Active Learning for GCN based Named Entity Recognition

Anand Bhattad*, Sai Krishna Bollam*, Unnat Jain*
$\{bhattad2, sbollam2, uj2\}$@illinois.edu

***Abstract*—Graph Convolutional Networks (GCN) is a popular topic within Deep Geometric Learning. They have found widespread application including image classification [19], protein-protein interaction prediction [9] and in few shot learning [10]. They have recently been deployed for named entity recognition (NER) [4]. In this work, we test a hypothesis: *Do GCNs aid active learning for text information systems*. For this course project, we restrict active learning schemes for the task of NER. We consider active learning by Least Confidence sampling scheme. We compare this with random sampling scheme. We found that actively sampled samples based on least confidence improves performance. Moreover, we also investigated three different methodologies for incorporating these actively sampled data points - finetuning, training from scratch and training only with new examples. We concluded that finetuning (with actively sampled data points) a model trained on labeled data is the best strategy for NER. The report includes several experiments which help in the inference.**

I. Introduction

Text mining and Natural Language Processing put forward various challenges which involve understanding and processing latent structure in data. Many classical graphical models like Hidden Markov Models, Conditional Random Field have proven successful to capture this structure and run efficient inference over them. The widespread success of deep learning based models on challenging machine learning tasks from computer vision [15], speech recognition [11], machine translation [22] and computational biology [9], has resulted in a resurgence of interest in this area. An efficient way of building these probabilistic models that learns from other examples in the training dataset itself would be the most desired outcome. But the scarcity of well-labeled data is of concern to build robust models that can infer on diverse unseen samples. Labeling these data is laborious and also expensive. If there exists an *oracle* that can provide us labels of certain queried examples, then one must develop clever strategies to request oracle labels for few of the examples, but only few of them so as to capture the variability in the data for an optimal and improved inference.

II. Background

A. Named Entity Recognition

Named Entity Recognition (NER) is one of the most important tasks in information extraction. It deals with identifying and classifying named entities in a text to categories such as location, organization, person, time etc. NER systems are typically statistical machine learning models such as Conditional Random Fields (CRFs) [16]. The performance of these models depends a lot on hand crafted features built based on the underlying grammar. Recently, there has been a surge in deep learning approaches for NER [6], [12], [17], [5]. These approaches doesn't require as much feature engineering. Most of the deep learning approaches use CRF as a decoder in final layer, instead of a softmax function. There is a lot of unannotated data available for NER, as any text can be used. Machine Learning models, however, require a large amount of annotated data to perform well and it is hard to collect annotations. So techniques that can reduce the annotation effort are going to be useful.

B. Active Learning

A general practice in machine learning is gathering a large amount of data randomly from online sources or create them synthetically. This data is then used for training different models to perform some kind of prediction. In the active learning setting, the main idea is to let the model and training algorithm choose the data to learn from. This is achieved by training a model on a very small dataset for which labeling is easier or already available. The model learns to infer some characteristics, but might not perform well on unseen and more complex data. The model, in conjunction with the training algorithm, chooses a subset of examples from unlabeled data to train on strategically. This updated model representation has better performance compared to traditional practices with very less amount of data.

The other obvious advantage of active learning based techniques is the reduced time for collecting large amount of data. This is important, as in practice, it is not always possible to build massive amount of annotated data. It also opens a scope of building models that can completely learn and collect data online.

There are various strategies used in active learning for sampling the unlabeled examples to request labels for. One such strategy is the uncertainty sampling strategy, in which the unlabeled examples are ranked according to current model's uncertainty in predicting label. Intuitively, this means we are selecting the batch of examples which would make the model more proficient than a batch that could have data with known information. Some measures used for identifying the uncertainty:

Least Confidence: Select the examples for which likelihood

* Listed in alphabetical order. Contribution was equal by all authors. Code available at: https://github.com/unnat/active_gcn_ner

