Multi-Label Patent Categorization with Non-Local Attention-Based Graph Convolutional Network

Pingjie Tang,[1*] **Meng Jiang,**[1] **Bryan (Ning) Xia,**[1] **Jed W. Pitera,**[2] **Jeffrey Welser,**[2] **Nitesh V. Chawla**[1†]
[1]University of Notre Dame, Notre Dame, Indiana 46556, USA
[2]IBM Research - Almaden, 650 Harry Road, San Jose, California 95120, USA
{ptang, mjiang2, nxia, nchawla}@nd.edu, {pitera, welser}@us.ibm.com

Abstract

Patent categorization, which is to assign *multiple* International Patent Classification (IPC) codes to a patent document, relies heavily on expert efforts, as it requires substantial domain knowledge. When formulated as a multi-label text classification (MTC) problem, it draws two challenges to existing models: one is to learn effective document representations from text content; the other is to model the cross-section behavior of label set. In this work, we propose a label attention model based on graph convolutional network. It jointly learns the document-word associations and word-word co-occurrences to generate rich semantic embeddings of documents. It employs a non-local attention mechanism to learn label representations in the same space of document representations for multi-label classification. On a large CIRCA patent database, we evaluate the performance of our model and as many as seven competitive baselines. We find that our model outperforms all those prior state of the art by a large margin and achieves high performance on $P@k$ and $nDCG@k$.

Introduction

Patent databases are valuable information sources reflecting universal inventive trends and evolution of real-world technologies. A large number of new patent applications arrive at patent offices around the world every day. Professional patent examiners are trained to assign proper category codes to patents so as to facilitate patent search and management (Gomez and Moens 2014). However, manually categorizing large volume of patents requires expensive human labor and examiners' broad domain knowledge. Furthermore, complex and hierarchical concepts of the patent category codes impose significant challenges on the categorization task.

IPC code[1] is the most commonly used taxonomy system for categorizing patents. It includes thousands of categories and is defined in a hierarchical way. IPC code is prefixed by a capital letter from "A" to "H", followed by a two-digit number and ends with a letter in uppercase. Take code "F01P" as a concrete example whose concept can be interpreted hierarchically. "F" stands for general concepts "mechanical engineering, lighting, heating, weapons and blasting", "F01" represents sub-concepts "machines or engines in general", and "F01P" indicates fine-grained concepts "cooling of machines or engines". Eventually, "F01P" will be assigned to relevant patents.

Claim 10. A process according to claim 9 wherein the substrate comprises a wind blade.
Claim 11. A process according to claim 1 wherein the viscosity of the wet resin is at most 275 Pa·s at a shear rate of 100 s^{-1} measured at a temperature of at least one of 5 °C.

Figure 1: An example of patent claim citation

Most patents consist of multiple related aspects. Such characteristic makes patents associated with multiple IPC codes (labels). Also, labels are not exclusive, actually, the number of related categories can range from a few to even hundreds depending on the application concepts a patent contains. This presents the compelling challenge for categorization of patents. There are many possible set of labels that can be assigned to each patent, and thus the the task of automated patent categorization becomes a multi-label text classification (MTC) problem.

Unique characteristics of patent's document and IPC codes bring unique challenges to the patent MTC task. First, from the perspective of a patent expert, words play the key role in the patent categorization, so a model should learn predictive word representations from the word co-occurrence and document-word associations. Second, patent document is often consisted of multiple "claim" sections and there are long-range dependencies between claims. For example, Fig. 1 shows that *claim 11* cites *claim 1* across a number of claims[2]. It requires a careful mechanism to model such long dependencies. Third, the IPC code covers a vast and sometimes disparate areas, creating thereby ambiguous semantic interpretations. For example, instead of defining one clear and consistent concept, IPC codes prefixed by "G09" refers to "Educating, Cryptography, Display, Advertising, Seals." The label representation is a mixture of topics that can only be learned from document-label associations.

To address the above challenges, we propose a novel

[*]This work was partially done when the first author was interning with IBM Research - Almaden.
[†]Nitesh V. Chawla is the corresponding author.

[1]https://www.wipo.int/classifications/ipc/en/
[2]https://patents.google.com/patent/US9381707

Graph Convolutional Network (GCN) based model with an adaptive non-local label attention layer. We build a textual graph based on word co-occurrence and document-word associations and then learn a GCN that generates word embeddings. Next, the proposed model has a non-local attention layer that achieves two goals. The first is to learn the long-range semantic dependencies in the patent content. The second is to embed the semantic information as representation for each individual label. We summarize our contributions as follows:

- We propose a new deep learning model based on GCN to capture rich semantic information for addressing the challenges in the multi-label patent text classification.
- We design an adaptive non-local second-order attention layer to model long-range semantic dependencies in text content as label attention for patent categories.
- We conduct extensive experiments to evaluate our model. We compare it with seven competitive baselines. Results show that our model consistently wins on evaluation metrics such as $Precision@K$ and $nDCG@K$.

