Glocal: Incorporating Global Information in Local Convolution for Keyphrase Extraction

Animesh Prasad[†‡] **Min-Yen Kan**[†‡]
[†] School of Computing
National University of Singapore
[‡] Institute for Application of Learning Science and Educational Technology
National University of Singapore
animesh@u.nus.edu kanmy@comp.nus.edu.sg

Abstract

Graph Convolutional Networks (GCNs) are a class of spectral clustering techniques that leverage localized convolution filters to perform supervised classification directly on graphical structures. While such methods model nodes' local pairwise importance, they lack the capability to model global importance relative to other nodes of the graph. This causes such models to miss critical information in tasks where global ranking is a key component for the task, such as in keyphrase extraction. We address this shortcoming by allowing the proper incorporation of global information into the GCN family of models through the use of scaled node weights. In the context of keyphrase extraction, incorporating global random walk scores obtained from TextRank boosts performance significantly. With our proposed method, we achieve state-of-the-art results, bettering a strong baseline by an absolute 2% increase in F_1 score.

1 Introduction

Learning directly on a graphical structure is a crucial requirement in many domains. These graphs represent information in many forms, ranging from interconnected user groups to contextually linked documents to a central document by shared vocabulary. Learning on graphs has been studied extensively in the form of spectral clustering (Ng et al., 2002). The potential of learning directly on graphs has realized in semi-supervised settings where labels for only a few of the nodes are available. Some prior work formulates such setup as propagating the label information using some form of graph-based regularization (Kipf and Welling, 2016). Recently proposed works have updated such methods to be end-to-end learnable in the deep learning style by employing gradient descent on nodes within a fixed neighborhood, approximating spectral clustering's means of approximating the graph's eigenvectors (Bronstein et al., 2017) by aggregating neighborhood features. Recent advancements in normalizing the gradient range further improve the efficiency of such solutions (Kipf and Welling, 2016). However, these techniques can only exploit local features within the neighborhood of individual nodes. For some tasks, such simplified local feature aggregation may be sufficient, but insufficient for tasks that need global relative importance information.

One such important graph-based task is keyphrase extraction. In this task, individual words or phrases serve as graph nodes, and edges represent some form of co-occurrence. Keyphrase extraction has been extensively studied, in both supervised (classification) and unsupervised (ranking) modes. Depending on the length of the text and the final application of the task, solutions can be sample-based classification, pairwise ranking or sequential labeling. For example, Kim et al.(2010) explore the case of extracting top keyphrases from complete documents for downstream indexing, while Augenstein et al.(2017) connects its usage for knowledge base generation, aiming to extract all plausible keyphrases within a short excerpt. Treating a full-text scenario is arguably more challenging than the treatment of an excerpt scenario, as it requires the understanding of the much larger scale of text and extracting its most salient aspects.

Traditional supervised models employ a host of hand-engineered features — $tf.idf$, candidate length, POS tags, sectional information, frequency, among others (Kim et al., 2013; Hasan and Ng, 2010) — trained with a wide range of classifiers. As they typically model the task as a binary classification task (*i.e.*, $keyphrase, \neg keyphrase$), they suffer severely from class imbalance as keyphrases are the exception among most plausible candidates.

Unsupervised methods use co-occurrence as a signal for the labels. Under the hypothesis that keyphrase saliency is strongly correlated with repetition, graphical methods for unsupervised keyphrase extraction employ centrality measures and random walk techniques to rank prospective keyphrases (Mihalcea and Tarau, 2004). This hypothesis is widely exploited, with proposed extensions further enriching the graph by incorporating topic, section and/or position information (Florescu and Caragea, 2017b,a; Jiang et al., 2018), among other forms of side information.

With these in mind, we make two important observations about the existing keyphrase extraction techniques:

- In the supervised setting, word importance is captured in metrics and engineered features, as is local random walk scores. However, the structure of the graph formed by the text is not exploited.
- In the unsupervised setting, most techniques do not tightly incorporate the rich semantic features common in the supervised setting. Furthermore, random walk scores are used as-is, without the capability of being fine-tuned by downstream supervision.

