Regularized Graph Convolutional Networks for Short Text Classification

Kshitij Tayal
University of Minnesota
Twin Cities
tayal@umn.edu

Saurabh Agrawal
Amazon
Palo Alto, CA 94301
airan@amazon.com

Nikhil Rao
Amazon
Palo Alto, CA 94301
nikhilsr@amazon.com

Xiaowei Jia
University of Pittsburgh
Pennsylvania
xiaowei@pitt.edu

Karthik Subbian
Amazon
Palo Alto, CA 94301
ksubbian@amazon.com

Vipin Kumar
University of Minnesota
Twin Cities
kumar001@umn.edu

Abstract

Short text classification is a fundamental problem in natural language processing, social network analysis, and e-commerce. The lack of structure in short text sequences limits the success of popular NLP methods based on deep learning. Simpler methods that rely on bag-of-words representations tend to perform on par with complex deep learning methods. To tackle the limitations of textual features in short text, we propose a Graph-regularized Graph Convolution Network (GR-GCN), which augments graph convolution networks by incorporating label dependencies in the output space. Our model achieves state-of-the-art results on both proprietary and external datasets, outperforming several baseline methods by up to 6% . Furthermore, we show that compared to baseline methods, GR-GCN is more robust to noise in textual features.

1 Introduction

Short-text classification is a common problem in information retrieval (Ji et al., 2014) and has applications in several domains including e-commerce (Yu et al., 2012; Shen et al., 2009), social media (Kateb and Kalita, 2015), healthcare (Pestian et al., 2007) and cognitive-biometric recognition (Pokhriyal et al., 2016). In this paper, we develop a short text classification technique for solving two problems relevant to product search on e-commerce platform: 1) **Product Query Classification (PQC)** - When the customer enters a free form query, it is important to understand their product type intent to recommend and advertise the relevant products. We classify customer search queries to one or more product types (e.g., shoe, televisions, skis), and 2) **Product Title Classification (PTC)** - We classify billions of product titles to one or more product categories. This is important for sellers to place their items in the correct product category and retrieve it when a customer queries it.

Unlike traditional text classification, classifying short-texts poses additional challenges. First, short texts in e-commerce typically involve sentences with an average length of 3 (for queries) to 15 words (for product titles). Second, unlike longer texts such as blogs or news articles, these customer queries or product titles lack "natural" language structure and are often plagued with spelling errors. For example, in PQC, queries like `Nike running`, `shoes size 9`, `nike shos` (misspelling variants) all belong to the `shoes` category. In addition, queries contain non-target language text and non-language text (like model/part numbers), which introduces noise in the embedding. A similar challenge could also be faced in PTC problem, where products from the same class have high diversity in their title texts. For example, titles `PhotoFast microSD to MS Pro Duo CR-5300`, `Kingston microSD Card` and `8GB card for Blackberry Storm 9530` all belong to the same genre of `microSD` card products and hence need to be listed under the same category. All these factors make it difficult to separate product-type classes by purely relying on text which is heterogeneous and contains noise.

In this work, we propose to enhance the textual information by leveraging additional knowledge about relationships between input short-texts as well as among class labels. For PQC, we can derive similarities between input user queries from (anonymized) user logs, by looking at commonly purchased items in response to different queries. The intuition is that two queries that consistently lead to the purchase of a

same set of items might have similar product-type intents. Likewise, in PTC, we can estimate similarity between two input product titles from historical information such as co-views. Similarly, in output space, relationships between product-type classes can be modeled using product-category taxonomies, which are typically hand-curated and readily available in e-commerce applications.

Such auxiliary information can be naturally represented in graphical form, where each node represents a short-text (input graph) or a class label (output graph), while an edge indicates magnitude of similarity between two nodes. We thus propose a Graph-regularized Graph Convolution Network (GR-GCN) approach, which augments the graph convolutional network (Tayal et al., 2019) to incorporate such graphical information in an end to end learning framework. The two key aspects of GR-GCN are: i) a GCN that leverages dependencies in the input space to learn more informative representations of nodes(input short texts), and ii) a graph-regularization (GR) term in the objective function that exploits label similarities to penalize contrasting predictions for similar class labels on each input sample, thereby restricting the solution space and making our approach more robust to noise in the data.