Related Work

Traditional MTC Models

One-vs-all classifiers. For each label, a separate binary classifier is trained independently to make predictions, which increases the computation complexity while dealing with huge number of labels. (Yen et al. 2017; 2016) use L_1 regularization to fulfill a sparse solution so as to reduce the computational complexity. (Babbar and Schölkopf 2017) learns a linear classifier per label and uses double layer of parallelization to control the model size. Another visible limitation of this method is its ignorance of label dependencies during training, which may weaken the model's generalizability. To address this limitation, (Zhang and Zhang 2010) adopts the Bayesian network to encode the label dependencies by computing the joint probability of label and feature sets.

Tree-based classifiers. Those models are inspired by the ideas of decision tree and build decision trees based on either labels or data instances by recursively splitting internal nodes. Predictions are made while a new data point is passed down the tree until reaching the leaf nodes. FastXML (Prabhu and Varma 2014) achieves significant accuracy by optimizing a ranking loss function directly. PfastreXML (Jain, Prabhu, and Varma 2016) is an extension of FastXML, it prioritizes prediction of tail labels and handles missing labels by proposing the propensity scored loss. Parabel(Prabhu et al. 2018) generates an ensemble of balanced trees over labels rather than data points, which could be considered as a generalized method of hierarchical softmax model to boost the performance.

Embedding based models. Embedding based methods employ compression functions to project label embedding to a lower dimensional linear subspace. SLEEC (Bhatia et al. 2015) is proposed to solve the limitation that low-rank label matrix assumption is usually violated in real world applications. AnnexML (Tagami 2017) presents a graph embedding method to cope with several limitations of SLEEC.

Deep Learning MTC Models

(Liu et al. 2017) adapts (Kim 2014) to a MTC task by adding a dynamic max pooling to catch more fine-grained information and uses a bottleneck hidden layer to reduce parameters size. (Nam et al. 2017) converts predicting a set of related labels into predicting sequence of binary values and uses Recurrent Neural Network (RNN) to make the prediction. (Yeh et al. 2017) proposes to employ canonical correlated autoencoder to jointly model textual features and label structures. (Zhang et al. 2018) builds a label graph that attempts to explore the label space. (Wang et al. 2018) learns a joint word-label embedding and uses the compatibility scores between each label and context words as attention coefficients to combine context vectors into one final document vector. Similarly, (You et al. 2018) uses multiple label attention vectors to allow the network to attend on multiple semantics for each label. It produces multiple label attention vectors, and each label vector is mapped to a single output. (Tang, Qu, and Mei 2015) builds a textual heterogeneous network to encode multi-level semantic information. Graph Convolutional Network has gradually gained popularity in multi-class classification. (Rousseau, Kiagias, and Vazirgiannis 2015) and (Yao, Mao, and Luo 2019) use GCN to jointly learn word and document embeddings over a graph textual representation. (Chen, Lin, and Cho 2017) applies GCN over a label graph to extract the label-wise information. However, GCN for multi-label text classification tasks has rarely been studied yet.

Proposed Model

Figure 2 presents the architecture of our proposed model, A-GCN+A-NLSOA, consisting of two components. *Attention-based GCN*, which is equipped with sufficient expressive power to allow the network to learn informative representation of words by capturing document-word associations and word co-occurrence over a textual graph. The second part is an *adaptive non-local second-order attention layer*, which can be utilized to effectively capture non-local and fine-grained semantic relations from the text representations produced by GCN. The attended semantic information also provides representations for labels. Therefore, the second part can also be seen as a form of label attention.

Graph-based Text Representations

In our task, capturing document-word association and word co-occurrence in the patent corpus is crucial. We build an undirected patent textual graph denoted as $G_{pw} = (V_p \cup V_w, E_{pw} \cup E_{ww})$ to encode those information aforementioned. G_{pw} consists of patent nodes V_p, word nodes V_w and patent-word edges E_{pw} as well as word-word edges E_{ww}. Edges E_{pw} are built between a patent and any word in it. There are no weights attached on E_{pw}. We leverage Pointwise Mutual Information (PMI) to compute the scores in order to build E_{ww} edges. PMI score is widely used in computational linguistics to capture collocations and associations between words (Yao, Mao, and Luo 2019). PMI between

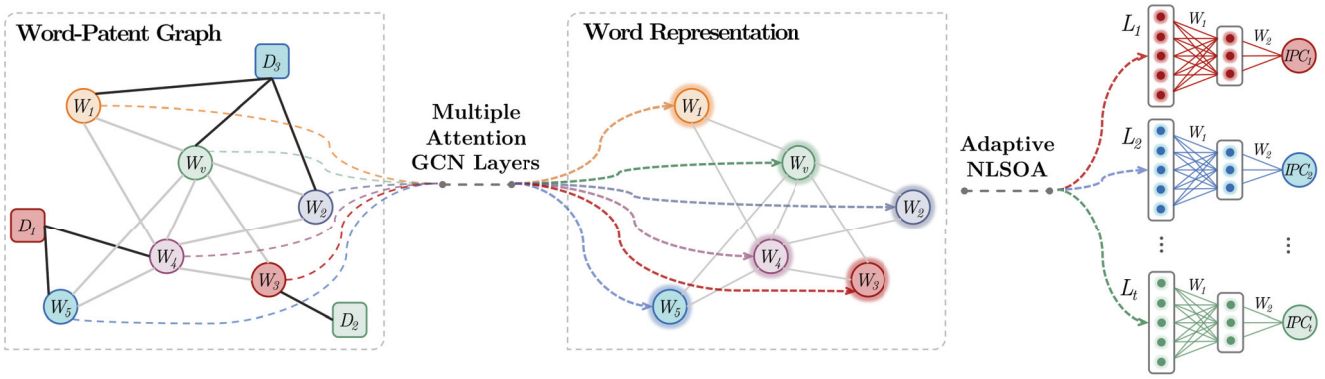

Figure 2: Overall Architecture of Non-local Attention-based Graph Convolutional Network