From this dichotomy, we see there is a gap to close in merging the advantages of both. We propose a Glocal (global–local portmanteau) technique which incorporates both components directly over the word–graph. Specifically, we contribute a neural model that elegantly incorporates the random walk scores, while incorporating parameters to fit keyphrase labels. To the best of our knowledge, our model is the only supervised full-text keyphrase extraction model that operates directly on the word–graph.

2 Related Work

Our work draws motivation from the introduction of random walks to NLP. In this regard, TextRank (Mihalcea and Tarau, 2004), is a central representative work that serves as a basis for many text extraction modeling techniques used in entity extraction and extractive summarization (we use random walk and TextRank interchangeably in this paper). The success of the application of such random walks in text extraction is based on the hypothesis that the important nodes aggregate more mass and are thereby representative of the graph as a whole. Importantly, TextRank can be viewed as a **ranking** model, as it induces a ranking of graph nodes via its centrality calculation. However, as noted, supervised techniques for properly incorporating this information natively within the model, in our opinion, has yet to be explored.

Recently, neural models have been developed to work on graphs. These models port the ideas of spectral clustering into the deep learning modality, allowing direct computation on graphs. Methods such as Graph Convolution Network (or GCN) can then be applied to many different task scenarios which natively feature graphical structures such as citation and community graphs, which are common in the database, information retrieval, and digital library domains (Kipf and Welling, 2016; Hamilton et al., 2017). They enrich the graph by aggregating features with information gathered from the neighborhood of the node to be classified. Enhancements to the model introduce much sophisticated local information aggregation between the node pairs as in Graph Attention Networks (GAT) Veličković et al.. However, we note that such prior methods fall inherently into the **classification** paradigm, and hence focus on only local aggregation; i.e., to pull in the most significant feature from its neighbors.

In the context of keyphrase extraction, Zhang et al. (2017) is a recent work that learns directly on the graph. Their method, MIKE, determines the weight of edges and nodes in a supervised manner, rather than just utilizing co-occurrence statistics. Their work features 5 orthogonal features, one of which is topic distribution. They consider the prominence of the tokens per topic as the surrogate for ranking, utilized for model training, by minimizing the difference in predicted and gold-standard rank between iterations. MIKE can be employed only when topic information is available, but unfortunately, does not generalize to the more common case where only gold-standard keyphrases are available for training.

In Augenstein et al.(2017) benchmarking, state-of-the-art rich semantic embeddings deep learning and handcrafted feature–based statistical sequential labeling models used LSTM (Ammar et al., 2017) and CRF (Prasad and Kan, 2017) models respectively. Meng et al.(2017) uses an encoder–decoder model with a copy mechanism for keyphrase extraction (as a special case of

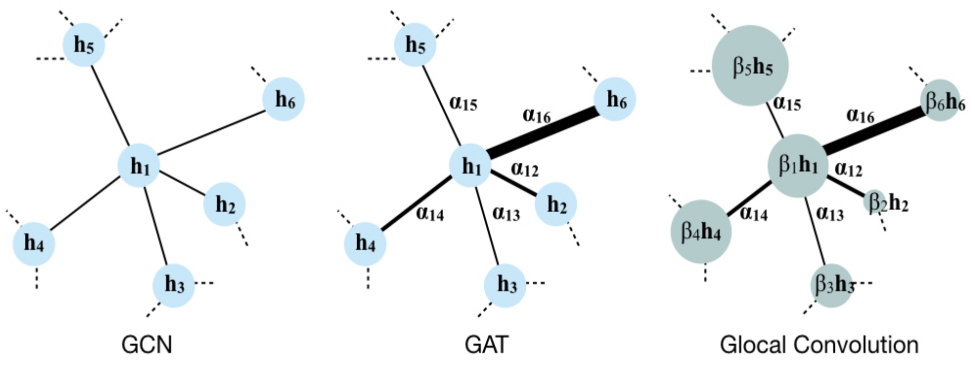

Figure 1: Graph Convolution Model Architectures. Illustrations of the Graph Convolution Network (GCN, left), Graph Attention Network (GAT, center) and our proposed Glocal technique (right), centered on node h_1 within its $1-$hop neighborhood. TextRank on the complete graph is used to compute parameter β_i. GAT parameterizes the edge weights based on gradients (α_i, also represented by the differing edge widths). Our Glocal technique adds the TextRank score (represented by different scaling of nodes), which is not derived from the gradient.