We perform extensive experiments on one proprietary, and two public datasets and demonstrate the improvement in classification accuracy for GR-GCN upto 6% compared to text-based baselines. Further, we add noise in the input data and show that the graph's presence makes our method more robust to noise as compared to baseline methods based on just textual features.

2 Proposed Approach

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a matrix with each row being the embedding vector of an input sample and $\mathbf{Y} \in \mathbb{R}^{n \times L}$ be the label indicator matrix. Here d is the dimension of embedding and L number of class labels. Let $\mathcal{G}_\mathcal{I} = (\mathcal{V}_\mathcal{I}, \mathcal{E}_\mathcal{I})$ be the graph on input samples, with a corresponding adjacency matrix $A_\mathcal{I} \in \mathbb{R}^{n \times n}$, $\bar{A_\mathcal{I}} = A_\mathcal{I} + I$. Let $\mathcal{G}_o = (\mathcal{V}_o, \mathcal{E}_o)$ be the graph in output space, with $A_o \in \mathbb{R}^{L \times L}$ being the adjacency matrix on the output labels.

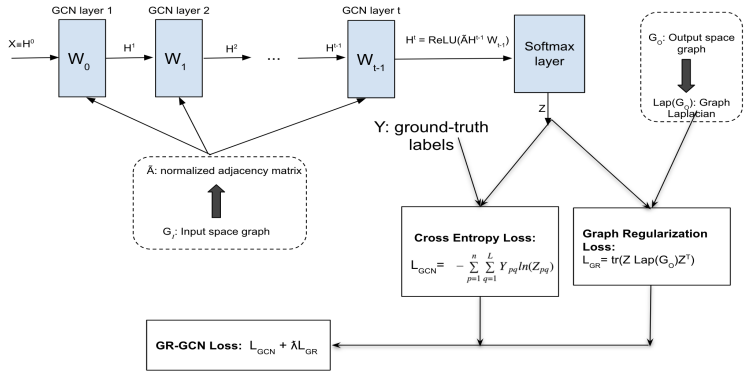

Figure 1: Schematic Illustration of GR-GCN. The GCN is used to learn node representations that respect the input graph structure, while the graph regularization is used to learn representations that respect the output graph structure.

Figure 1 illustrates out approach: GR-GCN. In this work, we use a 2-layer GCN (Kipf and Welling, 2016) on $G_\mathcal{I}$. The parameters are learnt by miniminizing the GR-GCN loss function: $L_{GCN} + L_{GR}$, where $L_{GCN} := \mathcal{L}(f_\theta(x), y)$ is the cross-entropy loss between the GCN predictions $f_\theta(x)$ for node x, parameterized by θ and the ground truth y. $L_{GR} := \sum_{i,j \in \mathcal{E}_o} \|f_\theta(x_i) - f_\theta(x_j)\|^2$ is the graph Laplacian based regularization that acts on the output node representations, and forces the predictions of adjacent nodes in the output graph to be similar. As we demonstrate in the experiments, this additional regularization makes our model especially robust to noise. Regularizers of the form L_{GR} have shown to be successful in factorization models (Johnson and Zhang, 2007; Rao et al., 2015; Zhou et al., 2012), and to the best of our knowledge, we are the first to apply it to regularize the output space for GCNs.

Table 1: Summary statistics of datasets

Dataset	#Docs	#Training	#Test	#Unique Words	#Edges	#Class	Average Length
Internal	200000	160000	40000	41880	3642910	2290	3.38
Electronics	188626	150,900	37,726	291,804	962,444	796	14.23
Home	279788	223,830	55,958	176,754	6549,740	1100	9.63

3 Graph Construction

In this section, we discuss the construction of graphs in input space and output space in the context of the two application problems that we focus on in this paper.