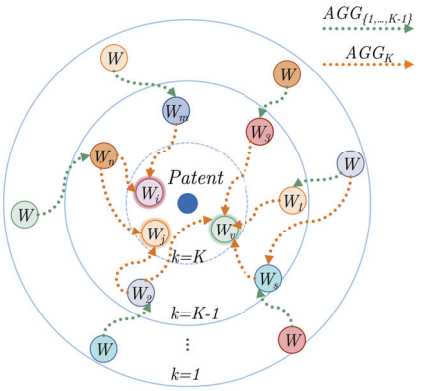

Figure 3: Node embedding generation process using GCN. Patent's (central) word nodes ($k = K$) are updated by recursively aggregating neighboring word nodes from search depth $k = 1$ to $K - 1$.

two words is defined as

$$PMI(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)} \quad (1)$$

We keep positive PMI scores and replace negative PMI values with zero as negative values could be problematic. Thus an E_{ww} edge is built only when PMI score is positive. We use a fixed size sliding window to collect the word co-occurrence statistics across the entire corpus.

$$p(w_i, w_j) = \frac{C(w_i, w_j)}{N} \quad (2)$$

$$p(w_*) = \frac{C(w_*)}{N} \quad (3)$$

$C(w_i, w_j)$ denotes the number of times words w_i and w_j co-appear in a sliding window. N is the total number of sliding windows in the corpus, $C(w_*)$ represents the number of sliding windows containing word w_*.

Attention-based Graph Convolutional Network

Spectral-based GCN models have only been used in the transductive setting (Kipf and Welling 2016), we apply GraphSAGE (Hamilton, Ying, and Leskovec 2017) to learn the graph node embeddings. GraphSage is a non-spectral GCN which provides support on batch-training without updating states over the entire graph and has obtained empirical success compared to other graph representation learning models. Figure 3 illustrates the general idea of the node embedding generation process of GCN over graph G_{pw}. The model recursively updates embedding for node v by (1) Aggregating information from node v's immediate neighbors denoted as $N(v), u \in N(v) : (u, v) \in E$, via the aggregator functions $AGG_k, \forall k \in 1, ..., K$, k is defined as search depth (hop). (2) Node v's representation $\vec{h}_v^k$ is updated by concatenating the aggregated neighborhood vector $\vec{h}_{N(v)}^k$ and its own node hidden state $\vec{h}_v^{k-1}$. When k = 1, each word node's embedding from previous search depth, $\vec{h}_v^{k-1}$, is initialized with pre-trained word embedding. The entire process is demonstrated by Eq.(4) and Eq.(5).

$$\vec{h}_{N(v)}^k \leftarrow \text{AGG}_k(\vec{h}_u^{k-1}, \forall u \in N(v)) \quad (4)$$

$$h_v^k \leftarrow \sigma\left(W^k \cdot \text{CONCAT}(h_v^{k-1}, h_{N(v)}^k)\right) \quad (5)$$

The following highlights the details regarding the adaptation of the GCN model to the patent MTC task.

It is worth noting that even though PMI can preserve word co-occurrence information to some extent, such frequency-based statistics is insufficient to capture various significance of words. Inspired by (Veličković et al. 2017) we design an attention-based BiLSTM layer as the aggregator corresponding to $AGG_{1,...,K-1}$ in Figure 3 to aggregate information from co-occurring words to update the target word v, and allows neighboring words to receive different emphasis. Figure 4 shows the basic architecture of this aggregator. Given s neighboring word embeddings, $\{\vec{h}_1^{k-1}, \vec{h}_2^{k-1}, \ldots, \vec{h}_s^{k-1}\}$ of the word v, we take the advantage of expressive ability of bidirectional long-short-term-memory network (BiLSTM) and apply it to such word sequence in order to first obtain discriminative embeddings. In

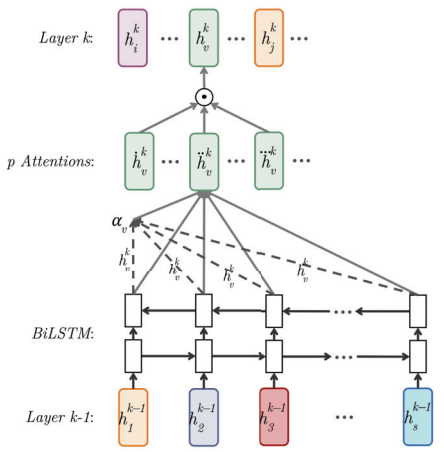

Figure 4: Aggregator to integrate information from co-occurring words in the neighborhood of the target word

order to compute the attention coefficients to represent the importance of each neighboring word node u for the target word v, an attention mechanism can be implemented by one linear layer parametrized by a weight vector $\mathbf{W_a}$, and a non-linear activation function such as LeakyReLU (with negative slope $\alpha = 0.1$) could be used. The fully expanded equation shown in (6) expresses the normalized attention coefficients α_{vu} using the softmax function,

$$\alpha_{vu} = \frac{exp(\sigma(\mathbf{W_a}^T[\text{CONCAT}(\vec{h}_v^k, \vec{h}_u^{k-1})]))}{\sum_{i \in N(v)} exp(\sigma(\mathbf{W_a}^T[\text{CONCAT}(\vec{h}_v^k, \vec{h}_i^{k-1})]))} \tag{6}$$

