Deep Dynamic Relational Classifiers: Exploiting Dynamic Neighborhoods in Complex Networks

Hogun Park
Dept of Computer Science
Purdue University
West Lafayette, IN
hogun@purdue.edu

John Moore
Dept of Computer Science
Purdue University
West Lafayette, IN
moore269@purdue.edu

Jennifer Neville
Dept of Computer Science
Purdue University
West Lafayette, IN
neville@purdue.edu

ABSTRACT

Using information about a node's link *neighborhood* in classification models of node behavior typically improves predictive performance in complex network domains. For example, in social networks, friends can provide additional information about a user's preferences, attitudes, and behaviors. Recent work in relational learning and social network mining has focused on developing models to exploit this type of information. Despite the fact that link information is often changing over time (e.g., emails, colocation events, transactions), there are few methods that model these dynamics with the aim of improving node classification. The majority of current methods either summarizes the temporal information into link weights or aggregates over time to produce a static network. In this work, we consider whether modeling the dynamics in a node's neighborhood can improve predictive performance. Our key insight is to use a convolutional neural network with max pooling to aggregate over the dynamics. Our model, the Deep Dynamic Relational Classifier (DDRC), represents a node's neighbors over time as a sequence, but the use of max pooling focuses the model on the existence of key events rather than the full temporal ordering. This is in contrast to previous approaches that focus on the full set of events over time. Our experimental results show that DDRC outperforms several competing baselines, including previous relational and temporal classifiers.

Keywords

Relational Learning; Dynamic Graphs; Node Classification; Convolutional Neural Networks

1. INTRODUCTION

In complex network domains, a node's link *neighborhood* can provide information about dependencies among nodes, and when it is used in classification models it typically improves predictive performance. For example, in social networks, friends can provide additional information about a user's preferences, attitudes, and behaviors. Recent work in relational learning and social network mining has focused on developing models to exploit this type of information such as direct neighbors [25], first and second-order proximity among vertices [23, 25], and random walks [16, 4]. In these models, they make predictions based on their neighborhood graph [6] or the new graph embedding space.

However, neighborhood link information is often changing over time. For example, students get information about registration and coursework in the beginning of the semester by email. Then, they regularly submit homeworks to their instructors and discuss their class-project with their teammates from the middle to the end of the semester. In this type of scenario, links are generated based on emails among users and keep evolving over time. While evolution and temporal dynamics in link structure are often exploited in link prediction [11, 10] and recommender system [26], link dynamics are relatively under-explored in node classification methods.

For improving node classification, some recent methods have utilized temporal dynamics in relational models [22, 18]. These methods summarize the temporal information into link weights or aggregate over time for use in a weighted relational classification scheme. However, the weighting schemes typically prefer events in the recent and decay the weight as time goes. Moreover, because the methods assign the same weights for each temporal graph, node-level dynamics is difficult to capture.

Recently, Convolutional Neural Network (CNN) [9] and Recurrent Neural Network (RNN) [12] have been used successfully for sequential data in natural language processing and speech recognition, but it is still challenging to apply them to network and graph data. Specifically, these neural network architectures require a series of fixed-length vectors or matrices, so networks, which are not grid-structured, are challenging to map as input to CNN or RNN. Recent work on deep learning for graph classification primarily focuses on graph-level prediction [15] or node-level collective classification [13]. However, these methods do not make use of temporal dynamics during learning.

In this paper, we propose a convolutional neural network architecture with max pooling for node classification, which models the dynamics among a node's neighbors. We refer to the model as a *Deep Dynamic Relational Classifier* (DDRC). The input to DDRC takes sequences of neighbors for a node, and the model architecture uses max pooling to learn from key events (e.g, the existence of particular neighbors at some point in time). This is in contrast to previous approaches that focus on weighted aggregation over time or modeling

MAISoN- WSDM Workshop '17 Cambridge, UK

DOI: 10.475/123_4

the temporal dynamics explicitly.

Our experimental results on two real-world datasets show that DDRC outperforms other alternatives such as previous relational classifiers and graph-based RNNs. We also demonstrate that DDRC consistently performs well regardless of the variability of a node's neighborhood.