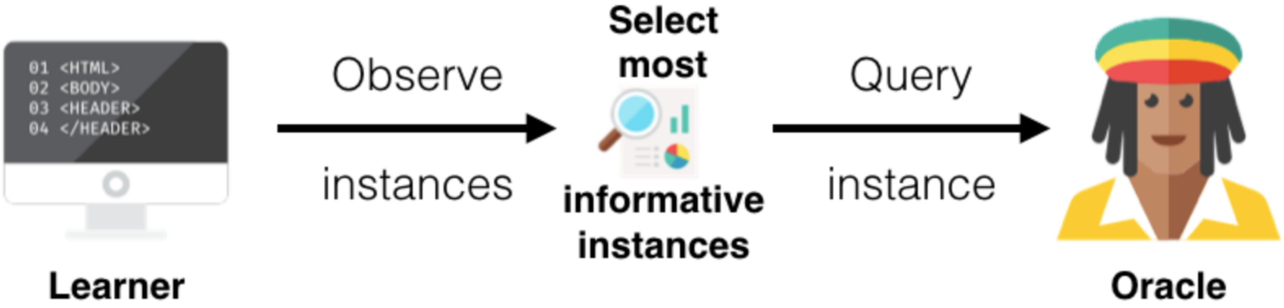

Fig. 1: Active Learning: The main idea is to train the learning model on limited amount of available labeled data and then use this model to strategically ask an oracle to give us labels for few unlabeled samples to improve the overall performance of the model. Image Credit:[1]

of the labels is least. Mathematically, choose:

$$\tilde{\mathbf{x}} = \arg\max_{\mathbf{x}} 1 - P\left(\mathbf{y} \mid \mathbf{x}\right), \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ denote the input sequence and predicted label sequence respectively.
Small Margin:

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} P\left(\mathbf{y}_1 \mid \mathbf{x}\right) - P\left(\mathbf{y}_2 \mid \mathbf{x}\right), \tag{2}$$

where $\mathbf{y}_1$ and $\mathbf{y}_2$ are the most probable predictions for the example $\mathbf{x}$.
Maximum Normalized Log-Probability (MNLP): Using the Least Confidence metric could make the model biased towards selecting longer sentence. [21] proposed a method to normalize the likelihood.

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \arg\max_{\mathbf{y}} \frac{1}{n}\sum_{i=1}^{n} \log P(y_i \mid y_1 ... y_{n-1}, \mathbf{x}) \tag{3}$$

Random sampling strategy is another strategy in which the model passively learns by selecting examples at random.

C. Graph Convolution Networks

Graph Convolution Networks (GCN) aim to generalize well-studied neural models like Recurrent Neural Networks (RNNs) or Convolutional Neural Networks (CNNs) to work on any graph structured data [7], [14], [3], [20], [2]. Such instances of graphs examples includes molecular structures, social networks, knowledge graphs, computer graphics etc. In this section, we briefly review the GCN model variants the relevant update equations.

1) Definition: A graph is a spatial representation of nodes interconnected through edges. Learning problems over graphs consider initial node and edge level features and develop advanced node level features that can capture local neighborhood interactions. These advanced features then can be used for learning node-level inferences, for our case, entity labels. Let the initial feature vector associated with node i be x_i and let A_{ij} be the initial feature vector associated with the edge connecting nodes i and j. Using these initial feature representations of the graph, we want to learn a more refined representation z_i. Drawing a parallel to deep models for computer vision, CNNs apply kernels that process neighboring pixels (the equivalent of nodes) in their receptive field to obtain z_i. Correspondingly, the advanced representation for each node i in the graph could be learned. This could be denoted as the operator $z_i = \sigma_W(x_i, x_{n_1}, x_{n_2}, ...x_{n_k})$ where $x_{n_1}, x_{n_2}, ...x_{n_k}$ are neighbors of node i that define the receptive field of the graph convolution filter. The non-linear activation function σ is parametrized by W, which are the trainable parameters. Multiple layers of these graph convolutions could be stacked to learn more advanced hierarchical representations. This allows integration of information across regions of increasing neighborhood in a graph (more hops).

