanford dependency trees (Marneffe et al., 2006) that embody linguistic relationships such as head-subject (H-S), head-modifier (complement) (H-M), head-object (H-O), etc. For example: *"How did Troops enter the area last Friday?"* is chunked as:
– Head (H): *enter* – Object (O): *area*
– Subject (S): *Troops* – Modifier (M): *last Friday*
Later, the feature functions (FF) are extracted based on generalized rules such as S and O's are usually considered topic/focus, H is usually an event, etc.

3.1.2 Features for Token-Label Pairs

Each node v_i in a graph $\mathcal{G}$ represents a relation of any token(word) i, x_{ti} to its label y_{ti}, denoted as a feature vector $x_{ti} \in \Re^d$. A list of feature functions are formed to construct multi-dimensional training examples. We extract mainly first and second order features to identify token-label relations, as follows:

Lexicon Features (LF): These features are over words and their labels along with information about words such as POS tags, etc. A sample first order lexicon feature, $z(y_t, x_{1:T}, t)$:

$$z = \begin{cases} 1 & \text{if } y_t = \text{(BE/IE) and POS}_{x_t}\text{=VB} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

is set to 1, if its assigned label y_t is of event type (BE/IE) and word's POS tag is VB(verb) (such token-label assignment would be correct). A similar feature is set to 1 if a word has 'VB' as its POS tag and it is a copula word, so it's correct label can only be "O:other". Nodes satisfying only this constraint and have a relation to "O" label get the value of '1'. Similar binary features are: if the word is a WH type (*query word*), if its POS tag is an article, if its POS tag is NN(P)(noun), IN, etc.

Compound Features (CF): These features exploit semantic compound information obtained from our PA and QDR modules, in which noun-phrases are labeled as focus/topics, or verb-phrases as event. For instance, if a token is part of a semantic compound, e.g., *subject*, identified via our QDR module, then for any of the 11 nodes generated for this token, if token-label is other than 'O(Other)', then such feature would be 1, and 0 otherwise. Similarly, if a word is part of a noun-phrase, then a node having a relation to any of the labels other than 'O/BE/IE' would be given the value 1, and 0 otherwise We eliminate inclusion of some nodes with certain labels such as words with "NN" tags are not usually considered *event*s.

Probability Feature Functions (PFF): We calculate word unigram and bigram frequencies from training samples to extract label conditional probabilities given a word, e.g., *P*(BT—"IBM"), *P*(O-BE—"Who founded"). When no match is found in

unigram and bigram label conditional probability tables for testing cases, we use unigram and bigram label probabilities given the POS tag of that word, e.g., *P*(BT—NNP), *P*(O-BE—"WP-VBD"). We extract 11 different features for each word corresponding to each possible label to form the probability features from unigram frequencies, max. of 11X11 features for bigram frequencies, where some bigrams are never seen in training dataset.

Second-Order Features (SOF): Such features denote relation between a token, tag and tag$_{-1}$, e.g.,:

$$z = \begin{cases} 1 & \text{if } y_{t-1} = \text{BT, } y_t = \text{IT and POS}_{x_t} = \text{NN} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

which indicates if previous label is a start of a topic tag (BT) and current POS tag is NN, then a node with a relation to label "In-Topic (IT)" would yield value '1'. For any given token, one should introduce 11^2 different nodes to represent a single property. In experiments we found that only a limited number of second order nodes are feasible.

4 Graph Construction for SSL

Let $\mathcal{X}_L = \{x_1, ..., x_l\}$ be labeled question tokens with associated labels $Y_L = \{y_1, ..., y_l\}^T$ and $\mathcal{X}_U = \{x_1, ..., x_u\}$ be unlabeled tokens, $\mathcal{X} = \mathcal{X}_L \cup \mathcal{X}_U$.

A weighted symmetric adjacency matrix $\mathcal{W}$ is formed in two steps with edges E in $\mathcal{G}$ where $\mathcal{W}_{ij} \in \Re^{nxn}$, and non-zero elements represent the edge weight between v_i and v_j. Firstly, similarity between each pair of nodes is obtained by a measure to create a full affinity matrix, $\mathcal{A} \in \Re^{nxn}$, using a kernel function, $\mathcal{A}_{ij} = k(x_i, x_j)$ as weight measure (Zhou et al., 2004) $w_{ij} \in \Re^{n \times n}$:

$$w_{ij} = exp\left(-\left\|x_i - x_j\right\| / 2\sigma^2\right) \quad (3)$$

Secondly, based on chosen graph sparsification method, a sparse affinity matrix is obtained by removing edges that do not convey with neighborhood assumption. Usually a k-nearest neighbor (kNN) or ϵ neighbor (ϵN) methods are used for sparsification.

