Graph-based Learning for Statistical Machine Translation

Andrei Alexandrescu
Dept. of Comp. Sci. Eng.
University of Washington
Seattle, WA 98195, USA
andrei@cs.washington.edu

Katrin Kirchhoff
Dept. of Electrical Eng.
University of Washington
Seattle, WA 98195, USA
katrin@ee.washington.edu

Abstract

Current phrase-based statistical machine translation systems process each test sentence in isolation and do not enforce global consistency constraints, even though the test data is often internally consistent with respect to topic or style. We propose a new consistency model for machine translation in the form of a graph-based semi-supervised learning algorithm that exploits similarities between training and test data and also similarities between different test sentences. The algorithm learns a regression function jointly over training and test data and uses the resulting scores to rerank translation hypotheses. Evaluation on two travel expression translation tasks demonstrates improvements of up to 2.6 BLEU points absolute and 2.8% in PER.

1 Introduction

Current phrase-based statistical machine translation (SMT) systems commonly operate at the sentence level—each sentence is translated in isolation, even when the test data consists of internally coherent paragraphs or stories, such as news articles. For each sentence, SMT systems choose the translation hypothesis that maximizes a combined log-linear model score, which is computed independently of all other sentences, using globally optimized combination weights. Thus, similar input strings may be translated in very different ways, depending on which component model happens to dominate the combined score for that sentence. This is illustrated by the following example (from the IWSLT 2007 Arabic-English translation task):

Source 1: Asf 1A ymknk *lk hnAk klfp HwAly vmAnyndwlAr lAIsAEp AlwAHdp
Ref: sorry you can’t there is a cost the charge is eighty dollars per hour
1-best: i’m sorry you can’t there in the cost about eighty dollars for a one o’clock

Source 2: E*rA lA ymknk t$gyl AltlfAz HtY tqIE AITA}rp
Ref: sorry you cannot turn the tv on until the plane has taken off
1-best: excuse me i you turn tv until the plane departs

The phrase 1A ymknk (you may not/you cannot) is translated differently (and wrongly in the second case) due to different segmentations and phrase translations chosen by the decoder. Though different choices may be sometimes appropriate, the lack of constraints enforcing translation consistency often leads to suboptimal translation performance. It would be desirable to counter this effect by encouraging similar outputs for similar inputs (under a suitably defined notion of similarity, which may include e.g. a context specification for the phrase/sentence).

In machine learning, the idea of forcing the outputs of a statistical learner to vary smoothly with the underlying structure of the inputs has been formalized in the graph-based learning (GBL) framework. In GBL, both labeled (train) and unlabeled (test) data samples are jointly represented as vertices in a graph whose edges encode pairwise similarities between samples. Various learning algorithms can be applied to assign labels to the test samples while ensuring that the classification output varies smoothly...
along the manifold defined by the graph. GBL has been successfully applied to a range of problems in computer vision, computational biology, and natural language processing. However, in most cases, the learning tasks consisted of unstructured classification, where the input was represented by fixed-length feature vectors and the output was one of a finite set of discrete labels. In machine translation, by contrast, both inputs and outputs consist of word strings of variable length, and the number of possible outputs is not fixed and practically unlimited.

In this paper we propose a new graph-based learning algorithm with structured inputs and outputs to improve consistency in phrase-based statistical machine translation. We define a joint similarity graph over training and test data and use an iterative label propagation procedure to regress a scoring function over the graph. The resulting scores for unlabeled samples (translation hypotheses) are then combined with standard model scores in a log-linear translation model for the purpose of reranking. Our contributions are twofold. First, from a machine translation perspective, we design and evaluate a global consistency model enforcing that similar inputs receive similar translations. Second, from a machine learning perspective, we apply graph-based learning to a task with structured inputs and outputs, which is a novel contribution in itself since previous applications of GBL have focused on predicting categorical labels. We evaluate our approach on two machine translation tasks, the IWSLT 2007 Italian-to-English and Arabic-to-English tasks, and demonstrate significant improvements over the baseline.