Target node v's representation vector can hence be derived from the linear combination of the immediate word neighbors by using the obtained attention coefficient α_{vu} after applying a nonlinear function σ.

Multi-head attention allows the model to jointly attend to information from different representation subspaces in parallel. We use multi-head self-attention to concatenate the results obtained from each single attention head. Eq.(7) details the final representation for the target word node v after employing the multi-head attention mechanism. P denotes the number of independent attention heads.

$$\vec{h}_v^k = \Big\Vert_{p=1}^{P} \sigma\left(\sum_{u \in N(v)} \alpha_{vu}^p \vec{h}_u^{k-1}\right) \tag{7}$$

We derive a GraphSage variant by replacing Eq. (4) and (5) with Eq.(6) and (7).

Since we need to establish connections between word representations with individual labels, we keep the sequence of word embedding vectors recursively updated from words in its vicinity instead of aggregating them into a single vector for the patent. We use BiLSTM to replace aggregator AGG_K as shown in Figure 3. Furthermore, it is crucial to maintain word order in our case. To this end, instead of uniformly sampling words appearing in a patent as its neighboring nodes as suggested in (Hamilton, Ying, and Leskovec 2017), we select top n words out of each patent to be its local neighbors without disturbing original word order.

Adaptive Non-local Second-order Attention Layer

The way to build immediate connections between words by estimating PMI values also lacks the capability to capture the long-range information due to the fact that the word pair frequency is calculated only for the words appearing within the same window (see Eq.(1)). However, the phenomenon that some semantic correlations can only be captured across multiple sentences or even paragraphs is particularly evident in the patent corpus, because there are strong reference relationships across different patent claims as illustrated in Figure 1. Moreover, the fact that IPC labels specify cross-domain concepts requires the model to capture more thorough information in order to establish the correlations between subtle document semantics and individual label.

Recent effort takes advantage of Non-Local Neural Networks and second-order information to achieve promising results in computer vision applications (Xia et al. 2019). However, it only provides an attention mechanism to model feature spatial correlations, and it fails to incorporate label supervision directly. We observe that the attention maps offer natural separation for salient regions, the attended feature vectors produced using these attention maps can be helpful for individual label supervision. We hence propose an adaptive non-local second-order attention (NLSOA) module, which not only enhances our model's capability of capturing long-range semantic dependencies and fine-grained information, but also establishes the direct connection between the separated information and individual labels. Therefore, it can also be seen as a form of label attention. We leverage the covariance matrix, instead of the commonly used inner-product operations, as a pairwise affinity function because covariance matrix can better capture the position-wise correlations among words and thus generate discriminative representations, which is designed as Eq.(8),

$$\mathbf{\Sigma} = \theta(\mathbf{X})\bar{\mathbf{I}}\theta(\mathbf{X})^T \tag{8}$$

where the input tensor $\mathbf{X} \in \mathbb{R}^{s \times c}$ consists of word sequence embeddings for patent generated from GCN. s is sequence length, and c is the feature dimension. We use a linear layer followed by a batch normalization layer and a LeakyReLU layer to form a function called θ, which reduces $\mathbf{X}$'s feature dimension from c to c/r. $\bar{\mathbf{I}} = \frac{1}{c/r}(\mathbf{I} - \frac{1}{c/r}\mathbf{1})$, $\mathbf{I}$ and $\mathbf{1}$ are identity matrix and all-one matrix respectively, and $\mathbf{\Sigma} \in \mathbb{R}^{s \times s}$. We then design a self-attention utilizing $\mathbf{\Sigma}$ to compute the response at a position in $\mathbf{\Sigma}$ to model fine-grained associations among words. We use another linear layer g as a transformation function to squeeze $\mathbf{X}$'s feature dimension from c to c/r , and the resulting $g(\mathbf{X})$ has shape of $s \times (c/r)$. We adopt $\frac{1}{\sqrt{c/r}}$ as the normalization factor for the covariance matrix before applying softmax, which yields (9):

$$\mathbf{Z} = softmax(\frac{\mathbf{\Sigma}}{\sqrt{c/r}})g(\mathbf{X}) \tag{9}$$