Graph formation is crucial in graph based SSL since sparsity ensures that the predicted model remains efficient and robust to noise, e.g., especially in text processing noise is inevitable. ϵN graphs provide weaker performance than the k-nearest neighborhood graphs (Jebara et al., 2009). In addition, the issue with kNN sparsification of graph is that the number of neighbors is fixed at the start, which may cause fault neighborhood assumptions even when neighbors are far apart. Additionally, kernel similarity functions may not rate edge weights because they might be useful locally but not quite efficient when nodes are far apart. Next, we present *Relaxed Linear Neighborhoods* to address these issues.

5 Relaxed Linear Neighborhoods (RLN)

Instead of measuring pairwise relations (3), we use neighborhood information to construct $\mathcal{G}$. When building a sparse affinity matrix, we re-construct each node using a linear combination of its neighbors, similar to *Locally Linear Embedding* (Roweis and Saul, 2000) and *Linear Neighborhoods* (Wang and Zhang, 2006), and minimize:

$$\min \textstyle\sum_i ||x_i - \sum_{j:x_j \in \mathcal{N}(x_i)} w_{ij} x_j||^2 \quad (4)$$

where $\mathcal{N}(x_i)$ is neighborhood of x_i, and w_{ij} is the degree of contribution of x_j to x_i. In (4) each node can be optimally reconstructed using a linear combination of its neighborhood (Roweis and Saul, 2000). However, having fixed k neighbors at start of the algorithm can effect generalization of classifier and can also cause confusion on different manifolds.

We present novel *RLN* method to reconstruct each object (node) by using dynamic neighborhood information, as opposed to fixed k neighbors of (Wang and Zhang, 2006). RLN approximates entire graph by a series of overlapped *linear neighborhood patches*, where neighborhood $\mathcal{N}(x_i)$ of a node x_i is captured dynamically via its neighbor's density.

Boundary Detection: Instead of finding fixed k neighbors of each node x_i (Wang and Zhang, 2006), RLN captures boundary of each node $\mathcal{B}(x_i)$ based on neighborhood information and pins each node within this boundary as its neighbors. We define weight $\mathcal{W}$ matrix using a measure like (3) as a first pass sparsification. We identify neighbors for each node $x_i \in X$ and save information in boundary matrix, $\mathcal{B}$. kNN recovers its k neighbors using a similarity function, e.g., a kernel distance function, and instantiates via:

$$\mathcal{N}_{x_i;k}(x_j) = \left\{ \begin{array}{ll} 1 & d(x_i, x_{j_1}) < d(x_i, x_{j_2}) \\ 0 & otherwise \end{array} \right\} \quad (5)$$

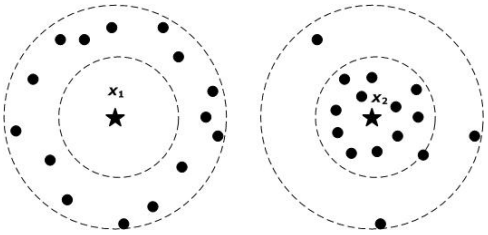

Figure 1: Neighborhood Boundary. Having same number of neighbors (n=15), boundaries of x_1 and x_2 are similar based on kNN (e.g., k=15), but dissimilar based on ϵN.

Similarly, with the ϵN approach the neighbors are instantiated when they are at most ϵ far away:

$$\mathcal{N}_{x_i;\epsilon}(x_j) = \left\{ \begin{array}{ll} 1 & d(x_i, x_j) < \epsilon \\ 0 & otherwise \end{array} \right\} \quad (6)$$

Both methods have limitations when sparsity or density is to concern. For sparse regions, if we restrict definition to k neighbors, then $\mathcal{N}(x_i)$ would contain dissimilar points. Similarly, improper threshold values could result in disconnected components or subgraphs or isolated singleton vertices. ϵ-radius would not define a graph because not every neighborhood radius would have the same density (see Fig. 1). Neighborhoods of two points (x_1, x_2) are different, although they contain same number of nodes.