2) Variants: The following are the variants of this operator proposed in the paper where N_i are neighborhood nodes of the node of interest x_i,

No Convolution: No neighborhood nodes considered in the computation of the advanced features

$$z_i = \sigma(Wx_i + b) \tag{4}$$

Single Weight Matrix: Consider neighborhood node features of N_i in the computation of advanced features z_i for x_i. A single weight matrix is used for x_i and the neighbors [8].

$$z_i = \sigma(Wx_i + \frac{1}{|N_i|}\sum_{j \in N_i} Wx_j + b) \tag{5}$$

Node Average: Consider neighborhood node features of N_i in the computation of advanced features z_i for x_i. Different parameter matrix is used for x_i and the neighbors.

$$z_i = \sigma(W^C x_i + \frac{1}{|N_i|}\sum_{j \in N_i} W^N x_j + b) \tag{6}$$

Node and Edge Average: Consider both neighborhood node and edge features N_i in the computation of advanced features z_i for x_i. Different parameter matrix is used for X_i and the node and edge features.

$$z_i = \sigma(W^C x_i + \frac{1}{|N_i|}\sum_{j \in N_i} W^N x_j + \frac{1}{|N_i|}\sum_{j \in N_i} W^E A_{ij} + b) \tag{7}$$

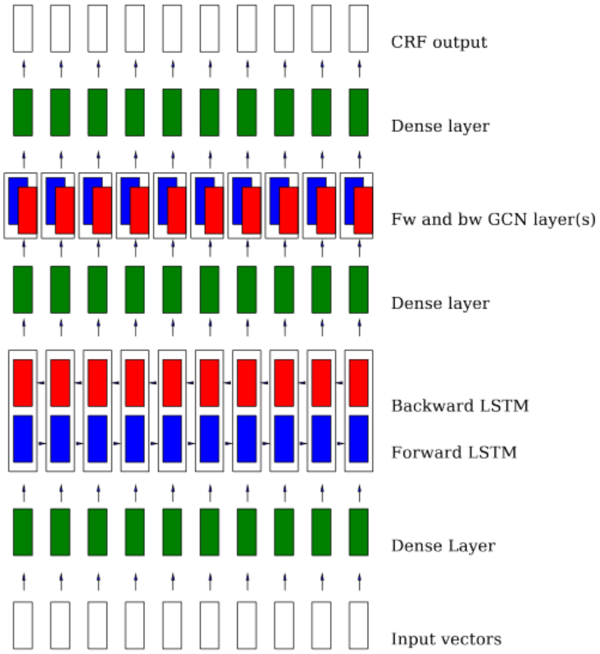

Fig. 2: Our network Architecture; model and image credits [4]

Order Dependent: To compare with order-independent methods, this order-dependent method considers ordering among its neighbors. The order is determined by distance from the node of interest and each neighbor has a unique weight matrix for the edge features based on the order.

$$z_i = \sigma(W^C x_i + \frac{1}{|N_i|}\sum_{j\in N_i} W_j^N x_j + \frac{1}{|N_i|}\sum_{j\in N_i} W_j^E A_{ij} + b) \tag{8}$$

III. Our Approach

A. Network Architecture

We adopt the model architecture from [4] with some tweaks. Fig. 2 summarizes the architecture. Input word features, passed through fully connected layers are fed into the Bi-LSTM for capturing sequential semantics. This better feature representation is passed through a *Bi-GCN* layer and finally a CRF helps predict the final output. Input word features are the concatenation of glove features, POS tags and morphological information. Bi-GCN architecture is the same as that introduced by [18]. The Bi-GCN architecture captures the *directional nature* of the graph. This is obtained by stacking two GCNs, one capturing the effect of incoming edges to a node and another capturing the effect of outgoing edges to a node. The representation of both these GCNs is concatenated.

Improving expressiveness: We improved over the architecture of [4], improving upon their implementation[1] (Fig. 2) by projecting output of both forward and backward GCNs to a non-linear domain using ReLU activations. [4] considers the node interaction in Graph Convolutions linear. We believe, the transfer of knowledge between nodes need not be linear. Our tweaks have zero cost in run time and also add no additional parameters. Hence, we get a small improvement without any model cost.