generation). This state-of-the-art technique exploits a complementary idea of sequential semantic modeling focused on generating keyphrases rather than merely extracting them. However, their model does not address the common scenario of keyphrase extraction from long documents but only for short excerpts (namely, the abstract). This assumption reduces the complexity of the problem for sequential models that can effectively encode short text spans but may be ineffective on full-text. We suspect this is a current limitation of the encoder–decoder based models, which necessarily reduces the entire textual sequence into a single vector during the encoding stage – making it susceptible to the vanishing gradient and representation underfitting on large text. Further advances using the encoder–decoder framework such as (Chen et al., 2018) further explore the sequential modeling architectures by improving the attention mechanism with traditional features like title guidance. Note that many forms of such structural information — such as sectional information and citation graph co-occurrence — can enhance basic models, however without loss of generality, in this work we consider only text-based features for all the models.

3 Method

Our proposed model exploits the strength of both supervised and unsupervised modalities by combining two baseline models. Our Glocal model has two components:

- For learning directly over the graph (classification), we use the recently proposed GCN as our baseline model.
- For incorporating global importance (ranking), we use TextRank as our baseline model.

We will first introduce the preliminaries: *e.g.*, Graph Convolution Network (GCN), followed by the modifications that result in the Graph Attention Network (GAT). We then explain how we modify the local convolution operation, to incorporate global importance scores.

3.1 The Graph Convolution Network (GCN)

Graph $G = (V, A)$, where V is a a finite set of vertices such that $|V| = n$ and $A \in R^{n \times n}$ is an undirected weighted adjacency matrix representing all edges, assume $x : V \to R^n$ maps each node to x_i, which is a n-dimensional feature vector. Spectral filtering on a signal x is then represented as (Defferrard et al., 2016):

$$y = g_\theta(L)x = Ug_\theta(\Lambda)U^T x, \tag{1}$$

where $L = I_n - D^{-1/2}AD^{-1/2}$ and where I_n is the identity matrix and $D_{ii} = \sum_j A_{ij}$. Further, parameterization and simplification of the filter by (Hammond et al., 2011) results in:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), \tag{2}$$

where the Chebyshev polynomial $T_k(x)$ is computed recursively. Note that Eqn. 2 is K-localized; *i.e.* it depends only on nodes that are at a maximum of K hops away from the target node.

A linear model can approximate this (Kipf and Welling, 2016) resulting in the simplified form:

$$y \approx \theta'_0 x + \theta'_1 (L - I_N) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x\,, \quad (3)$$

with two free, tunable parameters θ'_0 and θ'_1. Constraining $\theta = \theta'_0 = -\theta'_1$ further simplifies the approximation to a single-parameter form:

$$y \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x\,. \quad (4)$$

Note $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ has eigenvalues in the range $[0, 2]$, so a re-normalization trick was proposed previously $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, with $\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ to keep gradients stable. This formulation (GCN) supports the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)\,. \quad (5)$$

Here, $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph G with added self-connections and $W^{(l)}$ is the layer-specific set of weight parameters. σ denotes an activation function; in the GCN model, ReLU is preferred. $H^{(l)}$ is the matrix of activation in the l^{th} layer; $H^{(0)} = X$.

3.2 The Graph Attention Network (GAT)

Unfortunately, the expressive power represented by $\tilde{A}$ is also its biggest limitation: as such, the model only incorporates features from the neighborhood as weighted by the connecting edge normalized to unit sum (*cf.* Fig. 1, left). To address this, the Graph Attention Network model (Veličković et al., 2018) incorporates learnable scaling for each edge. This introduces a local function $attn : R^{F'} \times R^{F'} \rightarrow R$,

$$a_{ij} = attn(W\vec{h}_i, W\vec{h}_j), \quad (6)$$

that computes a score (attention) per node pair (or *edge*, inclusive of self-loops). Here, the $attn$ operator is a single feed-forward layer employing Leaky ReLU activation. This attention is normalized in GAT as:

$$\alpha_{ij} = \frac{\exp(a_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(a_{ik})}, \quad (7)$$

to smooth the gradients. In contrast to GCN, GAT replaces the $\tilde{A}$ with a learned $\tilde{A}'$ where each entry α_{ij} is a normalized score computed on each node pair by the gradient (Fig. 1, center). The normalized attention coefficients are used to compute a linear combination of the features in a node neighborhood $\mathcal{N}$, yielding the equivalent layer-wise propagation rule:

$$\vec{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W \vec{h}_j^{(l)} \right)\,. \quad (8)$$