2. BACKGROUND AND RELATED WORK

2.1 Neural Networks

Multilayer Perceptrons.

Multilayer Perceptrons (MLPs) or vanilla neural networks are simply compositions of non-linear and linear functions [19]. Activation functions, which are differentiable, take as input a linearly transformed input. Typically, the linear transformation will be represented by weight matrices, W^i and bias terms, b_i where i denotes the layer index. One can take the output of an activation function and further apply this process to obtain multiple layers. Softmax is usually applied to the final output to obtain probabilities for multi-class classification, where $softmax(\mathbf{z})_j = \frac{e^{z_j}}{\sum e^{z_i}}$ for the jth entry in vector $\mathbf{z}$. Backpropagation with an objective function such as cross entropy is typically used for learning. We sometimes call MLPs, Deep Neural Networks if the MLP is composed of many layers.

Long-Short Term Memory.

The Long Short Term Memory (LSTM) model attempts to model sequential inputs [5]. LSTM takes input at each timestep, t, and updates its hidden state. The LSTM has a way of "remembering" important information while disregarding the superfluous information through it's gating mechanisms. LSTMs have the advantage that it can model variable-sized inputs, while traditional MLPs and other forms of neural networks are only designed for fixed-length inputs.

Softmax, can be used on the last hidden state or intermediate hidden states to induce probability distributions for prediction. LSTMs are designed to be end-to-end differentiable since activation functions are chosen to be differentiable. Backpropagation through time (BPTT) proceeds by backpropagating normally through all timesteps, then averaging the weight updates derived from each timestep.

Convolutional Neural Network.

Convolutional Neural Network (CNN) [9] is a type of feed-forward neural network with convolution layers, which is originally developed for image classification. It is composed of multiple layers with sub-sampling and dropout and comes with fully connected layers in the end. The layers are to model the multi-dimensional structure of input data because the convolution function of each layer shares its feature map with the corresponding dimension. The local connection is followed by sub-samping and dropout before its signal activation. The convolution layer's parameters are composed of a set of feature maps, which have a fixed receptive field, and regularization constants like dropout. The feature map is generally 1 or 2-dimensional matrix. During the forward propagation, each feature map is convolved across the width and height of the input data, computing the dot product between entries of the filter and the input. It produces 1 or 2-dimensional activation matrices by each feature map. This activation matrix could be sub-sampled using 1 or 2-dimensional sampling through taking max or mean values along with the dimension, which is called max pooling or mean pooling. As a result, the network learns feature maps that can capture specific patterns of features through all input regions.

2.2 Graph-based Neural networks

The work most relevant to ours is known as Deep Collective Inference (DCI) [14]. DCI is a semi-supervised approach which operates on a static version of the network. Each node's and neighbor's attributes are encoded as sequential input to an LSTM, and current predictions are propagated as additional input in the next collective iteration. DCI, however, cannot be currently applied to networks with temporal data.

Node embedding research involves learning embeddings for nodes via unsupervised learning. Node based models can then be learned using these embeddings. DeepWalk [16] executes random walks from a given node and then utilizes a skip-gram architecture. Node2Vec [4] extends the skip-gram approach of DeepWalk by performing various sampling strategies to sample neighborhoods rather than performing random walks. Embeddings are useful because they encapsulate local and global structure. However, they are not directly applicable to classification in dynamic networks since they do not consider node attributes, do not directly optimize for class labels, and most importantly do not consider the dynamic nature of changing graphs through time.

The Graph Neural Network (GNN; [21]) is a specialized recurrent neural network for graph data. The model takes as input all node attributes and optionally uses attributes for edges and edge types. The model uses the node attributes as well as the whole network structure as input and propagates information about each node in each hidden state. GNNs are not node-based models and must use the whole graph as input, which limits its feasibility on large graphs, and on problems that involve a partially-labeled graph where there is only one network example. Furthermore, the vanilla GNN cannot model the dynamic nature of a changing network through time.