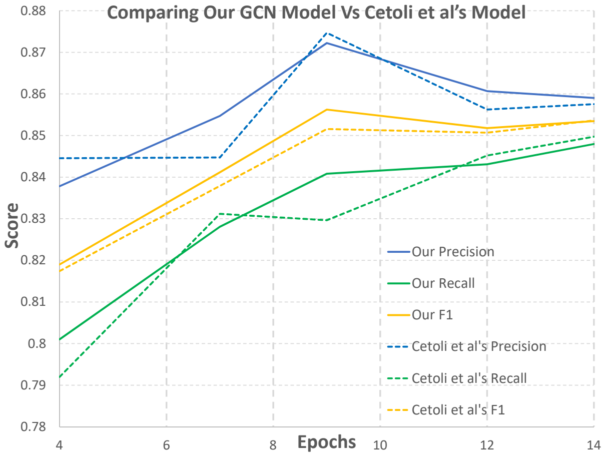

Fig. 3: Our Proposed Base Model: We modify [4]'s GCN network by projecting GCN's node interactions to a non-linear domain at an expense of zero additional parameters. We see that our non-linear projection improves the model performance as compared to [4]. From here, we use our proposed model as the Base model for the active learning experiments.

Evaluating confidence: The model uses the score of LSTM-CRF model for sequence tagging as proposed in [12]. The score of a sentence $[x]_1^T$ for tag sequence $[i]_1^T$ is the total of transition and network scores:

$$s([x]_1^T, [i]_1^T, \theta, A_{ij}) = \sum_1^T \left(A_{[i]_{t-1},[i]_t} + f(\theta, A_{ij})\right),$$

where θ represents the network parameters and A_{ij} is the transition matrix denoting probability of going from tag i to tag j. Computing this exhaustively is intractable, so Viterbi algorithm is used to compute the score.

Normalizing Uncertainty: For uncertainty sampling we chose the *least confidence* metric. Least confident examples can be selected by sorting them in the order of increasing confidence and then choosing the top 5%. However, the score (without normalization) is always smaller for large sentences. So we normalize the score by the length of the sequence and use the *length-normalized score* as the estimate of model confidence. We observed a good mix of sentence lengths in the selected samples this way.

B. Active Learning Setup

To simulate an active learning framework, we randomly chose 20% of the labeled data (N data points) to train the base model on and assume the remaining as unlabeled. We then pick, by two schemes - least confidence sampling and random sampling. A sample size 5% of N examples is sampled and

[1] https://github.com/ContextScout/gcn_ner

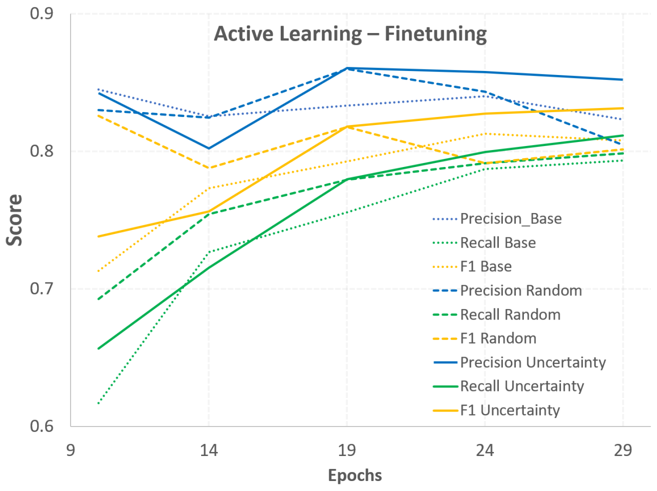

Fig. 4: **Scenario 1** (Finetuning): Precision, Recall and F1 scores. We first train our base model on 20% of the total data. Using this base model we sample 5% more data based on their uncertainty estimate. We compare them with random sampling approaches. Note that the final model is initialized with the pretrained weights of the base model.

their labels are uncovered by the *oracle*. Now we test these samples in three scenarios.

1) *Finetuning:* We use the actively sampled data points along with the labeled data to train the final network. The learning starts with the model initialized with the pre-trained weights of the base network.
2) *Training afresh:* We use the actively sampled data points along with the labeled data to train the final network. The learning starts *from scratch*, *i.e.*, no initialization.
3) *Training only on sampled data:* We use only actively sampled data points to train the final network. The learning starts with the model initialized with the pre-trained weights of the base network. This is useful for transfer-learning based applications where the data used to pretrain the network is no longer available.

IV. Experiments

A. Dataset

We run the experiments on OntoNotes-5.0 [23] English dataset which contains manually curated tags for entities in the training set. It has a total of 75,187 sentences in training, 9603 sentences in development and 9479 sentences in test sets. The data was converted to CoNLL format, which encapsulates the syntactic graph structure of each sentence along with the entity labels.