3.1 Product Query Classification (PQC):

The goal of PQC is to predict the product-type intent of user-typed search queries on an e-commerce website. To create the input graph, we use anonymous user logs that hold knowledge about query association. Intuitively, any two queries leading to the purchase of same items are more likely to have similar product-type intent. Following the intuition, we construct the graph such that for any two queries i and j, the adjacency matrix $A_{\mathcal{I}}$ is constructed as $A_{ij} =$ number of common purchases between query i and query j. To construct the output graph between product labels , we first represent each label (product category) with the mean of embeddings of the titles of products that belong to this category. We then apply cosine similarity between embedding vectors of labels to construct the output graph, and discard edges that do not meet the threshold.

3.2 Product Title Classification (PTC):

The goal of PTC is to classify products into product categories. Specifically, each input sample is the title of a product, while the output label is a product category. To construct graph $G_{\mathcal{I}}$, we use the co-viewed metadata of each product. Specifically, for two product titles i and j, the input matrix $A_{\mathcal{I}}$ is constructed as $A_{ij} =$ number of co-view between title i and title j. As with PQC problem, we used the same procedure to obtain the output space graph between labels. Co-views, is an intuitive means to construct the input graph, since items that are co-views are typically substitutes of each other. Thus, neighbors on the co-view graph tend to have similar categorization.

4 Experiments and Results

We evaluate GR-GCN approach on three datasets described in Table 1. For PQC, we used a proprietary e-commerce dataset (referred to as *Internal* dataset henceforth), whereas for PTC problem, we used a publicly available dataset consisting of Amazon product titles and other metadata including co-views for two departments: *Electronics* and *Home & Kitchen* (referred to as Home in this paper). (He and McAuley, 2016) [1].

Table 2: Classification accuracy of GR-GCN compared to multiple baselines

Model	Internal	Electronics	Home
TF-IDF + LR (Salton and Buckley, 1988)	80.9	59.70	61.2
CNN-rand (Kim, 2014)	79.45	55.87	61.42
CNN-non-static (Kim, 2014)	82.75	58.75	64.19
CharCNN (Zhang et al., 2015)	80.36	61.72	63.18
LSTM (Gers et al., 1999)	80.35	60.09	61.3
Bi-LSTM (Graves and Schmidhuber, 2005)	81.38	60.19	61.00
fastText (Joulin et al., 2016)	83.67	61.4	64.03
Graph-CNN-C (Defferrard et al., 2016)	80.08	58.60	59.25
Text GCN (Yao et al., 2019)	80.25	61.77	65.31
SWEM (Shen et al., 2018)	86.86	62.85	64.81
GR-GCN	**92.10**	**64.63**	**67.85**

All pre-trained word embeddings are 128-d learned by training FastText (Bojanowski et al., 2017; Joulin et al., 2016). For GR-GCN, we used a 2-layer GCN with learning rate 0.1, dropout set to 0.1 and

[1] http://jmcauley.ucsd.edu/data/amazon/links.html

L2 regularization factor λ of $1e^{-7}$. All the datasets are split into 70 % training, 10% validation, and 20 % testing.

4.1 Baselines :

The following are brief descriptions of the baselines in the comparative study. We have grouped our baselines into three categories i.e., Text (uses text features only), Graph (uses graph relation information only), and Text + Graph (which uses both textual and graph information).

Text Models: **TF-IDF+LR**: Bag-of-words model with TF-IDF as feature and Logistic Regression as a classifier. Low-frequency words appearing less than 5 times were removed. **CNN**: Two variants of CNN proposed in (Kim, 2014) are used: i) **CNN-rand** uses randomly initialized word embeddings and, ii) **CNN-non-static** uses pre-trained word embeddings. **CharCNN**: Character-level CNNs as proposed in (Zhang et al., 2015) **LSTM**: A simple LSTM block with 256 hidden states. We input pre-trained word embeddings. **Bi-LSTM**: Bidirectional LSTM block with 256 hidden states. We input pre-trained word embeddings. **fastText**: text classification tool from facebook. It averages words embedding, then feeds into a linear classifier. **SWEM**: employing average pooling operation (Shen et al., 2018) of feature and afterwards using feed forward network with architecture 256-512-1024-C as classifier, where C is the number of classes.