We can use both *kNN* and ϵNN approaches to define the neighborhood between any x_i and x_j as:

$$\mathcal{N}_{x_i;k,\epsilon}(x_j) = \left\{ \begin{array}{ll} 1 & |\mathcal{N}_\epsilon(x_i)| > k \\ \mathcal{N}_{x_i;k}(x_j) & otherwise \end{array} \right\} \quad (7)$$

$|\mathcal{N}_\epsilon(x_i)|$ denotes cardinality of ϵ-neighbors of x_i, and $\mathcal{N}_{x_i;k}(x_j) \in \{0, 1\}$ according to (5). Thus if there are enough number of nodes in the ϵ vicinity ($> k$), then the boundary is identified. Otherwise we use kNN. Boundary set of any x_i is defined as:

$$\mathcal{B}(x_i) = \left\{ x_{j=1..n} \in X \,\middle|\, \mathcal{I}_{\mathcal{N}_{x_i;k,\epsilon}(x_j)=1} \right\} \quad (8)$$

Relaxed Boundary Detection: Adjusting boundaries based on a neighborhood radius and density might cause some problems. Specifically, if dense regions (clusters) exist and parameters are set large for sparse datasets, e.g., k and ϵ, then neighborhood sets would include more (and even noisy) nodes than necessary. Similarly, for low density regions if parameters are set for dense neighborhoods, weak neighborhood bonds will be formed to re-construct via linear neighborhoods. An algorithm that can handle a wide range of change interval would be advantageous. It should also include information provided by neighboring nodes closest to the corresponding node, which can take neighborhood relation into consideration more sensitively. Thus we extend neighborhood definition in (7) and (8) accounting for sensitivity of points with varying distances to neighbor points based on parameter $k > 0$:

$$\mathcal{N}_{x_i}(x_j) = max\left\{(1 - k\left(d(x_i, x_j)/d_{max}\right)), 0\right\} \quad (9)$$

$$\begin{array}{l} d_{max} = \max_{x_i, x_j \in X} d(x_i, x_j) \\ d(x_i, x_j) = \sqrt{\sum_{p=1}^{m}(x_{ip} - x_{jp})^2} \end{array} \quad (10)$$

In (10) m is the max. feature vector dimension of any x_i, k plays a role in determining neighborhood radius, such that it could be adjusted as follows:

$$1 - k\left(\epsilon/d_{max}\right) = 0 \Rightarrow k = d_{max}/\epsilon \quad (11)$$

The new boundary set of any given x_i includes:

$$\mathcal{B}(x_i) = \{x_{j=1..n} \in X \,|\mathcal{N}_{x_i}(x_j) \in [0, 1]\} \quad (12)$$

In the experiments, we tested our RLN approach (9), $0 < \mathcal{N}_{x_i}(x_j) < 1$ for boundary detection, in comparison to the static neighborhood assignments where the number of neighbors, k is fixed.

(3) Graph Formation: Instead of measuring pairwise relations as in (3), we use neighborhood information to represent $\mathcal{G}$. In an analogical manner to (Roweis and Saul, 2000), (Wang and Zhang, 2006), for graph sparcification, for our *Relaxed Linear Neighborhood*, we re-construct each node using a linear combination of its dynamic neighbors:

$$\begin{array}{l} \min_w \sum_i \left\| x_i - \sum_{j:x_j \in \mathcal{B}(x_i)} \mathcal{N}_{x_i}(x_j) w_{ij} x_j \right\|^2 \\ s.t. \sum_j w_{ij} = 1, w_{ij} \geq 0 \end{array} \quad (13)$$

where $0 < \mathcal{N}_{x_i}(x_j) < 1$ is the degree of neighborhood to boundary set $\mathcal{B}(x_i)$ and w_{ij} is degree of contribution of x_j to x_i, to be predicted. A $\mathcal{N}_{x_i}(x_j) = 0$ means no edge link. To prevent negative weights, and satisfy their normalization to unity, we used a constraint in (13) for RLN.