B. Quantitative Evaluation

We use three metrics precision, recall and F1-score given by the implementation of [4]. Precision is the percentage of selected labels that are correct for an entity. Recall measures the percentage of correct labels that are identified for an entity. F1-score is the harmonic mean of precision and recall. We evaluate the active learning capacity of graph convolution neural networks in three different experiment scenarios as explained in Sec. III-B.

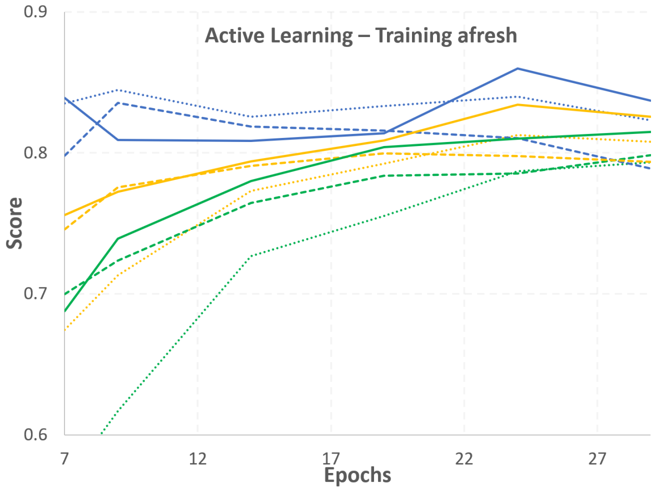

Fig. 5: **Scenario 2** (Trained afresh): Precision, Recall and F1 scores. We first train base model on 20% of the total data. Using their trained weights we sample 5% more data based on their uncertainty estimate. We compare them with random sampling approaches. Note that the final model is *re*-trained completely. Legend is same as Fig. 4

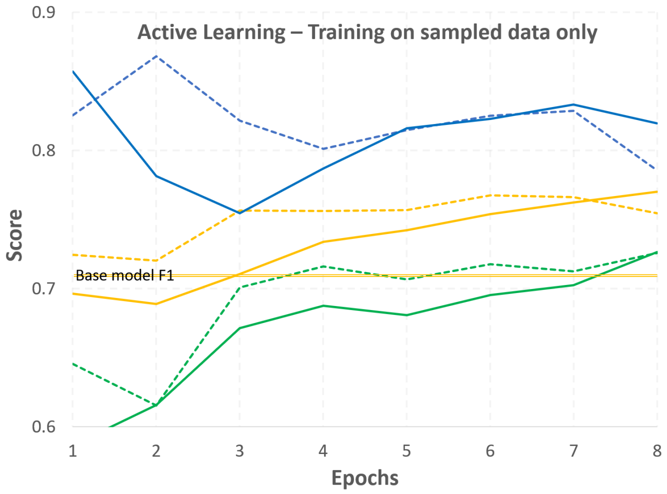

Fig. 6: **Scenario 3** (Training on only actively sampled data points): Precision, Recall and F1 scores. We first train our model on 20% of the total data (labeled data). Using their trained weights we sample 5% of the total data based on their uncertainty estimate. We compare them with random sampling scheme. Legend is same as Fig. 4

In our first experiment (*finetuning*), we train our Graph Convolution Network as discussed in Section III moderately well by using only 20% of the training data available. Using this trained weights we request labels of 5% more samples by the uncertainty approach obtained by smoothening and normalizing CRF output score as described in III-A. For the random sampling based active learning approach, We request labels of 5% of random sentences. Using these pre-trained weights as initialization, we finetune our network prediction. We report our results in Figure 4. We find uncertainty based approach outperforms random sampling approach in all the three metrics.

In the second experiment (*train afresh*), using the same additional samples obtained from uncertainty based approach and random sampling as before, we retrain our network from scratch. We find in the initial learning phase random samples to have an edge over uncertain samples. This is because of the fact that learning curve of the uncertain samples is harder as compared to random samples. But after few epochs, uncertainty based approach consistently outperforms random sampling. Comparing to fine-tuning, not surprisingly, fine-tuning outperforms training from scratch.