2 Graph-Based Learning

GBL algorithms rely on a similarity graph consisting of a set of nodes representing data samples \( x_i \) (where \( i \) ranges over \( 1, \ldots, l \) labeled points and \( l+1, \ldots, n \) unlabeled points), and a set of weighted edges encoding pairwise similarities between samples. The graph is characterized by a weight matrix \( W \) whose elements \( W_{ij} \geq 0 \) are the similarity values for edges between vertices \( x_i \) and \( x_j \), and by its label vector \( Y = (y_1, \ldots, y_l), y_i \in \{1, \ldots, C\} \) that defines labels for the first \( l \) points. If there is no edge linking nodes \( x_i \) and \( x_j \), then \( W_{ij} = 0 \). There is considerable freedom in choosing the weights. The similarity measure used to compute the edge weights determines the graph structure and is the most important factor in successfully applying GBL. In most applications of GBL, data samples are represented by fixed-length feature vectors, and cosine similarity or Euclidean distance-based measures are used for edge weights.

Learning algorithms on similarity graphs include e.g. min-cut (Blum and Chawla, 2001), spectral graph transducer (Joachims, 2003), random walk-based approaches (Szummer and Jaakkola, 2001), and label propagation (Zhu and Ghahramani, 2002). The algorithm proposed herein is based on the latter.

2.1 Label Propagation

Given a graph defined by a weight matrix \( W \) and a label set \( Y \), the basic label propagation algorithm proceeds as follows:

1. Initialize the matrix \( P \) as \( P_{ij} = \frac{W_{ij}-W_{ii}}{\sum_j W_{ij}-W_{ii}} \)
2. Initialize a \( n \times C \) matrix \( f \) with binary vectors encoding the known labels for the first \( l \) rows: \( f_i = \delta_C(y_i) \forall i \in \{1, 2, \ldots, l\} \), where \( \delta_C(y_i) \) is the Kronecker vector of length \( C \) with 1 in position \( y_i \) and 0 elsewhere. The remaining rows of \( f \) can be zero.
3. \( f' \leftarrow P \times f \)
4. Clamp already-labeled data rows: \( f'_i = \delta_C(y_i) \forall i \in \{1, 2, \ldots, l\} \)
5. If \( f' \cong f \), stop.
6. \( f \leftarrow f' \)
7. Repeat from step 3.

After convergence, \( f \) contains the solution in rows \( l+1 \) to \( n \) in the form of unnormalized label probability distributions. Hard labels can be obtained by

\[
\hat{y}_i = \arg \max_{j \in \{1, \ldots, C\}} f_{ij} \quad \forall i \in \{l+1, \ldots, n\} \quad (1)
\]

The algorithm minimizes the following cost function (Zhu, 2005):

\[
S = \sum_{k=1}^{C} \sum_{i \geq l \land j \geq l} W_{ij}(f_{ik} - f_{jk})^2 \quad (2)
\]

\( S \) measures the smoothness of the learned function, i.e., the extent to which the labeling allows large-weight edges to link nodes of different labels. By minimizing \( S \), label propagation finds a labeling
that, to the extent possible, assigns similar soft labels (identical hard labels) to nodes linked by edges with large weights (i.e., highly similar samples). The labeling decision takes into account not only similarities between labeled and unlabeled nodes (as in nearest-neighbor approaches) but also similarities among unlabeled nodes. Label propagation has been used successfully for various classification tasks, e.g., image classification and handwriting recognition (Zhu, 2005). In natural language processing, label propagation has been used for document classification (Zhu, 2005), word sense disambiguation (Niu et al., 2005; Alexandrescu and Kirchhoff, 2007), and sentiment categorization (Goldberg and Zhu, 2006).