We use a trainable transformation p to excite the reduced dimension c/r to t and add a skip transformation ϕ to map input feature dimension to t. Finally, the Adaptive NLSOA module can be formulated as Eq.(10):

$$\mathbf{X}' = \phi(\mathbf{X}) + p(\mathbf{Z}) \tag{10}$$

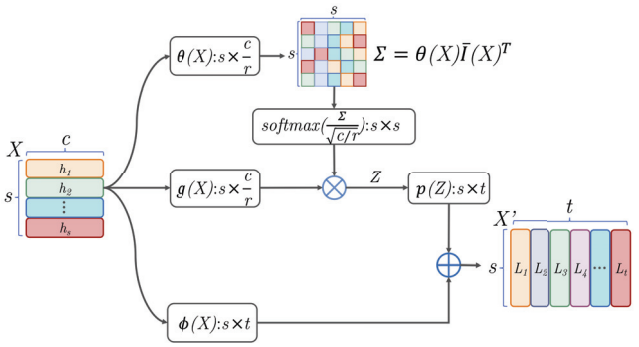

Figure 5: Adaptive Non-local Second-order Attention Layer

The Adaptive NLSOA has three appealing properties. Firstly, it uses only one branch of convolution layer's output to compute the covariance matrix. Moreover, it is capable of obtaining long-distance as well as fine-grained information from the covariance matrix. If we view the output of Adaptive NLSOA layer as a stack of t column vectors, as illustrated in Figure 5, each of these column vectors encodes the non-local semantic information, and can be mapped to a specific label. Thus Adaptive NLSOA can be interpreted as a form of label attention.

As aforementioned, IPC label embodies cross-domain concepts and thus is ambiguous. Namely, patents corresponding to various application domains can be associated with the same IPC label. In order to address this issue, for each patent document, we directly model the relationship between words and each individual label using the Adaptive NLSOA, which provides each label with particular semantic information inside the patent. We use two consecutive fully connected (FC) hidden layers $\mathbf{W_1} \in \mathbb{R}^{s\times f}$ and $\mathbf{W_2} \in \mathbb{R}^{f\times 1}$ following the Adaptive NLSOA layer. We use ReLU and softmax as two nonlinear activation functions σ_1 and σ_2 applied on the last two layers to make the final predictions. In order to avoid using more parameters, similar to (You et al. 2018), the parameters in W_1 and W_2 are shared across all labels. The resulting j-th label probability $\hat{y}_j$ could be obtained by

$$\hat{y}_j = \sigma_2(\mathbf{W_2}^T \sigma_1(\mathbf{W_1}^T L_j)) \tag{11}$$

where L_j is the j-th output column vector of the Adaptive NLSOA layer. Finally, we utilize a binary cross-entropy loss function which is a frequently used objective function tailored to the MTC task and formulated as in Eq.(12). Unlike (Liu et al. 2017), which uses the sigmoid function to produce the class probability, we find that softmax yields better results than sigmoid. It is also observed in work (Mahajan et al. 2018).

$$J = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{t}[y_{ij}log(\hat{y}_{ij}) + (1-y_{ij})log(1-\hat{y}_{ij})] \tag{12}$$

N is the number of training examples and t is the number of labels, y_{ij} is the binary ground truth label value, and $\hat{y}_{ij}$ is the predicted probability value demonstrating how likely the j-th label should be assigned to the i-th patent.

Table 1: Basic Statistics of Patent_CIRCA_45k Dataset

Dataset	N	D	L	$\bar{\mathbf{L}}$	$\hat{\mathbf{L}}$
Train	36,420	81,613	550	2.428	159.330
Test	9,106	38,344	443	2.430	39.868

Experiments

Datasets

We query 45,526 patents with 555 IPC codes from an internal patent database named "CIRCA". A patent document is primarily constituted by a patent number, bibliographic information, title, abstract, description and the claim sections and other metadata. Each patent includes one or more IPC codes to classify the patent contents in a uniform manner. We only use title, abstract and the claim which are viewed as the most informative textual sections to describe a patent and entire IPC labels assigned to each patent. We use NLTK to preprocess the data. We eliminate words containing special tokens and remove words appearing less than 5 times. Therefore there are totally 89,784 distinct words remain after the data cleaning. We use Gensim Word2vec library[3] to train the 100-dimensional patent word embedding model. For notational convenience, we name the dataset as "Patent_CIRCA_45k", and Table 1 provides the basic statistics of it. We split the data into training and testing in a 80/20 ratio, in actual experiments, we hold 10% training data as the validation set (unlisted in Table 1) that is used to choose the optimal parameters. In Table 1, **N** represents the number of data instances in the training and testing sets. **D** is the feature size which is equivalent to the number of words. **L** indicates the label count, $\bar{\mathbf{L}}$ stands for the average number of IPC labels for each data sample. $\hat{\mathbf{L}}$ represents the average number of documents per IPC label.