A graph-based convolutional neural network has been proposed, which aims to perform graph classification [15]. They employ an ordering algorithm for graph normalization so that 2-dimensional grid structure can be generated. However, this work cannot be applied to the node classification problem directly. Also, to the best of our knowledge, work involving CNNs have not been applied to the node classification problem in dynamic networks.

2.3 Collective Inference methods

Relational machine learning (RML) methods jointly model node labels given node attributes and relational structure under the assumption of a static non-changing network over time (e.g. Only the network at a given timestep is considered) [3]. Semi-supervised learning (SSL) RML methods seek to use predictions of unknown node labels in their learning procedures (e.g. [17]). They do so by utilizing the whole network to simultaneously estimate model parameters while predicting labels of unknown nodes. Pseudolikelihood Expectation Maximization ($PLEM$) is currently a state-of-the-art SSL RML model. It uses expectation maximization (EM) to estimate parameters and performs predictions an in iterative fashion. The one developed by [17] utilizes a Maximum Entropy constraint in the inference step to produce

highly calibrated probability estimates. This is the one we compare to in Section 5.

2.4 Dynamic time based models

There is some work on dynamic models of graphs changing through time. The Time Varying Relational Classifier (TVRC) attempts to model dynamic structure through a two-step process [22]. The graph summarization phase attempts to exploit the temporal relationship influence by summarizing them well. After this, the relational classification phase then utilizes the summarization to build a classification model. The key idea behind TVRC is to model the dynamic structure through an exponential weight decay kernel, where the implicit assumption is that network structure in recent past is more important than the structure in the earlier past. Since an LSTM can learn more complex representations than this simple kernel, LSTM is expected to perform better or the same.

For link prediction, several generative models [11] [10] are proposed in dynamic networks. [11] proposes a Neighbor Influence Clustering algorithm using Restrictive Boltzmann Machine, and [10] uses conditional Restrictive Boltzmann Machine to learn latent features between senders and receivers. However, they are limited to link prediction, and the feasibility for node classification has not been addressed yet.

3. PROBLEM DEFINITION

We define a graph sequence as a set of graphs such that $\mathbf{G} = G_1, G_2, ..., G_m$. Each G_k has the same set of nodes, $\mathbf{v_i} \in \mathbf{V}\ \forall i \in [1, n]$, but a different set of edges, $\mathbf{E_k} \subseteq \mathbf{V} \times \mathbf{V}$ such that $G_i = \langle \mathbf{V}, \mathbf{E_k} \rangle$. If $e_{ij} \in \mathbf{E_k}$, there is an edge between $\mathbf{v_i}$ and $\mathbf{v_j}$ at time k, otherwise there is not. Alternatively, let $\mathbf{A} = A_1, A_2, ..., A_m$ be the set of adjacency matrices for $\mathbf{G}$, where $A_{k_{ij}} = 1$ if $e_{ij} \in \mathbf{E_k}$, 0 otherwise. The corresponding adjacency row indexed by node $\mathbf{v_i}$ at time k is $\mathbf{A_{k_i}} \in R^{1 \times n}$, where the j^{th} entry in $\mathbf{A_{k_i}}$ indicates the existence of a neighbor for $\mathbf{v_i}$ such that $e_{ij} \in \mathbf{E_k}$.

While the network structure is changing over time, we assume that the node attributes are not changing over time (or changing at a much slower rate than links). Let $\mathbf{F}$ be the feature set over the nodes. Each $\mathbf{v_i} \in \mathbf{V}$ has a corresponding feature vector $\mathbf{f}_i \in \mathbf{F}$ describing each node. $\mathbf{Y}$ is the label set over the nodes, where only a subset of the nodes, $\mathbf{v_i} \subseteq \mathbf{V}$, have a class label, $y_i \in \mathbf{Y}$. The goal is to learn a model from the partially labeled network and use the model to make predictions $\hat{\mathbf{y}}$ for the unlabeled nodes $\{v_i\}$ $s.t.$ $y_i \notin \mathbf{Y}$. In this work, we assume that $\mathbf{Y}$ can be multi-labeled and takes $\{0, 1, ...\}$. Moreover, each prediction $\hat{y}_i$ has estimated probability for $\mathbf{v_i}$. $\mathbf{V_L}, \mathbf{V_U}$ refers to the nodes that are labeled and unlabeled, respectively.