In terms of Eqn. 5, the fixed weights in the adjacency are replaced with learned weights or equivalently hard neighborhood is replaced by soft neighborhood. Since all edge weights are parameterized, a number ($= T$) of multiple random initializations learn different representations, resulting in a number of different scalings for multiple linear combinations. A final result is achieved by either concatenating or averaging the multiple representations as formulated in Eqns. 9 and 10, respectively.

$$\vec{h}_i^{(l+1)} = \big\Vert_{t=1}^{T} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^t \mathbf{W}^t \vec{h}_j^{(l)} \right) \quad (9)$$

$$\vec{h}_i^{(l+1)} = \sigma\left(\frac{1}{T} \sum_{t=1}^{T} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^t \mathbf{W}^t \vec{t}_j^{(l)} \right) \quad (10)$$

3.3 Glocal: GAT with Random Walk Scores

In both GCN and GAT, a model with K layers incorporates the feature for a node up to its K hop neighbors. Though GAT improves upon GCN by assigning different importance to nodes via learned weights as compared to the static edge weight in GCN, it is still a local computation. The attention factor, *i.e.* the scaling coefficients α_{ij}, are the function of pairwise feature interaction within the local neighborhood and do not account for node centrality nor the global graph structure. We fix this with our Glocal model.

Consider the random walk based score generated for the graph G such that:

$$\beta_j = TextRank(i). \quad (11)$$

We introduce this parameter β_j to the GAT model. Considering this as the global importance component to the node, we obtain two alternative formulations that encode the node importance as either an additive or a multiplicative form:

$$\vec{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\vec{h}_j^{(l)} + \sum_{j\in\mathcal{N}_i}\beta_j\mathbf{W}'\vec{h}_j^{(l)}\right) \quad (12)$$

$$\vec{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i}\beta_j\alpha_{ij}\mathbf{W}\vec{h}_j^{(l)}\right) \quad (13)$$

From Eqns. 12 and 13, we see two scaling factors α and β, which either reinforce or diminish the other's effect (Fig. 1, right). Unlike α, β is neither calculated locally nor normalized. Not normalizing β is an essential component for having both classification and ranking ability. With normalization, even in a neighborhood with minimal TextRank scores, we will get an aggregation of features with unit sum weights. While without normalization it would perform almost no feature aggregation from such nodes.

In our experiments, we use the formulation of Eqn. 13, as a multiplicative formulation easily enables multi-head attention (achieved by multiplying $\tilde{A}$ with B such that $B_{ii} = \beta_i$). Note that this modeling applies to GCN as well: the second component of the Eqn. 13 actually closely resembles the GCN formulation for our proposed model.

4 Experiments

Dataset	Total	Training	Testing
Schutz (2008)	1,323	1,058*	265*
Krapivin (2009)	2,304	1844*	460*
SemEval (2010)	1,095	708	387

Table 1: Number of documents in our keyphrase extraction datasets. '*' denotes that the dataset does not have an official split; results are based on random splits.

We investigate keyphrase extraction, using the most commonly reported full-text datasets, as shown in Table 1. We divide the training data in 80 : 20 fraction for train and validation splits. Our complete pipeline comprises the following steps:

1. Feature Processing. First we perform TextRank on the complete text of each document, retaining only tokens that are nouns and adjectives, filtering out other words (equivalent to simplex noun phrases). We use the gensim library (Řehůřek and Sojka, 2010) to perform TextRank and compute the scores. This process helps in two ways – first, it gives us the node importance value for each keyphrase, needed by Glocal; second, it helps to manage the graph size and lessen the label skew on the minority positive label by removing extraneous tokens. For documents larger than a max size (1200 tokens) we drop the extra least scored tokens. We find that the tokens that have a TextRank score in bottom 50% possess only 16% of partial or full keyphrases in the validation dataset. Hence dropping them from the processing does not affect the recall much. The nodes of the graph are single tokens and not complete phrases, therefore all the tokens of multi-token keyphrases are marked as $keyphrase$ during learning ion the graph. For the node/keyphrase representations, we map our vocabulary to GloVe embeddings using the $2.2M$ vocabulary sized, 300 dimension vector variant (Pennington et al., 2014). For reference, we observe GloVe covers about 90% of the words overall 3 datasets. We then extract various textual features for the candidate keyphrases, including their position of the first occurrence, $tf.idf$ and n-gram count. These features are appended to the word embedding to obtain a final feature vector representing each node. Rather than discard them, we choose to append the n-gram features to retain rich lexical signals obtained from the tokens.

2. Learning. The second step is to train the model with the formulated graphs. We use a 2-layer network with 128 units with ReLU activations for hidden layers, followed by a simple 2-way softmax classification layer ($keyphrase$, $\neg keyphrase$). We further employ 8 attention heads at all layers. We follow Glorot initialization (Glorot and Bengio, 2010) for all setting initial parameters weights, use a dropout of 0.5, and employ a $L2$ regularization of 0.001. We train with Adam optimizer on cross-entropy loss and initial learning rate of 0.01 for 200 epochs using an early stopping strategy with patience set to 20 epochs. In both evaluation and training, as gold standard keyphrases have multiple tokens, we use each token of the gold keyphrase as the true label for each token.

3. Post-processing. This step reconstructs the multi-token keyphrase from the probability scores as generated by the Glocal model. This formation step then requires a re-ranking (calculating $R(p)$) of the resultant phrase as:

$$R(p) = len(p) * \sum_{w_i \epsilon p} r(w_i) \quad (14)$$

	Schutz			Krapivin			SemEval		
Model	F_1@5	F_1@10	F_1@15	F_1@5	F_1@10	F_1@15	F_1@5	F_1@10	F_1@15
$tf.idf$	11.3	13.7	15.2	6.9	7.3	9.4	9.1	12.2	13.5
TextRank (Mihalcea and Tarau, 2004)	10.2	12.4	14.9	7.6	9.3	9.9	11.2	14.4	15.2
RNN	3.2	3.6	4.0	2.6	2.9	3.6	3.0	3.2	3.7
GRU	3.8	3.2	3.9	3.1	3.4	5.1	2.6	2.8	3.9
CopyRNN	5.8	6.2	7.5	6.6	6.9	7.1	5.4	5.6	6.1
CopyRNN† (Meng et al., 2017)	29.3	30.2	32.2	**26.2**	25.3	**27.1**	28.7	29.4	31.1
GCN (Kipf and Welling, 2016)	16.7	17.8	19.2	19.2	19.8	20.9	18.7	19.5	21.4
GAT (Veličković et al., 2018)	25.2	28.1	29.3	21.1	23.1	24.2	22.5	26.8	25.9
Glocal	**30.7***	30.3	**33.9***	24.7	**25.6**	**27.1**	**28.9**	**29.8**	**33.5***

Table 2: Main comparative system evaluations on keyphrase extraction. All figures are $F_1@K$. † uses only abstract, rest all models are trained on full-text, * shows significant improvement over strongest baseline CopyRNN.

The initial rank of each candidate token is in this case equal to the probability of the $keyphrase$, i.e., $\forall w_i \in p$, $r(w_i) = p_{keyphrase}(w_i)$. We also constrain the process such that the actual word sequence must appear in the original text.

An important note that we strictly do not normalize the β_i scores; the re-ranking process and the preservation of raw scores work in tandem. Topologically, such graphs generated from textual data often have a few dense neighborhoods and many sparse ones, resulting in significant raw score differences that can benefit from scaling down the feature appropriately. If normalization is done in each neighborhood (as done for α_i), it will scale up individual nodes in a sparse neighborhood and suppress nodes in a dense neighborhood, the reverse of the intended operation.