Baseline Methods

We compare our model to seven competitive state-of-the-art methods. FastXML (Prabhu and Varma 2014), PfastreXML (Jain, Prabhu, and Varma 2016) and Parabel (Prabhu et al. 2018) are frequently used strong tree-based algorithms in extreme MTC domain. We exclude embedding-based and one-vs-all methods because tree-based methods outperform them on CIRCA dataset. For those three algorithms, we use the code provided by the online extreme classification repository[4]. Besides using bag-of-words as the features suggested by those algorithms in the paper, we enrich the features by integrating with phrases extracted by (Shang et al. 2018) to boost their performance. We include representative deep learning based MTC models, such as XML-CNN (Liu et al. 2017), and the classic BiLSTM. Specifically, we use 512 hidden neurons for BiLSTM which takes word embedding as input, and utilize softmax and binary cross entropy loss to make the prediction. In order to evaluate the effectiveness of capturing the document-label relation, we compare our model with AttentionXML (You et al. 2018). Since we use the graph neural network as the building block for our

[3]https://radimrehurek.com/gensim/models/word2vec.html

[4]http://manikvarma.org/downloads/XC/XMLRepository.html

model, we compare our model against GraphSAGE (Hamilton, Ying, and Leskovec 2017) as well. We implement all deep learning models using the PyTorch framework (Paszke et al. 2017). Furthermore, we design experiments to investigate impacts on the final performance due to the hyperparameters variation, such as size of the sampled neighboring nodes, different number of attention heads.

Experimental Setup and Evaluation Metrics

For all experiments, we use 256 hidden units in BiLSTM for GCN aggregators. We choose maximal search depth k as 2, a recommended setting in (Hamilton, Ying, and Leskovec 2017), which provides a consistent performance boost. We reduce the internal feature dimension to half of the input feature dimension in the Adaptive NLSOA module to compute the covariance matrix as attention maps, and the attended tensor is mapped back to the original input feature dimension. We only use the covariance matrix rather than the power normalized covariance matrix in (Li et al. 2017). In our model, we observe a better performance without the power normalization. Also, we use the same two FC layers (512 hidden units and 256 hidden units, respectively) for all label vectors. We use ReLU and Dropout (0.5 rate) between these FC layers. We train our model with 128 batch size and Adam optimizer[5] with weight-decay of 1.0e-6 to accelerate the training process for fast convergence. We also apply a warming up strategy with an initial learning rate 2.4e-5 and increase by 2.4e-5 every 2 epochs until it reaches 2.4e-4 at epoch 20.We then reduce the learning rate to 2.4e-5 for the remaining 10 epochs. It takes about 2 minutes per epoch for the entire 30 epochs to obtain our best model, which has 20 attention heads, 100 word nodes for each patent and 10 neighboring words for each word. We use two Nvidia Titan Xp GPUs and keep the settings suggested by all those competing methods. To calculate PMI scores, we choose 20 as the sliding window size as it leads to the optimal result compared to size of 5, 10, 20 and 25.

We employ the ranking-based metrics, which are widely used in the multi-label classification task (Prabhu and Varma 2014; Jain, Prabhu, and Varma 2016; Prabhu et al. 2018; Bhatia et al. 2015). Precision@k ($P@k$) counts the fraction of correct label predictions in the top k scoring labels from the predicted labels list. Another evaluation metric is normalized Discounted Cumulative Gain at k ($nDCG@k$), a measure of ranking quality. We will present results when k = 1, 3, 5 following prior MTC works.

Experimental Results and Ablation Studies

Table 2 summarizes the performance of relevant methods together with our proposed model in terms of $P@k$ and $nDCG@k$. Bold numbers represent the best results and underlined numbers are the results second to the best ones. Our model Attentive-GCN + Adaptive-NLSOA (A-GCN + A-NLSOA in Table 2) consistently outperforms baselines by a substantial margin, obtaining average 4.5%, 2.6% and 1.7% for $P@1,3,5$ respectively, and average 4.5%, 4.1% and 3.9% for $nDCG@1,3,5$ separately over the prior state