4. MODEL DESCRIPTION

In this paper, we propose a *Deep Dynamic Relational Classifier* (DDRC) for node classification in dynamic networks. DDRC is a convolutional neural network, which takes modified network inputs that reflect dynamic behavior and node attributes. DDRC models nodes dynamic patterns using non-linear combinations of multiple shared-feature maps.

4.1 DDRC Input Representation

DDRC uses the adjacency matrices , $A_1, A_2, ..., A_m$, describing dynamic behavior for $\mathbf{G}$ with the feature set, $\mathbf{F}$. For a given node $v_i \in \mathbf{V}$, F_i and A_{k_i} refer to the feature vector and adjacency row for v_i at time k, respectively.

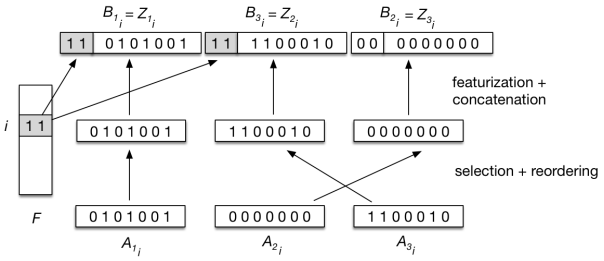

Figure 1: Input Representation for DDRC.

Let $B_{k_i} = (F_i, A_{k_i})$, where we simply concatenate the feature vector and adjacency row together to form a new vector. We further concatenate B_{k_i}'s to form B_i, ie

$$B_i = \langle B_{1_i}; B_{2_i}; ...; B_{m_i} \rangle \tag{1}$$

To form our final input, we simply push any B_{k_i} to the end where the corresponding A_{k_i} is composed of all 0s. Any B_{k_i} that fits this criteria is also set to 0. Order among B_{k_i} that does not fit this criteria is preserved. The all-zero adjacency matrices often have meaningful information for modeling temporal dynamics, but it makes input so sparse for DDRC and shows worse results in experiment. As a result, its new transformed input is shown as the below:

$$Z_i = \langle Z_{1_i}; Z_{2_i}; ...; Z_{m_i} \rangle \tag{2}$$

This input specification aids in representing the temporal behavior of v_i's neighborhoods through time. Figure 1 describes the input representation given an example sequence of adjacency matrices, A_1, A_2, and A_3, and shows the two-step process in forming Z_i's. Note that it is crucial to perform the second reordering step as irrelevant 0 entries may be ignored.

4.2 DDRC Model

Figure 2 represents the convolutional layers, feature maps (2-b), and max-pooling (2-c) of DDRC. In DDRC, we use a 1-dimensional convolution function over a sequence, Z_{1_i}, Z_{2_i}, ..., Z_{m_i} of node v_i. The 1-dimensional convolution is an operation between a feature map, $\mathbf{M} \in R^{s \times (|V|+|F|)}$, where s is the size of the convolutional feature map, and $\mathbf{Z}_{i,j:j+s}$, where j is the current index of convolution function within the whole input sequence. In other words, the 1-dimensional convolution function takes the dot product of matrix $\mathbf{M}$ with the sub-sequence of $\mathbf{Z}_{i,j:j+s}$ to form $\mathbf{C}$ as in the following equation:

$$\mathbf{C}_{i,j} = g(\mathbf{M}^T \mathbf{Z}_{i,j:j+s} + b) \tag{3}$$

where g is the non-linear differentiable function (e.g. the hyperbolic tangent or rectified linear unit)

In this context, max pooling (Figure 2-(c)), provides a way to capture salient patterns by taking the max values from the previous hidden layer. Thus, the new layer $p \in R^{1 \times \rho}$ of length $\rho = \lceil m/k \rceil$, where k is the size of max pooling, is generated, flattened and connected to a softmax layer to predict the class label. (Figure 2-(d))

If k is the same or larger than m, then we have one node per a feature map after max pooling, which is same as *max-*