In the third experiment (*training on only sampled data*), we trained the network only on actively learned samples to see how they perform on the test dataset (*dev* set of *OntoNotes 5.0*). This scenario is important for cases when training data is unavailable for various reasons and a pretrained model is available. The task is to leverage the available model to collect new data samples and retrain the network for improved performance. Against the previous norm, we find that uncertainty based method doesn't perform well in this scenario as compared to random sampling. We propose an intuition for this observation. Training on a harder dataset (least confident examples), unlearns some important features that the model learned from the original training data. Whereas, in the random sampling, training with them doesn't necessarily breaks the previously learned useful representations. Please refer Figure 6

C. *Implementation details*

We use the same hyperparameters as [4]. The embedding dimension for tags is 15 and 300 for word embedding. The dense layers before LSTM project each one-hot input to a 40 dimensional vector. The recurrent units in both forward and backward layers are GRU cells with a memory dimension of 160 and a 2 layer stack. This forms the encoder part of the network. The decoder has a dense layer projecting the output of recurrent units to same dimension of 160. The bi-directional GCN layers use a hidden size of 200. Final feed-forward layer concatenate the outputs from both forward and backward GCNs and project them to a dimension of 200. The output dimension is 19 (number of NER tags).

Training is performed using mini-batch stochastic gradient descent with Adam optimizer [13] and a fixed learning rate of $1e^{-4}$. The training loss for back-propagation is computed using cross-entropy loss metric.

V. Conclusions and Future Work

In this work, we investigated a yet unexplored aspect of actively learning for text information systems. We chose the task of parsing dependencies for Named Entity Recognition (NER). We evaluate the impact of Graph Convolutional Networks (GCNs) for active learning based NER. We find the GCNs are indeed effective in indicating which unlabeled examples should we uncover labels for. Uncertainty sampling proves more effective than random sampling in improving performance over the unseen test set.

Due to computational and time resources, we limited the scope of this project. Going forward, we would want to use other text information tasks (beyond NER) as the test bed of for our approach. Also we would be excited to explore more sophisticated active learning methodologies.

Acknowledgements: The authors wish to thank ChengXiang Zhai, Qihao Shao and Bingjie Jiang for supporting us throughout the CS410 course. A special thanks to Ismini Lourentzou for sitting with us for multiple meetings and giving us a chance to present Graph Neural Networks to TIMAN's deep Learning group on 30th April 2018.

References

[1] Active-learning tutorial, datacamp community. 2

[2] Atwood, J., and Towsley, D. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems* (2016), pp. 1993–2001. 2

[3] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine 34*, 4 (2017), 18–42. 2

[4] Cetoli, A., Bragaglia, S., O'Harney, A., and Sloan, M. Graph convolutional networks for named entity recognition. *arXiv preprint arXiv:1709.10053* (2017). 1, 3, 4, 5

[5] Chiu, J. P., and Nichols, E. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308* (2015). 1

[6] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. Natural language processing (almost) from scratch. *Journal of Machine Learning Research 12*, Aug (2011), 2493–2537. 1

[7] Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems* (2016), pp. 3844–3852. 2

[8] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems* (2015), pp. 2224–2232. 2

[9] Fout, A., Byrd, J., Shariat, B., and Ben-Hur, A. Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems* (2017). 1

[10] Garcia, V., and Bruna, J. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043* (2017). 1

[11] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine 29*, 6 (2012), 82–97. 1

[12] Huang, Z., Xu, W., and Yu, K. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015). 1, 3

[13] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 5

[14] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016). 2

[15] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105. 1

[16] Lafferty, J., McCallum, A., and Pereira, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 1

[17] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* (2016). 1

[18] Marcheggiani, D., and Titov, I. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826* (2017). 3

[19] Marino, K., Salakhutdinov, R., and Gupta, A. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844* (2016). 1

[20] Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., and Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nature communications 8* (2017), 13890. 2

[21] Shen, Y., Yun, H., Lipton, Z. C., Kronrod, Y., and Anandkumar, A. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928* (2017). 2

[22] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (2014), pp. 3104–3112. 1

[23] Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., Xue, N., Taylor, A., Kaufman, J., Franchini, M., El-Bachouti, M., Belvin, R., and Houston, A. Ontonotes release 5.0. 4