3 Graph-Based Learning for Machine Translation

Our goal is to exploit graph-based learning for improving consistency in statistical phrase-based machine translation. Intuitively, a set of similar source sentences should receive similar target-language translations. This means that similarities between training and test sentences should be taken into account, but also similarities between different test sentences, which is a source of information currently not exploited by standard SMT systems. To this end we define a graph over the training and test sets with edges between test and training sentences as well as between different test sentences. In cases where a test sentence does not have any connections to training sentences but is connected to other test sentences, helpful information about preferred translations can be propagated via these edges.

As mentioned above, the problem of machine translation does not neatly fit into the standard GBL framework. Given that our samples consist of variable-length word strings instead of feature vectors, the standard cosine or Euclidean-distance based similarity measures cannot be used meaningfully, and the number of possible “labels”—correct translations—is unbounded and practically very large. We thus need to modify both the graph construction and the label propagation algorithms.

First, we handle the problem of unlimited outputs by applying GBL to rescoring only. In most SMT systems, an $N$-best list (generated by a first decoding pass) approximates the search space of good hypotheses reasonably well, provided $N$ is large enough. For all hypotheses of all sentences in the test set (set we denote with $H$), the system learns a ranking function $r : H \rightarrow [0, 1]$. Larger values of $r$ indicate better hypotheses. The corresponding loss functional is

$$\mathcal{L}(r) = \sum_{i,j} W_{ij} [r(x_i) - r(x_j)]^2$$ (3)

$\mathcal{L}(r)$ measures the smoothness of $r$ over the graph by penalizing highly similar clusters of nodes that have a high variance of $r$ (in other words, similar input sentences that have very different translations). The smaller $\mathcal{L}(r)$, the “smoother” $r$ is over the graph. Thus, instead of directly learning a classification function, we learn a regression function—similar to (Goldberg and Zhu, 2006)—that is then used for ranking the hypotheses.

3.1 Graph Construction

Each graph node represents a sentence pair (consisting of source and target strings), and edge weights represent the combined similarity scores computed from comparing both the source sides and target sides of a pair of nodes. Given a training set with $l$ source and target language sentence pairs $(s_1, t_1), \ldots, (s_l, t_l)$ and a test set with $l + 1, \ldots, n$ source sentences, $s_{l+1}, \ldots, s_n$, the construction of the similarity graph proceeds as follows:

1. For each test sentence $s_i$, $i = l + 1, \ldots, n$, find a set $S_{train_i}$ of similar training source sentences and a set $S_{test_i}$ of similar test sentences (excluding $s_i$ and sentences identical to it) by applying a string similarity function $\sigma$ to the source sides only and retaining sentences whose similarity exceeds a threshold $\theta$. Different $\theta$’s can be used for training vs. test sentences; we use the same $\theta$ for both sets.

2. For each hypothesis $h_s$, generated for $s_i$ by a baseline system, compute its similarity to the target sides of all sentences in $S_{train_i}$. The overall similarity is then defined by the combined score

$$\alpha_{ij} = \kappa(\sigma(s_i, s^j), \sigma(h_s, t^j))$$ (4)

where $i = l + 1, \ldots, n$, $j = 1, \ldots, |S_{train_i}|$ and $\kappa : \mathbb{R}_+ \times \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is an averaging function.
If \( \alpha_{ij} > 0 \), establish graph nodes for \( h_{si} \) and \( t_j \) and link them with an edge of weight \( \alpha_{ij} \).

3. For each hypothesis \( h_{si} \) and each hypothesis generated for each of the sentences \( s_k \in \Sigma_{test} \), compute similarity on the target side and use the combined similarity score as the edge weight between nodes for \( h_{si} \) and \( h_{sk} \).

4. Finally, for each node \( x_t \) representing a training sentence, assign \( r(x_t) = 1 \) and also define its synthetic counterpart: a vertex \( x'_t \) with \( r(x'_t) = 0 \). For each edge incident to \( x_t \) of weight \( W_{th} \), define a corresponding edge of weight \( 1 - W_{th} \).

The synthetic nodes and edges need to be added to prevent the label propagation algorithm from converging to the trivial solution that assigns to prevent the label propagation algorithm from converging to the trivial solution that assigns.