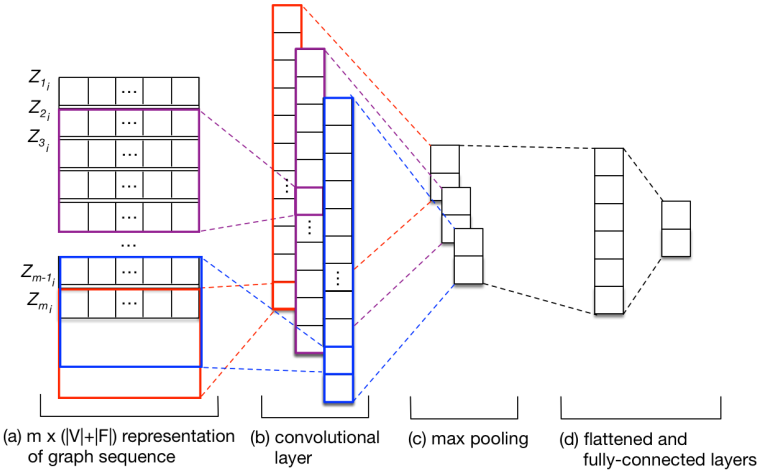

Figure 2: DDRC Convolutional Neural Network Architecture.

Table 1: Data Summary

	$\|V\|$	$\|E\|$	$\|T\|$	Unit of T	$\|F\|$	$\|C\|$
Facebook	1,741	80,715	52	7 days	2	2
IMDB	1,375	30,623	20	1 year	4	2

over-time pooling [1] [8] While *max-over-time pooling* is reported that it shows good performance at document classification, we now suggest a more generalized version for graph sequences which could have complex repeating patterns over time.

For learning, we use categorical cross-entropy as a loss function at the final layer

$$L_i = -\sum_j y_{i,j} log(\hat{y}_{i,j}) \quad (4)$$

In equation 4, j is the number of class labels. Since all activation functions are differentiable, learning is simply done via back-propagation.

5. EVALUATIONS

5.1 Data

Facebook.

The Facebook network was scraped from a Purdue University group. Each user (node) is associated with political views, religious views, and gender. They all can have only binary values. Political views are used as class labels, while religious views and gender are used as features. The number of users is 1,741. An edge is formed when a user writes a post to his or her friend's wall. We did not count self-posts, and users who posted more than 1 times a week during at least 6 weeks are chosen. The positive proportion of their class labels is 0.639. The data is a variation from [17].

IMDB.

The Internet Movie Database (IMDB: www.imdb.com) releases information about movies such as directors, actors, studios, gross, budget, etc. For evaluation, we use Kaggle's IMDB 5000 movie dataset from IMDB. For this work, we choose budget, content rating, the number of faces in a movie poster, and genres as features. The budgets are quantized from 0 to 9 using the percentile of the amount of numbers. For example, if the budget of a movie is in the 90-100 percentiles, it assigns 9. Each feature is transformed into binaries. An edge is formed when two movies share an actor or actress. A movie has a positive label if the movie gross is larger than 10 million dollars. The positive proportion of classes in the dataset is 0.776. All movies in this dataset are connected with more than 1 movie, and their minimum number of time windows is 5.

5.2 Comparison Models

Other ways of performing node classification in dynamic networks are utilized to see how temporal information improves performance. State-of-the-art neural network architectures for sequential data are also included for comparison.

Logistic Regression (LR).

Logistic regression model is performed using only node features. This allows us to compare how relational and dynamic structure improve performance.

Pseudo-likelihood Expectation Maximization (PL-EM).

PL-EM with maximum entropy constraints is a semi-supervised approach which considers node attributes and network structure, but its temporal structure is not utilzed. Therefore, we modify our network such that if a node has an edge at any timestep, then the edge appears in a static version of the graph. Although DDRC does not perform collective inference over the whole network, PL-EM is chosen to see how significant temporal information aids in prediction.

Multi-layered Neural Network (NN).