5 Results

We compare our Glocal model's results against other models on the core task of keyphrase extraction. We select baselines which represent the related state-of-the-art under particular learning paradigms: unstructured retrieval-based ($tf.idf$), unsupervised graph-based (TextRank), supervised sequence learning (RNN and derivatives) and supervised structured models (GCN and derivatives). The marked performance differences help us to ablate and understand the gain brought about by the supervision directly on the graph over unsupervised graph-based techniques, as well as that brought by incorporation of the random walk-based score into the supervised model. We also report TextRank and $tf.idf$ baselines to measure ablative effects, as they contribute to Glocal.

For the sake of comparison, we restrict our comparison to modeling approaches, deciding to keep the feature inventory constant; *i.e.* textual and statistical features. Closely related work discussed earlier — such as MIKE (Zhang et al., 2017) — are not directly comparable, since such works may use many other orthogonal features. Similarly, many supervised techniques use additional features. The best reported SemEval-2010 systems show very close performance to our proposed model, but they take advantages of other sources of side information, such as Wikipedia term acquisition, logical section information, bagging, and ensembles; hence, they are not directly comparable (reported in (Kim et al., 2010)). We argue that our main contribution is in the capacity of modeling; other features utilized in prior work can enhance Glocal's performance and suitable incorporation is future work. In a similar vein, PositionRank (Florescu and Caragea, 2017b) enhances the random walk itself which can again act as a replacement for the TextRank in our model and is thus not strictly a comparable method.

We argued earlier that the recently proposed supervised encoder–decoder based CopyRNN (Meng et al., 2017) deep learning models trained using word embedding do not scale well to the full-text setup. In their work, they only report results for extracting (and generating) keyphrases from abstracts. For direct comparison, we have retrained these models in the full-text scenario to validate our claim that these models have difficulty with scaling. Further, we still compare with the abstract-only trained model for CopyRNN to benchmark both approaches.

The results are reported in F_1@K where $F_1 = 2 * precision * recall/(precision + recall)$ and K is the number of keyphrases generated by the model. Table 2 shows that our Glocal model outperforms both GCN and GAT; and that no other text-only based model shows comparable performance. In the full-text scenario, sequential neural models fare comparably to TextRank, Text example from SemEval: Edge Indexing in a Grid for Highly Dynamic Virtual Environments. Newly emerging game-based application systems such as Second Life provide 3D virtual environments where multiple users interact with each other in real-time. They are filled with autonomous, mutable virtual content which is continuously augmented by the users......

Gold annotations (Author-assigned): 3d object stream, object pop problem, spatial index, visibl model
Gold annotations (Reader-assigned): edg index, dynam virtual environ, game-base applic, mutabl virtual content, spatial databas, spatial index method, real-time visibl test,object-initi view model, object pop, 3d spatial extens

TextRank: *virtual environ*, *applic*, user, interac, system
GAT: *environ*, *method*, *databas*, *model*, user
Glocal: **dynam virtual environ**, *real*, **edg index**, game, *applic*

Text example from Schutz: Debugging Larch Shared Language Specifications. The checkability designed into the LSL (Larch shared language) is described, and two tools that help perform the checking are discussed. LP (the Larch power) is the principal debugging tool. Its design and development have been motivated primarily by work on LSL, but it also has other uses (e.g., reasoning about circuits and concurrent algorithms). Because of these other uses, and because they also tend to use LP to analyze Larch interface specifications, the authors have tried not to make LP too LSL-specific......

Gold annotations: debugging, formal specification, parallel programming, inference mechanisms, consistency, design, program debugging, larch power, larch shared language specifications, checkability, concurrent algorithms, development, static semantics, theory containment

TextRank: *larch*, *programming*, program, *language*, *algorithm*
GAT: *parallel*, **parallel programming**, *language*, *larch*, *interface*
Glocal: *larch shared*, **parallel programming**, **design**, **program debugging**, *programming*

Figure 2: Examples from the SemEval and Schutz datasets. Full matches are bolded; partial matches, italicized. Output from SemEval dataset are stemmed before evaluation

but the recent graphical deep learning models represented by GCN and GAT outperforms it easily. Further, notice that TextRank alone performs weakly in comparison to GAT, but does synergize well with the base GAT in our Glocal system yielding state-of-the-art performance over all three datasets. These results are consistent microscopically as well. Fig. 2 gives two illustrations of keyphrases generated by TextRank, GAT, and Glocal, which show the ability of Glocal to pull out a larger ratio of exact keyphrase matches.