Edge weights of $\mathcal{G}$ are found using above relaxed boundary assumption, and relaxed neighborhood

method. A sparse relaxed weight matrix $(\tilde{\mathcal{W}})_{ij} = \tilde{w}_{ij}$ is formed representing different number of connected edges for every node, which are weighted according to their neighborhood density. Since w_{ij} is constructed via linear combination of varying number of neighbors of each node, $\tilde{\mathcal{W}}$ is used as the edge weights of $\mathcal{G}$. Next we form a regularization framework in place of label propagation (LP).

6 Regularization and Inference

Given a set of token-label assignments $\mathcal{X} = \{x_1, ..., x_l, x_{l+1}, ..., x_n\}$, and binary labels of first l points, $\mathcal{Y} = \{y_1, ..., y_l, 0, .., 0\}$, the goal is to predict if the label assignment of any token of a given test question is true or false. Let $\mathcal{F}$ denote set of classifying functions defined on $\mathcal{X}$, and $\forall \mathbf{f} \in \mathcal{F}$ a real value f_i to every point x_i is assigned. At each iteration, any given data point exploits a part of label information from its neighbors, which is determined by RLN. Thus, predicted label of a node x_i at t+1:

$$f_i^{t+1} = \lambda y_i + (1-\lambda) \textstyle\sum_j \mathcal{N}_{x_i}(x_j) w_{ij} f_j^t \quad (14)$$

where $x_j \in \mathcal{B}x_i$, $0< \lambda <1$ sets a portion of label information that x_i gets from its local neighbors, $\mathbf{f}^t = (f_1^t, f_2^t, ..., f_n^t)$ is the prediction label vector at iteration t and $f^0 = y$. We can re-state (14) as:

$$\mathbf{f}^{t+1} = \lambda y_i + (1-\lambda)\tilde{\mathcal{W}}\mathbf{f}^t \quad (15)$$

Each node's label is updated via (15) until convergence, which might be at $t \rightarrow \infty$. In place of LP, we can develop a regularization framework (Zhou et al., 2004) to learn $\mathbf{f}$. In graph-based SSL, a function over a graph is estimated to satisfy two conditions: (*i*) close to the observed labels , and (*ii*) be smooth on the whole graph via following loss function:

$$\begin{aligned} arg\min \mathcal{Q}(f) = \textstyle\sum_{i=1}^n (f_i - y_i)^2 + \\ \lambda \textstyle\sum_{i,j=1}^n \sum_{j:x_j \in \mathcal{B}(x_i)} \phi_{x_i}(x_j) \langle f_i, f_j \rangle \end{aligned} \quad (16)$$

where $\phi_{x_i}(x_j) = \mathcal{N}_{x_i}(x_j)\tilde{w}_{ij}$. Setting gradient of loss function $\mathcal{Q}(f)$ to zero, we obtain:

$$\partial_f \mathcal{Q}(f) = 2(\mathcal{Y} - \mathbf{f}) + \lambda[(\mathcal{I} - \Phi) + (\mathcal{I} - \Phi)^T]\mathbf{f} \quad (17)$$

Relaxed weight matrix $\tilde{\mathcal{W}}$ is normalized according to constraint in (13), so as degree matrix, $\mathcal{D} = \sum_j \tilde{\mathcal{W}}_{ij}$, and graph Laplacian, i.e., $\mathcal{L} = (\tilde{\mathcal{D}} - \tilde{\mathcal{W}})/\tilde{\mathcal{D}} = \mathcal{I} - \tilde{\mathcal{W}}$. Since f is a function on the manifold and the graph is discretized form of a manifold (Belkin and Niyogi, 2002a), $\mathbf{f}$ can also be regarded as the discrete form of f, which is equivalent at the nodes of graph. So the second term of (16) yields:

$$[(\mathcal{I} - \tilde{\mathcal{W}}) + (\mathcal{I} - \tilde{\mathcal{W}})^T]\mathbf{f} \approx 2\mathcal{L}f \approx [(\mathcal{I} - \tilde{\mathcal{W}})]\mathbf{f} \quad (18)$$

Hence optimum f^* is obtained by new form of derivative in (17) after replacing (18):

$$f^* = (1-\lambda)\left(\mathcal{I} - \lambda\tilde{\mathcal{W}}\right)^{-1}\mathcal{Y} \quad (19)$$

Most graph-based SSLs are transductive, i.e., not easily expendable to new testing points. In (Delalleau et al., 2005) an induction scheme is proposed to classify a new point x_{Te} by

$$\hat{f}(x_{Te}) = \textstyle\sum_{i \in L \cup U} \tilde{\mathcal{W}}_{x_i} f_i / \sum_{i \in L \cup U} \tilde{\mathcal{W}}_{x_i} \quad (20)$$

Thus, we use induction, where we can, to avoid reconstruction of the graph for new test points.