4-layered Neural network is also used for comparison. In order to learn the network, the static version of the graph is used to train and test. In other words, all Z_ms are aggregated in a vector, whose size is |V| + |A|, and normalized to make the sum of elements to 1. If a user communicates with some users more frequently, then they have highly weighted scores. We call the neural network as *NN-W*. On the other hand, its unweighted version, *NN-U*, assigns binary values using the occurrence of the corresponding neighbors. The number of hidden nodes at each layers before the last softmax layer are [1024, 512, 64].

LSTM.
LSTM is one of alternatives to classify sequential data such as natural language [2] and speech data [20]. We use LSTM for comparison since they can handle variable-length input, and our input Z_i's described in section 4 may be regarded as variable length if any empty or zeroed out Z_{j_i} exists and is removed. In this LSTM, we use the final activation of hidden nodes to classify. The number of hidden nodes is chosen from grid search. The grid search is from [8, 16, 32, 64, 128].

LSTM with Pooling.
While the previous LSTM represents the entire document by one vector (document embedding), this *LSTM with Region Embedding + Pooling* [7] is designed to detect regions of text (of arbitrary sizes) that are relevant to the task and representing them by vectors (region embedding). For this, every hidden layer in LSTM returns output nodes and uses them with a pooling layer before the softmax layer. [7] said that it is effective when they use one-hot representation for document classification. The grid search is also from [8, 16, 32, 64, 128] for the number of hidden nodes. Before maxpooling, a layer of time-distributed neural network is used to improve performance. The size of hidden nodes is searched from [8, 16, 32], and the size of max pooling is chosen from [1, 2, 4].

5.3 Evaluation Methodology

We train the models on training/validation sets and report results on the test set. Every result on experiment is from the average of results on 20 randomly shuffled node sets. 70, 20, and 10 percent of the data are used for training, testing, and validation, respectively.

For all neural network models, max epochs are set to 30, and if accuracy on validation dataset does not increase during 15 epochs, it stops training. We also use dropout regularization (0.2) and rectified linear units for activation functions. For optimization, *rmsprop* [24] (learning rate = 9,991, rho = 0.9, fuzz factor = 1e-8, and decay = 0) is used to do optimization using same parameters.

For DDRC, the number of feature maps is searched over [1, 4, 16], and the size of feature maps is chosen from [1, 2, 4]. Max pooling size is selected from [1, 2, 4, |T|].

6. RESULTS

Figure 3 and Figure 4 show results of experiment, and DDRC shows better accuracy on two datasets. For the Facebook dataset, DDRC shows the best results. LR uses only features of nodes to predict labels and is a simple baseline for comparison. It performs worse as expected compared to all other alternatives. NN-W and NN-U use weighted and unweighted neighbor distribution with features, respectively, and their accuracy is between other temporal models and logistic regression. In this dataset, NN-U shows better performance than NN-W. It means that representing the strength of edges using the number of occurrences is not helpful to increase performance. PL-EM has the advantage of using predictions on test nodes in an iterative fashion in the learning process so performs extremely well despite its lack of temporal information. PL-EM performs second-best

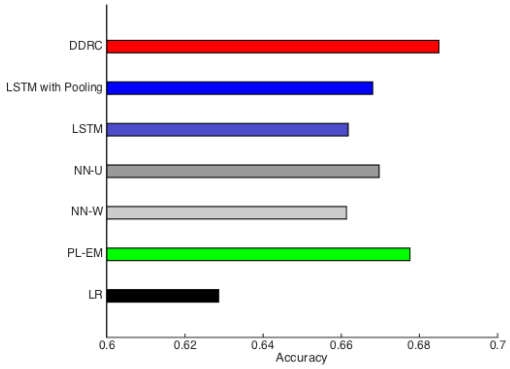

Figure 3: Accuracy for Facebook

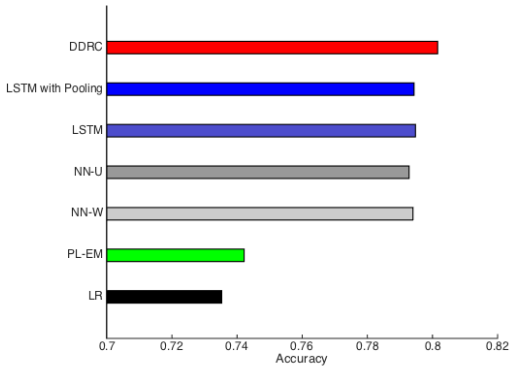

Figure 4: Accuracy for IMDB

to DDRC on Facebook.