[5]https://www.fast.ai/2018/07/02/adam-weight-decay/

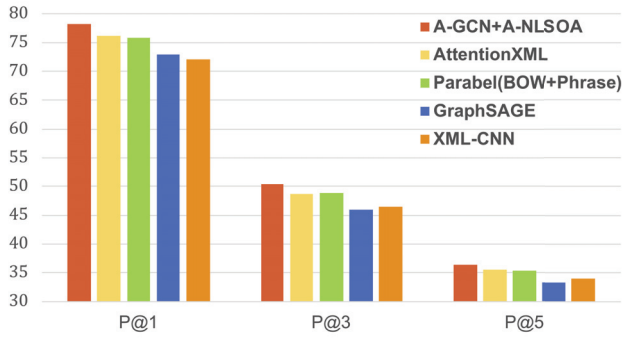

Figure 6: Precision@k performance

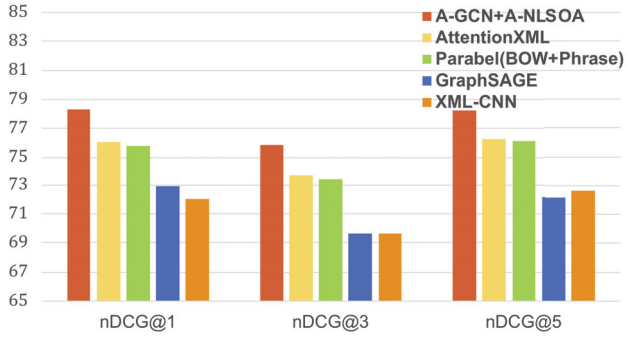

Figure 7: nDCG@k performance

of the art. We observe that rich features (BOW+Phrase) indeed attain benefits for tree-based approaches relying feature engineering. Parabel keeps its leading role among tree-based methods for almost entire metrics. It builds a label tree by incorporating text information, hence it performs better than other methods which fail to leverage label information during the training process. BiLSTM and XML-CNN present similar results, but BiLSTM shows subtle winning margin over XML-CNN because BiLSTM extracts useful semantic features in a bi-directional fashion. Compared to XML-CNN and BiLSTM, our model derives benefits from taking relations between documents and words and words co-occurrence information as well as document and label correlation into account, which harvests 6%, 4% and 3% improvements in $P@1,3,5$ and 6%, 5% and 6% with respect to $nDCG@1,3,5$ over those two methods. We develop GraphSage aggregators using 512 hidden units BiLSTM. Table shows our method obtains 5%, 5% and 3% improvements in precision and 5%, 6%, and 6% in nDCG over GraphSAGE. Our method also outperforms the competitive method AttentionXML with 2%. We pick the representative baseline methods from Table 2 and visualize the performance comparison results in Figure 6 and 7.

We conduct ablation studies to evaluate the effectiveness of each component in our A-GCN + A-NLSOA model and present the results in Table 3. Specifically, we first compare GraphSAGE with Attentive-GCN (A-GCN) to show the lack

Table 2: Results of Multi-Label Classification on Patents

Model	P@1	P@3	P@5	nDCG@1	nDCG@3	nDCG@5
FastXML(BOW)	73.10	47.84	34.74	73.10	71.66	74.02
FastXML(BOW+Phrase)	73.92	48.50	35.21	73.92	72.54	74.90
PfastreXML(BOW)	73.74	48.35	35.18	73.74	72.58	75.10
PfastreXML(BOW+Phrase)	74.42	48.93	35.52	74.42	73.31	75.76
Parabel(BOW)	75.10	48.23	35.01	75.10	72.92	75.37
Parabel(BOW+Phrase)	75.79	48.92	35.43	75.79	73.38	76.13
XML-CNN	72.06	46.48	33.94	72.06	69.68	72.63
BiLSTM	72.77	46.71	34.02	72.77	70.35	72.96
AttentionXML	76.09	48.71	35.47	76.09	73.65	76.26
GraphSAGE	72.95	45.88	33.40	72.95	69.67	72.19
A-GCN+A-NLSOA	**78.30**	**50.33**	**36.41**	**78.30**	**75.88**	**78.28**

Table 3: Ablation Study

Model	P@1	P@3	P@5	nDCG@1	nDCG@3	nDCG@5
A-GCN	75.74	48.29	34.92	75.74	73.01	75.42
A-GCN+LA	76.72	49.68	35.96	76.72	74.79	77.20
A-GCN+A-NLSOA	**78.30**	**50.33**	**36.41**	**78.30**	**75.88**	**78.28**

of differentiating word significance and inadequacy of extracting more complex textual features have big impacts on GraphSAGE's final performance. In patent MTC task, individual IPC label can be associated with diverse concepts at the same time. This property requires the model to capture the most relevant semantic parts of text for each IPC label. AttentionXML captures correlations between documents and labels in a direct manner using attention, and hence obtains superiority to competitor deep learning models. Our model follows the same direction, which directly explores the text and label relations using attention mechanism to advance predictive power. However, prediction ability of AttentionXML is bounded by its relatively weak textual representation and simple label attention. It adopts BiLSTM to learn text representation yet ignores more intricate textual features such as word co-occurrences and non-local words relationships. Thus we secondly show that our A-GCN module with AttentionXML's label attention ("A-GCN + LA") outperforms AttentionXML, which demonstrates the effectiveness of our GCN textual representation learning. Finally, we show our A-GCN + A-NLSOA presents 2% improvement than AttentionXML and 1.58% improvement than A-GCN + LA. It explains the effectiveness of modeling non-local and subtle semantics for patent classification, such design not only effectively captures information from semantically related words located non-locally in the document, but provides attended feature vectors as isolated label representations for each individual label. We claim that even though our work focuses on solving the patent text multi-label classification problem, the model we designed can be generalized to solve the generic multi-label text classification problems.

Sensitivity Analysis on Hyper-parameters

In this section, we evaluate the performance changing by varying the model hyper-parameters. Results are illustrated in Figure 8. For each attention head setting, we make 12 combinations by changing the number of sampled neighbors in the A-GCN module for search depth $k = 1, 2$. X-axis reflects the number of neighbors for $k = 1$. Number of heads and number of neighbors for $k = 2$ are separated by comma in the legend. It is observable that 20 attention heads obtains the best and stable performance for various neighbor combinations. Such observation is consistent with what is observed in (Vaswani et al. 2017). We only report parameter sensitivity for $P@1$. Results for $P@3, 5$ and $nDCG@1, 3, 5$ are omitted due to space limitation. Experiments show the optimal performance for entire metrics is achieved at 20 attention heads, 10 and 100 neighbors for $k = 1, 2$ respectively.