7 Experiments and Discussions

In the next, we evaluate the performance of the proposed RLN in comparison to the other methods on syntactic and real datasets.

Exp. 1. Graph Construction Performance: Here we use a similar syntactic data in (Jebara et al., 2009) shown in Fig.2.a, which contains two clusters of dissimilar densities and shapes. We investigate three graph construction methods, linear *k*-neighborhoods of (Roweis and Saul, 2000) in Fig.2.b, *b*-matching(Jebara et al., 2009) in Fig.2.c and RLN of this work in Fig.2.d using a dataset of 300 points with binary output values. b-matching permits a given datum to select k neighboring points but also ensures that exactly k points selects given datum as their neighbor.

In each graph construction method Gaussian kernel distance is used. Experiments are run 50 times where at each fold only 2 labeled samples from opposite classes are used to predict the rest. The experiments are repeated for different *k*, *b* and ϵ values. In Fig. 2, average of trials is shown when *k*, *b* are 10 and ϵ *>0.5*. We also used the ϵN approach but it did not show any improvement over *k*NN approach.

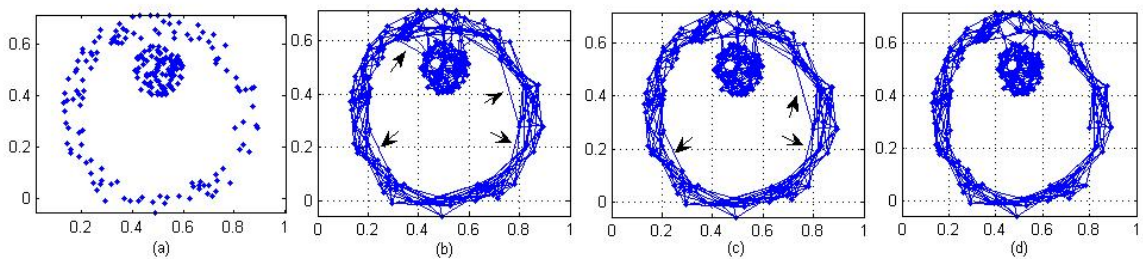

Figure 2: Graph Construction Experiments. (a) Syntactic data. (b) linear k-neighborhood (c) b-matching (d) RLN.

In Fig. 2.d, RLN can separate two classes more efficiently than the rest. Compared to the *b*-matching approach, RLN clearly improves the robustness. There are more links between clusters in other graph methods than RLN, which shows that RLN can separate two classes much efficiently. Also since dynamic number of edges are constructed with RLN, unnecessary links are avoided, but for the rest of the graph methods there are edges between far away nodes (shown with arrows). In the rest of the experiments, we use b-matching for benchmark as it is the closest approach to the proposed RLN.

Exp. 2. Semantic Component Recognition: We demonstrate the performance of the new RLN with two sets of experiments for sequence labeling of question recognition task. As a first step in understanding semantic components of questions, we asked two annotators to annotate a random subsample of 4000 TREC factoid and description questions obtained from tasks of 1999-2006. There are 11 predefined semantic categories (section 3), close to 280K labeled tokens. Annotators are told that each question must have one topic and zero or one focus and event, zero or more of the rest of the components. Inter-tagger agreement is $\kappa = 0.68$, which denotes a considerable agreement.

We trained models on 3500 random set of questions and reserved the rest of 500 for testing the performance. We applied pre-processing and feature selection of section 3 to compile labeled and unlabeled training and labeled testing datasets. At training time, we performed manual iterative parameter optimization based on prediction accuracy to find the best parameter sets, i.e., $k = \{3, 5, 10, 20, 50\}$, $\epsilon \in \{0, 1\}$, distance = $\{linear, gaussion\}$.