DDRC also shows better results in IMDB dataset in Figure 4. Since many neural network-driven models show similar and high accuracy as opposed to LR and PLEM, it seems that direct neighbors and temporal dynamics are significant factors in prediction; however, DRCC is still the best performing model on this dataset.

To verify that ordering the temporal input properly helps, we compare DDRC with a reversed version (DDRC-R) and a randomized sequences version (DDRC-RS) (e.g. Z_i's are randomly shuffled). The results are reported in Figure 5 and Figure 6. DDRC-RS is worse DDRC and DDRC-R. DDRC and DDRC-R have essentially the same performance, which makes sense because a reversed order should not have a significant affect on learning. In the figures, DDRC-Z shows results when input representation includes empty (all zeros) interaction at the corresponding time windows. In other word, A_i is used for Z_i before the zero-padding in DDRC-Z. It is better than random shuffling but makes CNN worse.

LSTM and LSTM with Pooling do not perform as well as DDRC in both datasets. It would seem that having the whole sequential information at once is more important than modeling changing hidden states through time.

To see how variability in the neighborhood of nodes affects performance, we investigate how accuracy varies with neighborhood variability. More specifically, we calculate how often the neighbors of a node appear over time to measure the variability of its neighborhood. For example, assume that a node v_i's neighbors, A_{1_i}, A_{2_i}, and A_{3_i} at time 1, 2, and 3 are [0, 1, 0, 1], [0, 0, 0, 1], and [1, 1, 0, 0]]. The proportion of their occurrences is calculated as [1/3, 2/3, 0, 2/3] for each neighbor node. Intuitively, if the average over the

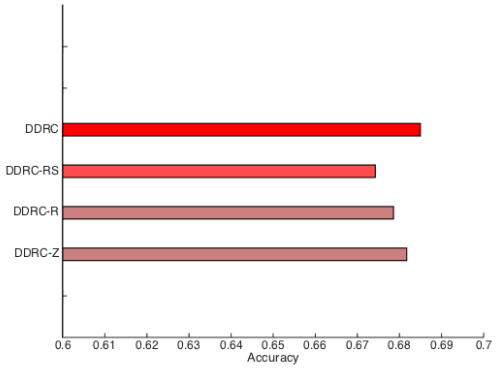

Figure 5: DDRC's Accuracy on 4 Different Input Representations for Facebook

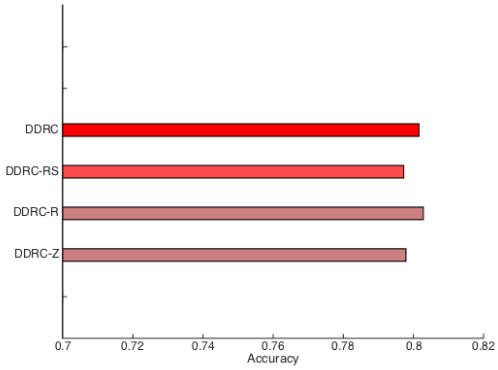

Figure 6: DDRC's Accuracy on 4 Different Input Representations for IMDB

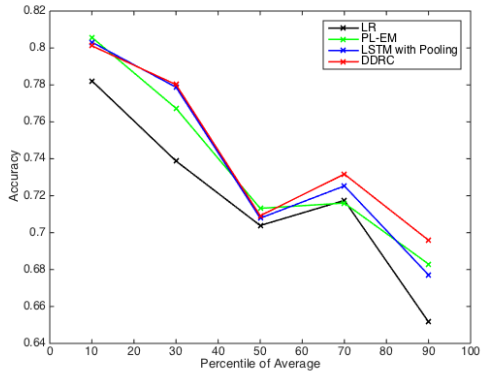

Figure 7: Accuracy on Classification for IMDB (X-axis: Average of Neighbor's Proportions)