Graph Models: **Graph-CNN-C**: CNN model that performs convolutions across word embeddings relation graph using Chebyshev filter (Defferrard et al., 2016)

Text+Graph Models: **Text GCN**: GCN model where we construct corpus graph using documents and word as nodes (Yao et al., 2019).

4.2 Quantitative Results

For all baselines, we used default parameters as in their original paper/implementation. The results are summarized in Table 2. GR-GCN can be seen to outperform all baseline models in classification accuracy by a margin of 6% for the Internal dataset, 2.8% on Electronics and 3.8% on Home dataset. Further, we make the following observations from our results: **1)** A simple model, TF-IDF + LR performs well on short text datasets beating CNN random on internal and electronics datasets and performing comparably on home dataset. This reinforces our observation that short text documents often lack the structure that complex models such as neural networks can capture, and in some cases, using the latter might over parameterized the problem. **2)** LSTM based methods that use pre-trained embeddings also do not perform better than TFIDF based methods, for a similar reason: the word order is seldom important in user typed queries and product titles, and there's no "natural language" structure to exploit. **3)** TextGCN shows competitive performance on Electronics and Home dataset but performs poorly on the internal dataset. This is because the texts in the internal dataset are super short, with an average length below 4, contain many spelling errors, and the label space is vast. On further examination, we noticed that due to spelling errors, TextGCN is creating a lot of spurious nodes in the graph, which is a limitation of the learning graph from the corpus.

4.3 Impact of Incorporating Auxiliary Graphs

To evaluate the effect of each graph (input and output) individually on model performance, we perform two more experiments. In first experiment we incorporate output graph to the best "Text" performance model in our baseliens (SWEM) and measure the performance. We refer to this model as SWEM-GR-out. In the second experiment, we only use the input graph, thus eliminating the graph regularization. We refer to this model as GR-GCN-inp. From Table 3, we see that both input and output graphs in their individual capacity are adding meaning to the classification accuracy.

4.4 Robustness comparison

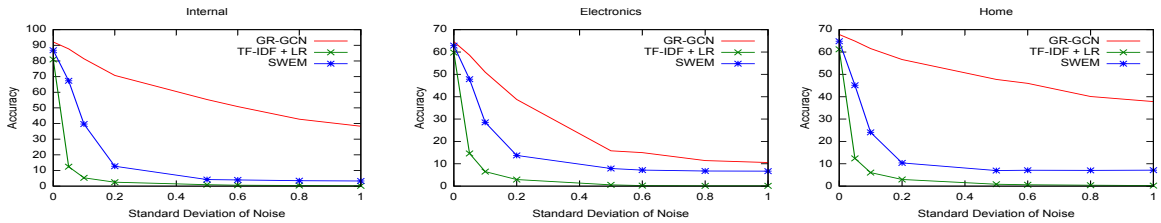

Figure 2: Test accuracy by varying standard deviation of added noise (best seen in color). GR-GCN consistently outperforms the baselines.

Model	Internal	Electronics	Home
SWEM	86.86	62.85	64.81
GR-GCN	92.10	64.63	67.85
SWEM-GR-*out*	86.90	63.26	65.20
GR-GCN-*inp*	91.95	63.99	66.9

Table 3: Performance Comparison

Search queries on e-commerce platforms contain a lot of misspelled keywords that introduce noise in the embedding. To simulate this behavior in our model, we evaluate the robustness of GR-GCN by introducing Additive White Gaussian Noise with zero mean and varying standard deviations in our embedding(Zhang and Yang, 2018). Figure 2 reports test accuracy with varying standard deviation (σ) of the noise. We compare GR-GCN with two text-base baselines: **TF-IDF+LR**, a simple bag of words model and **SWEM** (Shen et al., 2018), a state of the art deep learning model.