We use the average loss ($\bar{L}$) per sequence (query) to evaluate the semantic chunking performance:

$$\bar{L} = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{1}{L_i} \sum_{j=1}^{L_i} \mathcal{I}\left((\hat{y}_i)_j \neq (y_i)_j\right) \right] \quad (21)$$

where $\hat{y}$ and y are predicted and actual sequence respectively; N is the number of test examples; L_i is the length of i^{th} sequence; $\mathcal{I}$ is the 0-1 loss function.

	other	*topic*	*focus*	*event*	*rest*
# Samples	1997	1142	525	264	217
CRF	0.935	0.903	0.823	0.894	0.198
b-matching	0.871	0.900	0.711	0.847	0.174
RLN	0.911	0.910	0.761	0.834	0.180

Table 2: Chunking accuracy on testing data. *'other'*=O, *'topic'*=BT+IT, *'focus'* = BF+IF, *'event'*= 'BE+IE", *'rest'*= rest of the labels, i.e., IE, BC, IC, BCL, ICL.

(1) *Chunking Performance:* Here, we investigate the accuracy of our models on individual component prediction. We use CRF, b-matching and our RLN to learn models from labeled training data and evaluate performance on testing dataset. For RLN and b-matching we use training as labeled and testing as unlabeled dataset in transductive way to predict token labels. The testing results are shown in Table 2 for different group of components. The accuracy for 'topic' and 'focus' components are relatively high compared to other components. Most of the errors on the 'rest' labels are due to confusion with 'topic' or 'focus'. On some components, i.e., *topic*, *other*, RLN performed significantly better than b-matching based on t-test statistics (at 95% confidence). No statistical significance between CRF and RLN is observed indicating that RLN's good performance on individual label scoring, as it shows that RLN can be used efficiently for sequence labeling.

(2) *Question Labeling Performance.* Having

Labeled	CRF	SSL	sCRF	b-match	RLN
1%	0.240	0.235	0.223	0.233	**0.220**
5%	0.222	0.218	0.215	0.203	**0.189**
10%	0.170	0.219	0.186	0.194	0.180
25%	0.173	0.196	0.175	0.174	**0.170**
50%	0.160	0.158	0.147	0.156	0.158
75%	0.140	0.163	0.138	0.160	0.155
100%	0.120	0.170	0.123	0.155	0.149

Table 3: Test Data Average Loss on graph construction with RLN, b-matching, standard SSL with kNN as well as CRF, CRF with Self Learning (sCRF).

demonstrated that RLN is an alternative method to the standard sequence learning methods for the question labeling task, next we evaluate per sequence (question) performance, rather than individual label performance using unlabeled data. Firstly, we randomly select subset of labeled training dataset, $X_L^i \subset X_L$ with different sample sizes, $n_L^i = 5\% * n_L$, $10\% * n_L$, $25\% * n_L$, $50\% * n_L$, $75\% * n_L$, $100\% * n_L$, where n_L is the size of X_L. Thus, instead of fixing the number of labeled records and varying the number of unlabeled points, we propose to fix the percentage of unlabeled points in training dataset. We hypothetically use unselected part of the labeled dataset as unlabeled data at each random selection. We compare the result of RLN to other graph based methods including standard SSL (Zhu et al., 2003) using kNN, and b-matching. We also build a CRF model using the same features as RLN except the output information, which CRF learns through probabilistic structure. In addition, we implemente self training for CRF (sCRF), most commonly known SSL method, by adding most confident $(x, f(x))$ unlabeled data back to the data and repeat the process 10 times. Table 3 reports average loss of question recognition tasks on testing dataset using these methods.

When the number of labeled data is small ($n_L^i < 25\% n_L$), RLN has better performance compared to the rest (an average of 7% improvement). The SSL and sCRF performance is slightly better than CRF at this stage. As expected, as the percentage of labeled points in training is increased, the CRF outperforms the rest of the models. However, observing no statistical significance between CRF, b-matching and RLN up to 25-50% labeled points indicates RLNs performance on unlabeled datasets. Thus, for sequence labeling, the RLN can be a better alternative to known sequence labeling methods, when manual annotation of the entire dataset is not feasible.

# Unlabeled tokens	25K	50K	75K	100K
Average Loss	0.150	0.146	0.141	0.139

Table 4: Average Loss Results for RLN graph based SSL as unlabeled tokens is increased.