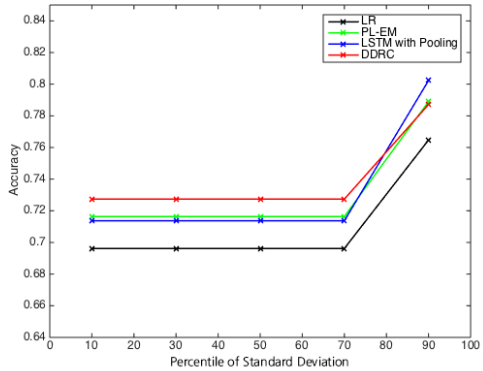

Figure 8: Accuracy on Classification for IMDB (X-axis: : Standard Deviation of Neighbor's Proportions)

proportion vector is small, the corresponding node has more variability. Similarly, if the standard deviation is larger, it represents more variability.

Figure 7 and 8 show the results if we partition nodes based on these two criteria. In this experiment, we use percentiles from all averages or standard deviations to measure its effect. Percentiles in figures are depicted at 5 points: 0-20, 20-40, 40-60, 60-80, and 80-100. DDRC shows good results over all percentiles. Although LSTM with Pooling shows comparable results with DDRC for nodes with high variability, it shows worse performance for nodes with less variability. This is probably because the LSTM is good at modeling the relationship of neighbors in each time window, but doesn't have the advantage of looking at all inputs at once as in DDRC.

7. CONCLUSIONS

We have described a convolutional neural network architecture, DDRC, which exploits dynamic neighborhoods over time. Different from previous relational models, our DDRC model is able to capture node-level temporal dynamics using max pooling, which allows to model the existence of key events. Our evaluation on real-world datasets demonstrates that DDRC shows better accuracy on node classification, and the performance is also stable with respect to neighborhood nodes' variability through time.

8. REFERENCES

[1] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, Nov. 2011.

[2] A. M. Dai and Q. V. Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015.

[3] L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[4] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 2016.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[6] T. Jebara, J. Wang, and S.-F. Chang. Graph construction and b-matching for semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 441–448. ACM, 2009.

[7] R. Johnson and T. Zhang. Supervised and semi-supervised text categorization using one-hot lstm for region embeddings. In *Proceedings of The 33rd International Conference on Machine Learning*, ICML '16, pages 526–534, 2016.

[8] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1746–1751, 2014.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner.

Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] K. Li, J. Gao, S. Guo, N. Du, X. Li, and A. Zhang. Lrbm: A restricted boltzmann machine based approach for representation learning on linked data. In *Proceedings of International Conference on Data Mining*, ICDM '14, pages 300–309. IEEE, 2014.

[11] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang. A deep learning approach to link prediction in dynamic networks. In *Proceedings of SIAM International Conference on Data Mining*, SDM '14, pages 289–297. SIAM, 2014.

[12] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. *Interspeech*, 2:3, 2010.

[13] D. D. Monner and J. A. Reggia. Recurrent neural collective classification. *IEEE transactions on neural networks and learning systems*, 24(12):1932–1943, 2013.

[14] J. Moore and J. Neville. Deep collective inference. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017.

[15] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of The 33rd International Conference on Machine Learning*, ICML '16, 2016.

[16] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14, pages 701–710. ACM, 2014.

[17] J. J. Pfeiffer, III, J. Neville, and P. N. Bennett. Overcoming relational learning biases to accurately predict preferences in large scale networks. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 853–863. ACM, 2015.

[18] R. Rossi and J. Neville. Time-evolving relational classification and ensemble methods. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, PAKDD '12, pages 1–13. Springer, 2012.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 318–362. MIT Press, 1986.

[20] H. Sak, A. W. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, pages 338–342, 2014.

[21] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *Trans. Neur. Netw.*, 20(1):61–80, Jan. 2009.

[22] U. Sharan and J. Neville. Temporal-relational classifiers for prediction in evolving domains. In *Proceedings of Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 540–549. IEEE, 2008.

[23] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077. ACM, 2015.

[24] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[25] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1225–1234. ACM, 2016.

[26] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 723–732. ACM, 2010.