We drill down on SemEval dataset for a closer look, as in Table 3, which has the highest positive label ratio at 14.8 keyphrases per document on average. Keyphrases in SemEval are further classified as either author- or reader-annotated. Despite the task being difficult (15% of the reader- and 19% of the author-assigned keyphrases are not in the text) Glocal's performance edge holds out well, and the results are consistent on both forms of assigned keyphrases.

We experiment with adding the neighborhood normalization for β_j which resulted in a significant decrease in performance of the Glocal model. To further explore this issue of whether normalization thus formed classification model has any benefit over GCN and GAT, we experiment with the Cora and Citeseer scientific document classification tasks (as reported in (Kipf and Welling, 2016)

Model	**Author**	**Reader**
KP-Miner	17.1%	21.5%
Maui	16.2%	16.1%
TextRank	14.5%	15.1%
GAT	25.5%	26.0%
Glocal	**32.2%**	**34.5%**

Table 3: Summary of the fine-grained $F_1@15$ on SemEval. We compare with other state-of-the-art SemEval systems using text-only feature-based models.

and (Veličković et al., 2018)) which assigns a document into one of six or seven topical categories. We find that the incorporation of global random walk information does not influence topical categorization much (with a minor gain in the case of Cora dataset), and hence Glocal's performance is almost identical to the less expressive GAT model in terms of classification.

6 Discussion

We now discuss three aspects of the model with respect to the task of keyphrase extraction.

1. Feature versus Scaling. TextRank scores are used in supervised classifiers, traditionally incorporated as a feature. How well would just incorporating TextRank as an additional feature to GAT work? While it does help to improve performance, it does not incorporate the ranking component element in the model; *i.e.* even nodes with very low centrality can positively contribute to its neighbors' scores. Glocal's tight integration is beneficial as it allows further gradient optimization, rather than just simply adding a new feature dimension. Our experiment with adding TextRank as an additional feature show $\approx 2\%$ performance gain for GCN and $< 0.5\%$ performance gain for GAT on average as compared to Glocal showing gain of 5% on average. The behavior is consistent with that of the classification model as in the absence of β_j, the GAT model is inherently a classification model.

As an example (*cf.* Fig. 1, right), consider a neighboring Node h_5 to the prospective keyphrase represented by Node h_1. Node h_5 has a low similarity with the target (low α_{15}), but if a prominent node in the graph will be accorded a higher β_5. In this way, such a node channels more to its neighbor, exerting comparatively steeper gradients to less essential nodes, hence giving a larger chance for Node h_1 to be considered a keyphrase. However, without Glocal's scaling mechanism, such modeling is not captured and essentially unlearnable. We further elaborate on this point next, which shows how our scaling is equivalent to known embedding aggregation techniques (in contrast to adding an extra feature dimension).

2. TextRank Averaged Node Embedding. Kipf and Welling argue that GCN is analogous with node embedding approach with the Weisfeiler-Lehman algorithm for graph isomorphism. Namely, for every node i in V, and scalar feature h_i, the algorithm repeats for k steps or until convergence:

$$\vec{h}_i \leftarrow hash\left(\sum_{j \in \mathcal{N}_i} \vec{h}_j\right). \tag{15}$$

In case $\vec{h}_j$ is a scalar this as well represents the TextRank.

Comparing with layer-wise propagation rules for GAT, GCN, and Glocal, we find a common structure among the methods for incorporating features. Eqn. 16b (representative of GAT) converts a constant aggregation of Eqn. 16a (representing GCN) to be a locally learnable parametric scaling. Eqn. 16c (Glocal) further factorizes the scaling with one locally learnable parametric and one global random walk score. In short, our means of generating node representations (*i.e.*, the post-training embedding) weights the representations for each node/candidate keyphrase by its random walk score. Such a procedure generates stable representations, similar to $tf.idf$ weighted word embeddings used for sentence representation. Such techniques have shown better performance as compared against complex aggregators (like LSTMs) on tasks with insufficient data to train end-to-end models.