A more general way of computing similarity between strings is provided by string kernels (Lodhi et al., 2002; Rousu and Shawe-Taylor, 2005), which have been extensively used in bioinformatics and email spam detection. String kernels map strings into a feature space defined by all possible substrings of the string up a fixed length \( k \), and computing the dot product between the resulting feature vectors. Several variants of basic string kernels exist, notably those allowing gaps or mismatches, and efficient implementations have been devised even for large scale applications. Formally, we define a sentence \( s \) as a concatenation of symbols from a finite alphabet \( \Sigma \) (the vocabulary of the language) and an embedding function from strings to feature vectors, \( \phi : \Sigma^* \rightarrow \mathcal{H} \). A kernel function \( K(s, t) \) computes the distance between the resulting vectors for two sentences \( s \) and \( t \). In our case, the embedding function is defined as

\[
\phi_k^u(s) := \sum_{i: u = s(i)} \lambda^{|i|} \quad u \in \Sigma^k
\]

where \( k \) is the maximum length of substrings, \( |i| \) is the length of \( i \), and \( \lambda \) is a penalty parameter for each gap encountered in the substring. \( \mathcal{K} \) is defined as

\[
\mathcal{K}(s, t) = \sum_u \langle \phi_u(s), \phi_u(t) \rangle w_u
\]

where \( w \) is a weight dependent on the length of the substring \( u \). Finally, the kernel score is normalized by \( \sqrt{\mathcal{K}(s, s) \cdot \mathcal{K}(t, t)} \) to discourage long sentences from being favored. Thus, our similarity measure is a gapped, normalized string kernel, which is a more general measure than BLEU in that it considers non-contiguous substrings. We use a dynamic programming implementation of string kernels (Rousu and Shawe-Taylor, 2005).
For the combination of source-side and target-side similarity scores (the function we denoted as $\kappa$) we test two simple schemes, using either the geometric or the arithmetic mean of the individual scores. In the first case, large edge weights only result when both source and target are close to each other; the latter may produce high edge weights when only one of them (typically the source score) is high. More sophisticated combination schemes, using e.g. weighted combination, could be used but were not investigated in this study.

**Scalability** Poor scalability is often mentioned as a drawback of graph-based learning. Straightforward implementations of GBL algorithms often represent the joint training and test data in working memory and therefore do not scale well to large data sets. However, we have developed several techniques to improve scalability without impeding accuracy. First, we construct separate graphs for each techniques to improve scalability without impeding accuracy. First, we construct separate graphs for each test sentence without losing global connectivity information. The graph for a test sentence is computed as the transitive closure of the edge set $E$ over the nodes containing all hypotheses for that test sentence. This smaller graph does not affect the outcome of the learning process for the chosen sentence because in label propagation the learned value $r(x_i)$ can be influenced by that of another node $x_j$ if and only if $x_j$ is reachable from $x_i$. In the worst theoretical case, the transitive closure could comprehend the entire graph, but in practice the edge set is never that dense and can be easily pruned based on the heuristic that faraway nodes connected through low-weight edges have less influence on the result. We use a simple embodiment of this heuristic in a work-list approach: starting from the nodes of interest (hypotheses for the focal sentence), we expand the closure starting with the direct neighbors, which have the largest influence; then add their neighbors, which have less influence, and so forth. A threshold on the number of added vertices limits undue expansion while capturing either the entire closure or a good approximation of it. Another practical computational advantage of portioning work is that graphs for different hypothesis sets can be trivially created and used in parallel, whereas distributing large matrix-vector multiplication is much more difficult (Choi, 1998). The disadvantage is that overall redundant computations are being made: incomplete estimates of $r$ are computed for the ancillary nodes in the transitive closure and then discarded.

Second, we obtain a reduction in graph size of orders of magnitude by collapsing all training vertices of the same $r$ that are connected to the same test vertex into one and sum the edge weights. This is equivalent to the full graph for learning purposes.