We see test performance of TF-IDF + LR, and SWEM drops immediately with minimal noise, while GR-GCN is robust to noise. On Internal dataset with $\sigma = 0.05$, TF-IDF performance drops from 80% to 12%, SWEM performance drops from 86% to 67%, while GR-GCN performance dropped from 91% to 87%. We attribute this noise-tolerant feature to the presence of input and output graph, which other methods lack.

4.4.1 Effect of the size of the Labelled Data

To assess the impact of the size of the labeled data, we tested the top-performing models with varying proportions of the training data. Figure 3 reports test accuracy with various sized subsamples of the training datasets. We observe that GR-GCN can achieve higher test accuracy with limited labeled documents.

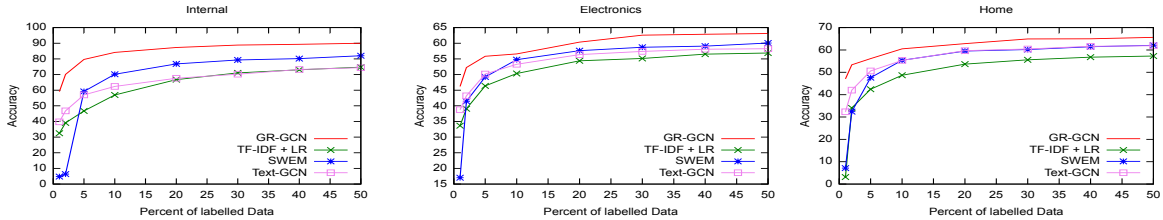

Figure 3: Test accuracy by varying training data (best seen in color).

5 Conclusion

In this paper, we propose GR-GCN to classify short texts that capture dependency on two levels, i.e., within the text samples (input space graph) and amongst the output label (output space graph). We demonstrated its efficacy on two commercial e-commerce applications and its robustness to noise.

We note that the proposed method can add value to other domains like medical science, where the input graph can capture drug similarity, and the output graph can capture the relationship among various types of illness; remote sensing, where the input graph can capture the distance, depth etc. between different ground points and output graph can capture similarity between similar labels such as pasture and vegetation as well as distinguish between entirely different labels such as river and residence.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Rie Johnson and Tong Zhang. 2007. On the effectiveness of laplacian normalization for graph semi-supervised learning. *Journal of Machine Learning Research*, 8(Jul):1489–1517.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Faris Kateb and Jugal Kalita. 2015. Classifying short text in social media: Twitter as case study. *International Journal of Computer Applications*, 111(9).

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

John P Pestian, Christopher Brew, Paweł Matykiewicz, Dj J Hovermale, Neil Johnson, K Bretonnel Cohen, and Włodzisław Duch. 2007. A shared task involving multi-label classification of clinical free text. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, pages 97–104. Association for Computational Linguistics.

Neeti Pokhriyal, Kshitij Tayal, Ifeoma Nwogu, and Venu Govindaraju. 2016. Cognitive-biometric recognition from language usage: A feasibility study. *IEEE Transactions on Information Forensics and Security*, 12(1):134–143.

Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in neural information processing systems*, pages 2107–2115.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Dou Shen, Ying Li, Xiao Li, and Dengyong Zhou. 2009. Product query classification. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 741–750. ACM.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *arXiv preprint arXiv:1805.09843*.

Kshitij Tayal, Rao Nikhil, Saurabh Agarwal, and Karthik Subbian. 2019. Short text classification using graph convolutional network. *NIPS workshop on Graph Representation Learning*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Hsiang-Fu Yu, Chia-Hua Ho, Prakash Arunachalam, Manas Somaiya, and Chih-Jen Lin. 2012. Product title classification versus text classification. *Csie. Ntu. Edu. Tw*, pages 1–25.

Dongxu Zhang and Zhichao Yang. 2018. Word embedding perturbation for sentence classification. *arXiv preprint arXiv:1804.08166*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. 2012. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *Proceedings of the 2012 SIAM international Conference on Data mining*, pages 403–414. SIAM.