$$\vec{h}_i^{(l+1)} \leftarrow \sigma\left(\sum_{j \in \mathcal{N}_i} c_{ij} \vec{h}_j^{(l)} \mathbf{W}\right), \tag{16a}$$

$$\vec{h}_i^{(l+1)} \leftarrow \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \vec{h}_j^{(l)} \mathbf{W}\right), \tag{16b}$$

$$\vec{h}_i^{(l+1)} \leftarrow \sigma\left(\sum_{j \in \mathcal{N}_i} \beta j \alpha_{ij} \vec{h}_j^{(l)} \mathbf{W}\right). \tag{16c}$$

3. Generating Longer Keyphrases. The re-ranking trick for promoting the generation of longer keyphrases – discussed earlier and in the final step of our pipeline – is fielded in many systems (Zhang et al., 2017). This is a difficult requirement to incorporate directly into the model, relegating such techniques to post-processing by default. A nice offshoot effect of our model is that it implicitly favors generating longer keyphrases, due to the inherent nature of the graph convolution operator in aggregating neighboring nodes' features. This, compounded with a high random walk score for the prospective keyphrase node, results in a higher fraction of features propagating from such nodes being included to its neighboring nodes. In turn, this favors dense local keyphrase neighborhoods of highly weighted keyphrases, making the re-ranking step easier.

7 Conclusion

We have presented Glocal, a global plus local graph convolution model for incorporating the global importance of the node within the local convolution operation for supervised learning on graphs. We argue both theoretical and validate empirically that such a model has benefits in strengthening the graph node ranking component, particularly helpful in tasks such as keyphrase extraction. On our detailed experiments on keyphrase extraction on 3 real-world full-text datasets, our model achieves better performance than traditional, graph-based unsupervised graph-based ranking models and bests sequential supervised classifiers as well. The specific component of incorporating global importance further improve the performance by up to 8.0% absolute F_1 on different evaluation criteria on full-text setup as compared to GAT and up to 2% absolute gain as compared to CopyRNN.

Acknowledgments

We would like to acknowledge the support of the NExT research grant funds, supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC @ SG Funding Initiative. We would also like to gratefully acknowledge the support of NVIDIA Corporation with the donation of the GeForce GTX Titan X GPU used for this research.

References

Waleed Ammar, Matthew Peters, Chandra Bhagavatula, and Russell Power. 2017. The AI2 system at SemEval-2017 task 10 (scienceie): semi-supervised end-to-end entity and relation extraction. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 592–596.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 546–555. Association for Computational Linguistics.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.

Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. 2018. Keyphrase generation with correlation constraints. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4057–4066.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of Advances in Neural Information Processing Systems*, pages 3844–3852.

Corina Florescu and Cornelia Caragea. 2017a. A position-biased PageRank algorithm for keyphrase extraction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Corina Florescu and Cornelia Caragea. 2017b. PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1024–1034.

David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics, Posters*, pages 365–373. Association for Computational Linguistics.

Shenhao Jiang, Animesh Prasad, Min-Yen Kan, and Kazunari Sugiyama. 2018. Identifying emergent research trends by key authors and phrases. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 259–269. Association for Computational Linguistics.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26. Association for Computational Linguistics.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic keyphrase extraction from scientific articles. *Language Resources and Evaluation*, 47(3):723–742.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference for Learning Representations*.

Mikalai Krapivin, Aliaksandr Autayeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction. *Technical Report DISI-09-055*.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 582–592. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Proceedings of Advances in Neural Information Processing Systems*, pages 849–856.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1532–1543. Association for Computational Linguistics.

Animesh Prasad and Min-Yen Kan. 2017. WING-NUS at SemEval-2017 task 10: Keyphrase identification and classification as joint sequence labeling. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 973–977.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50. ELRA.

Alexander Thorsten Schutz. 2008. Keyphrase extraction from single documents in the open domain exploiting linguistic and statistical methods. *Master's thesis, National University of Ireland*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of International Conference on Learning Representations*.

Yuxiang Zhang, Yaocheng Chang, Xiaoqing Liu, Sujatha Das Gollapalli, Xiaoli Li, and Chunjing Xiao. 2017. MIKE: keyphrase extraction by integrating multidimensional information. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*, pages 1349–1358. ACM.