### 3.2 Propagation

Label propagation proceeds as follows:

1. Compute the transitive closure over the edges starting from all hypothesis nodes of a given sentence.
2. On the resulting graph, collapse all test-train similarities for each test node by summing edge weights. Obtain accumulated similarities in row and column 1 of the similarity matrix $W$.
3. Normalize test-to-train weights such that $\sum_j W_{1j} = \sum_j W_{j1} = 1$.
4. Initialize the matrix $P$ as $P_{ij} = \frac{W_{ij}}{1 - W_{i1} + \sum_j W_{ij}}$.
   (The quantity $1 - W_{i1}$ in the denominator is the weight of the dissimilarity edge.)
5. Initialize a column vector $f$ of height $n$ with $f_1 = 1$ (corresponding to node $x_1$) and 0 in the remaining positions.
6. $f' \leftarrow P \times f$
7. Clamp $f'_1$: $f'_1 = 1$
8. If $f' \cong f$, continue with step 11.
9. $f \leftarrow f'$
11. The result $r$ is in the slots of $f$ that correspond to the hypotheses of interest. Normalize per sentence if needed, and rank in decreasing order of $r$. 

![Figure 1: Connections for hypothesis node $x_h$. Similarity edges with weights $W_{ih}$ link the node with train sentences $x_i$, for which $r(x_i) = 1$. For each of these edges we define a dissimilarity edge of weight $1 - W_{ih}$, linking the node with node $x'_i$ for which $r(x'_i) = 0$. The vertex is also connected to other test vertices (the dotted edges).](image)
Convergence Our algorithm's convergence proof is similar to that for standard label propagation (Zhu, 2005, p. 6). We split \( P \) as follows:

\[
P = \begin{bmatrix} 0 & P_{LU} \\ P_{UL} & P_{UU} \end{bmatrix}
\]  

(7)

where \( P_{UL} \) is a column vector holding global similarities of test hypotheses with train sentences, \( P_{LU} \) is a horizontal vector holding the same similarities (though \( P_{LU} \neq P_{UL}^T \) due to normalization), and \( P_{UU} \) holds the normalized similarities between pairs of test hypotheses. We also separate \( f \):

\[
f = \begin{bmatrix} 1 \\ f_U \end{bmatrix}
\]  

(8)

where we distinguish the first entry because it represents the training part of the data. With these notations, the iteration formula becomes:

\[
f_U' = P_{UU}f_U + P_{UL}
\]  

(9)

Unrolling the iteration yields:

\[
f_U = \lim_{n \rightarrow \infty} \left[ (P_{UU})^n f_U + \left( \sum_{i=1}^{n} (P_{UU})^{i-1} \right) P_{UL} \right]
\]

It can be easily shown that the first term converges to zero because of normalization in step 4 (Zhu, 2005). The sum in the second term converges to \((I - P_{UU})^{-1}\), so the unique fixed point is:

\[
f_U = (I - P_{UU})^{-1} P_{UL}
\]  

(10)

Our system uses the iterative form. On the data sets used, convergence took 61.07 steps on average.

At the end of the label propagation algorithm, normalized scores are obtained for each N-best list (sentences without any connections whatsoever are assigned zero scores). These are then used together with the other component models in log-linear combination. Combination weights are optimized on a held-out data set.

4 Data and System

We evaluate our approach on the IWSLT 2007 Italian-to-English (IE) and Arabic-to-English (AE) travel tasks. The first is a challenge task, where the training set consists of read sentences but the development and test data consist of spontaneous dialogues. The second is a standard travel expression translation task consisting entirely of read input. For our experiments we chose the text input (correct transcription) condition only. The data set sizes are shown in Table 1. We split the IE development set into two subsets of 500 and 496 sentences each. The first set (dev-1) is used to train the system parameters of the baseline system and as a training set for GBL. The second is used to tune the GBL parameters. For each language pair, the baseline system was trained with additional out-of-domain text data: the Italian-English Europarl corpus (Koehn, 2005) in the case of the IE system, and 5.5M words of newswire data (LDC Arabic Newswire, Multiple-Translation Corpus and ISI automatically extracted parallel data) in the case of the AE system.