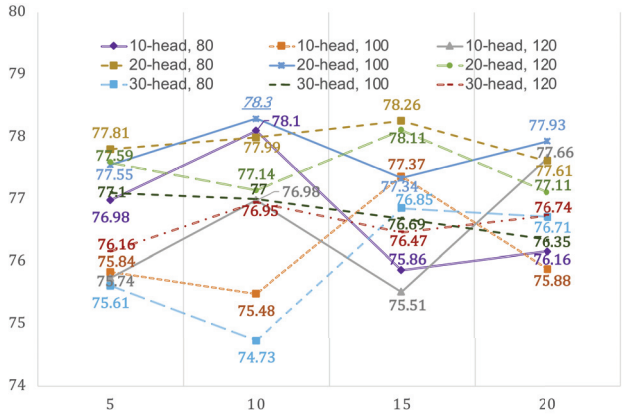

Figure 8: Parameter Sensitivity for $P@1$

Conclusions

In this paper, we present an attention-based GCN model which is characterized by Adaptive NLSOA layer. We apply it over a textual graph to solve a patent MTC problem. Model is designed to increase the representation power of the learned node embeddings in the context of long patent documents. Based on unique properties presented in the patent data, the Adaptive NLSOA not only manages to capture non-local and fine-grained semantic information, but also directly models the relationship between different semantic components in the document and individual label in order to alleviate the multi-meaning issue of IPC code. We conduct extensive experiments to evaluate the effectiveness of our model. Results demonstrate that our proposed model achieves very competitive performance, providing significant improvements over the current state-of-the-art.

Acknowledgements

The work is supported in part by the National Science Foundation (NSF) grant CNS-1629914, CHE-1925607 and IBM Research.

References

Babbar, R., and Schölkopf, B. 2017. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM)*, 721–729. ACM.

Bhatia, K.; Jain, H.; Kar, P.; Varma, M.; and Jain, P. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems (NeurIPS)*, 730–738.

Chen, M.; Lin, Z.; and Cho, K. 2017. Graph convolutional networks for classification with a structured label space. *arXiv preprint arXiv:1710.04908*.

Gomez, J. C., and Moens, M.-F. 2014. A survey of automated hierarchical classification of patents. In *Professional Search in the Modern World*. Springer. 215–249.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1024–1034.

Jain, H.; Prabhu, Y.; and Varma, M. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *KDD*, 935–944. ACM.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, 1746–1751.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR) 2017*.

Li, P.; Xie, J.; Wang, Q.; and Zuo, W. 2017. Is second-order information helpful for large-scale visual recognition? In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2070–2078.

Liu, J.; Chang, W.-C.; Wu, Y.; and Yang, Y. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 115–124. ACM.

Mahajan, D.; Girshick, R.; Ramanathan, V.; He, K.; Paluri, M.; Li, Y.; Bharambe, A.; and van der Maaten, L. 2018. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 181–196.

Nam, J.; Mencía, E. L.; Kim, H. J.; and Fürnkranz, J. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Advances in neural information processing systems (NeurIPS)*, 5413–5423.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. *NIPS 2017 Workshop*.

Prabhu, Y., and Varma, M. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, 263–272. ACM.

Prabhu, Y.; Kag, A.; Harsola, S.; Agrawal, R.; and Varma, M. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, 993–1002. International World Wide Web Conferences Steering Committee (WWW).

Rousseau, F.; Kiagias, E.; and Vazirgiannis, M. 2015. Text categorization as a graph classification problem. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL—IJCNLP) (Volume 1: Long Papers)*, 1702–1712.

Shang, J.; Liu, J.; Jiang, M.; Ren, X.; Voss, C. R.; and Han, J. 2018. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 30(10):1825–1837.

Tagami, Y. 2017. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *KDD*, 455–464. ACM.

Tang, J.; Qu, M.; and Mei, Q. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*, 1165–1174. ACM.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems (NeurIPS)*, 5998–6008.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. In *International Conference on Learning Representations (ICLR) 2018*.

Wang, G.; Li, C.; Wang, W.; Zhang, Y.; Shen, D.; Zhang, X.; Henao, R.; and Carin, L. 2018. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*.

Xia, B. N.; Gong, Y.; Zhang, Y.; and Poellabauer, C. 2019. Second-order non-local attention networks for person re-identification. In *The IEEE International Conference on Computer Vision (ICCV)*.

Yao, L.; Mao, C.; and Luo, Y. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7370–7377.

Yeh, C.-K.; Wu, W.-C.; Ko, W.-J.; and Wang, Y.-C. F. 2017. Learning deep latent space for multi-label classification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Yen, I. E.-H.; Huang, X.; Ravikumar, P.; Zhong, K.; and Dhillon, I. 2016. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning (ICML)*, 3069–3077.

Yen, I. E.; Huang, X.; Dai, W.; Ravikumar, P.; Dhillon, I.; and Xing, E. 2017. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *KDD*, 545–553. ACM.

You, R.; Dai, S.; Zhang, Z.; Mamitsuka, H.; and Zhu, S. 2018. Attentionxml: Extreme multi-label text classification with multi-label attention based recurrent neural networks. *arXiv preprint arXiv:1811.01727*.

Zhang, M.-L., and Zhang, K. 2010. Multi-label learning by exploiting label dependency. In *KDD*, 999–1008. ACM.

Zhang, W.; Yan, J.; Wang, X.; and Zha, H. 2018. Deep extreme multi-label learning. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval (ICMR)*, 100–107. ACM.