Exp. 3. Unlabeled Data Performance: Here we evaluate the effect of the size of unlabeled data on the performance of RLN by gradually increasing the size of unlabeled questions. The assumption is that as more unlabeled data is used, the model would have additional spatial information about token neighbors that would help to improve its generalization performance. We used the questions from the Question and Answer pair dataset distributed by Linguistic Data Consortium for the DARPA GALE project (LDC catalog number: LDC2008E16). We compiled 10K questions, consisting of 100K tokens.

Although the error reduction is small (Table 4), the empirical results indicate that unlabeled data can have positive effect on the performance of the RLN method. As we introduce more unlabeled data, the RLN performance is increased, which indicates that there is a lot to discover from unlabeled questions.

8 Conclusions

In this paper, we presented a graph-based semi-supervised learning method with a new graph construction. Our new graph construction relaxes the neighborhood assumptions yielding robust graphs when the labeled data is sparse, in comparison to previous methods, which set rigid boundaries. The new algorithm is particularly appealing to question semantic component recognition task, namely question understanding, in that in this task we usually deal with very few labeled data and considerably larger unlabeled data. Experiments on question semantic component recognition show that our semi-supervised graph-based method can improve performance by up to 7-10% compared to well-known sequence labeling methods, especially when there are more unlabeled data than the labeled data.

References

A. Alexandrescu and K. Kirchhoff. 2007. Data-driven graph construction for semi-supervised graph-based learning in nlp. In *Proc. of HLT 2007*.

M. Belkin and P. Niyogi. 2002a. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*.

M. Belkin and P. Niyogi. 2002b. Using manifold structure for partially labeled classification. In *Proc. of NIPS 2002*.

M. Belkin, P. Niyogi, and V. Sindhwani. 2006. A geometric framework for learning from examples. In *Journal of Machine Learning Research*.

J. D. Burger. 2006. Mitre's qanda at trec-15. In *Proc. of the TREC-2006*.

J.Y. Chai and R. Jin. 2004. Discourse structure for context question answering. In *Proc. of HLT-NAACL 2004*.

O. Delalleau, Y. Bengio, and N.L. Roux. 2005. Efficient non-parametric function induction in semi-supervised learning. In *Proc. of AISTAT-2005*.

H. Duan, Cao Y, C.Y. Lin, and Y. Yu. 2008. Searching questions by identifying question topic and question focus. In *Proc. of ACL-08*.

S. Fan, Y. Zhang, W.W.Y. Ng, Xuan Wang, and X. Wang. 2008. Semantic chunk annotation for complex questions using conditional random field. In *Coling 2008: Proc. of Workshop on Knowledge and Reasoning for Answering Questions*.

GD. Forney. 1973. The viterbi algorithm. In *Proc. of IEEE 61(3)*, pages 269–278.

A. Goldberg and X. Zhu. 2009. Keepin' it real: Semi-supervised learning with realistic tuning. In *Proc. of NAACL-09 Workshop on Semi-Supervised Learning for NLP*.

E. Hajicova, P. Sgall, and H. Skoumalova. 1993. Identifying topic and focus by an automatic procedure. In *Proc. of the EACL-1993*.

T. Jebara, J. Wang, and S.F. Chang. 2009. Graph construction and b-matching for semi-supervised learning. In *Proc. of ICML-09*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the ACL-2003*, pages 423–430.

J.D. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of 18th International Conf. on Machine Learning (ICML'01)*.

M. Maier and U.V. Luxburg. 2008. Influence of graph construction on graph-based clustering measures. In *Proc. of Neural Infor. Proc. Sys. (NIPS 2008)*.

M.-C.D. Marneffe, B. MacCartney, and C.D. Manning. 2006. Generating typed-dependency parsers from phrase structure parsers. In *In LREC2006*.

S.T. Roweis and L.K. Saul. 2000. Nonlinear dimensionality reduction by locally embedding. In *Science*, volume 290, pages 2323–2326.

F. Wang and C. Zhang. 2006. Label propagation through linear neighborhoods. In *Proc. of the ICML-2006*.

Dengyong Zhou, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16:321–328.

Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. 2003. *Semi-supervised learning: From Gaussian Fields to Gaussian processes*. Technical Report CMU-CS-03-175, Carnegie Mellon University, Pittsburgh.