<table>
<thead>
<tr>
<th>Set</th>
<th># sent pairs</th>
<th># words</th>
<th># refs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE train</td>
<td>26.5K</td>
<td>160K</td>
<td>1</td>
</tr>
<tr>
<td>IE dev-1</td>
<td>500</td>
<td>4308</td>
<td>1</td>
</tr>
<tr>
<td>IE dev-2</td>
<td>496</td>
<td>4204</td>
<td>1</td>
</tr>
<tr>
<td>IE eval</td>
<td>724</td>
<td>6481</td>
<td>4</td>
</tr>
<tr>
<td>AE train</td>
<td>23K</td>
<td>160K</td>
<td>1</td>
</tr>
<tr>
<td>AE dev4</td>
<td>489</td>
<td>5392</td>
<td>7</td>
</tr>
<tr>
<td>AE dev5</td>
<td>500</td>
<td>5981</td>
<td>7</td>
</tr>
<tr>
<td>AE eval</td>
<td>489</td>
<td>2893</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 1: Data set sizes and reference translations count.

Our baseline is a standard phrase-based SMT system based on a log-linear model with the following feature functions: two phrase-based translation scores, two lexical translation scores, word count and phrase count penalty, distortion score, and language model score. We use the Moses decoder (Koehn et al., 2007) with a reordering limit of 4 for both languages, which generates N-best lists of up to 2000 hypotheses per sentence in a first pass. The second pass uses a part-of-speech (POS) based trigram model, trained on POS sequences generated by a MaxEnt tagger (Ratnaparkhi, 1996). The language models are trained on the English side using SRILM (Stolcke, 2002) and modified Kneser-Ney discounting for the first-pass models and Witten-Bell discounting for the POS models. The baseline system yields state-of-the-art performance.
Table 2: GBL results (%BLEU/PER) on IE task for different weightings of labeled-labeled vs. labeled-unlabeled graph edges (BLEU-based similarity measure).

<table>
<thead>
<tr>
<th>Weighting</th>
<th>dev-2</th>
<th>eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>none (baseline)</td>
<td>22.3/53.3</td>
<td>29.6/45.5</td>
</tr>
<tr>
<td>(a)</td>
<td>23.4/51.5</td>
<td>30.7/44.1</td>
</tr>
<tr>
<td>(b)</td>
<td>23.5/51.6</td>
<td>30.6/44.3</td>
</tr>
<tr>
<td>(c)</td>
<td>23.2/51.8</td>
<td>30.0/44.6</td>
</tr>
</tbody>
</table>

Table 3: GBL results (%BLEU/PER) on IE tasks with string-kernel based similarity measure.

<table>
<thead>
<tr>
<th>System</th>
<th>dev-2</th>
<th>eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>22.3/53.3</td>
<td>29.6/45.5</td>
</tr>
<tr>
<td>GBL</td>
<td>24.3/51.0</td>
<td>32.2/42.7</td>
</tr>
</tbody>
</table>

Table 4: Effect of GBL on IE system trained with matched data (eval set).

<table>
<thead>
<tr>
<th>System</th>
<th>BLEU (%)</th>
<th>PER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>37.9</td>
<td>38.4</td>
</tr>
<tr>
<td>GBL</td>
<td>39.2</td>
<td>37.2</td>
</tr>
</tbody>
</table>

Table 5: AE system results (%BLEU) on IE tasks.

For the AE task we used $\theta = 0.5$; however, this threshold was not tuned extensively. Results using BLEU similarity are shown in Table 5. The best result on the eval set yields an improvement of 1.2 BLEU points though only 0.2% reduction in PER. Overall, results seem to vary with parameter settings and nature of the test set (e.g. on dev5, used as a test set, not for optimization, a surprisingly larger improvement in BLEU of 2.7 points is obtained!).

Overall, sentence similarities were observed to be lower for this task. One reason may be that the AE system includes statistical tokenization of the source side, which is itself error-prone in that it can split the same word in different ways depending on the con-
Table 5: AE results (%BLEU/PER, $\theta = 0.5$)

<table>
<thead>
<tr>
<th>Method</th>
<th>dev4</th>
<th>dev5</th>
<th>eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>30.2/43.5</td>
<td>21.9/48.4</td>
<td>37.8/41.8</td>
</tr>
<tr>
<td>GBL</td>
<td>30.3/42.5</td>
<td>24.6/48.1</td>
<td>39.0/41.6</td>
</tr>
</tbody>
</table>

text. Since our similarity measure is word-based, this may cause similar sentences to fall below the threshold. The string kernel does not yield any improvement over the BLEU-based similarity measure on this task. One possible improvement would be to use an extended string kernel that can take morphological similarity into account.

**Example** Below we give an actual example of a translation improvement, showing the source sentence, the 1-best hypotheses of the baseline system and GBL system, respectively, the references, and the translations of similar sentences in the graph neighborhood of the current sentence.

**Source**: A+ mE*rp A ymknk {ltqAT Swrp lnA

**Baseline**: i’m sorry could picture for us

**GBL**: excuse me could you take a picture of us

**Refs**:  
excuse me can you take a picture of us  
pardon would you mind taking a photo of us  
pardon me could you take our picture  
pardon me would you take a picture of us  
excuse me could you take a picture of u

**Similar sentences**:  
could you get two tickets for us  
please take a picture for me  
could you please take a picture of us

6 Related Work

GBL is an instance of semi-supervised learning, specifically transductive learning. A different form of semi-supervised learning (self-training) has been applied to MT by (Ueffing et al., 2007). Ours is the first study to explore a graph-based learning approach. In the machine learning community, work on applying GBL to structured outputs is beginning to emerge. Transductive graph-based regularization has been applied to large-margin learning on structured data (Altun et al., 2005). However, scalability quickly becomes a problem with these approaches; we solve that issue by working on transitive closures as opposed to entire graphs. String kernel representations have been used in MT (Szedmak, 2007) in a kernel regression based framework, which, however, was an entirely supervised framework. Finally, our approach can be likened to a probabilistic implementation of translation memories (Maruyana and Watanabe, 1992; Veale and Way, 1997). Translation memories are (usually commercial) databases of segment translations extracted from a large database of translation examples. They are typically used by human translators to retrieve translation candidates for subsequences of a new input text. Matches can be exact or fuzzy; the latter is similar to the identification of graph neighborhoods in our approach. However, our GBL scheme propagates similarity scores not just from known to unknown sentences but also indirectly, via connections through other unknown sentences. The combination of a translation memory and statistical translation was reported in (Marcu, 2001); however, this is a combination of word-based and phrase-based translation predating the current phrase-based approach to SMT.

7 Conclusion

We have presented a graph-based learning scheme to implement a consistency model for SMT that encourages similar inputs to receive similar outputs. Evaluation on two small-scale translation tasks showed significant improvements of up to 2.6 points in BLEU and 2.8% PER. Future work will include testing different graph construction schemes, in particular better parameter optimization approaches and better string similarity measures. More gains can be expected when using better domain knowledge in constructing the string kernels. This may include e.g. similarity measures that accommodate POS tags or morphological features, or comparisons of the syntax trees of parsed sentence. The latter could be quite easily incorporated into a string kernel or the related tree kernel similarity measure. Additionally, we will investigate the effectiveness of this approach on larger translation tasks.

**Acknowledgments** This work was funded by NSF grant IIS-032676 and DARPA under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of these agencies.
References

A. Alexandrescu and K. Kirchhoff. 2007. Data-Driven Graph Construction for Semi-Supervised Graph-Based Learning in NLP. In *HLT*.


A. Goldberg and J. Zhu. 2006. Seeing stars when there aren’t many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL Workshop on Graph-based Algorithms for Natural Language Processing*.


