Short Text Classification using Graph Convolutional Network

Kshitij Tayal[*]
Department of Computer Science
University of Minnesota, Twin Cities
tayal007@umn.edu

Nikhil Rao
Amazon
Palo Alto, CA 94301
nikhilrao86@gmail.com

Saurabh Agarwal
Amazon
Palo Alto, CA 94301
15.saurabh.16@gmail.com

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

Abstract

Short text classification is a fundamental problem in natural language processing, social network analysis, and e-commerce. Traditional approaches for classifying text do not generalize to short texts, due to the lack of structure that is prevalent in longer sentences and paragraphs. More recently, deep learning-based methods have been applied to this problem, with limited success. To overcome the limitations of textual features in short text, in this paper, we propose to apply Graph Convolutional Network (GCN) to classify short text. The method uses side-information present in the dataset to construct a graph and then learn a Short Text Graph Convolutional Networks (ST-GCN). We demonstrate the efficacy of our method in the context of two variants of the problem in the e-commerce domain where the text is short: product query classification and product title classification. The use of a GCN to represent our corpus allows us to capture dependencies between the text samples and this additional construction allows us to surpass several baseline methods. Our model achieves state-of-the-art results on both proprietary and external datasets, where ST-GCN outperforms other methods by up to 5.89 $\%$ in classification accuracy. Furthermore, we show that compared to baseline methods, ST-GCN is relatively more robust to noise in textual features.

1 Introduction

Short-text classification is a fundamental problem in e-commerce [1, 2], social media [3], bio-informatics, healthcare, and information retrieval. In social media platforms, such as Twitter, it is important to categorize tweets to hashtags and recommend them to users [4]. Similarly, in health-care applications, short notes written by medical practitioners are categorized for planning subsequent actions [5].

In this work, we solve two short-text classification problems relevant to product search:

- **Product Query Classification (PQC)**: classifying customer search queries to one or more product types (such as shoe, socks, shirt, etc.) This is critical for serving relevant results to the customers that results in higher conversion.

[*]Work was done while the author was at Amazon

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

- **Product Title Classification (PTC)**: classifying billions of product titles to one or more product categories. This is important for sellers to place their items in relevant categories. Consequently, it gets pulled up by the search engine during matching.

While the above problems fall under the umbrella of "text classification" [6], they pose additional challenges. First, these problems deal with very short texts with an average length from 3 words (for queries) to 15 words (for product titles), curtailing the information that can be obtained from text-based features. Second, unlike longer texts such as news articles or wiki pages, customer queries and product titles lack language structure and seldom follow grammar rules, thereby limiting the effectiveness of NLP-based techniques [7]. Furthermore, product-type classes may not have a compact representation in text feature space due to several factors. For instance, in PQC, customers have several ways to express the same buying intent. As an example, queries like `nike shoes`, `nike shoos` (misspelling variants), `running shoes`, `nike running`, `nike boomerang`, `black shoes walking`, and `shoes number size 9 soft sole broad base` all lead to the purchase of running shoes, and thus have similar product type intent. 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.

In this work, we overcome these difficulties of using sparse textual features by using relational input as side-information. This side information is obtained from customer behavioral data. For PQC, we use (anonymized) user logs to discover query similarities, by looking at commonly purchased items in response to different queries. The inspiration is that if two queries drive consistently the purchase of the same items, then the intent behind these queries is likely to be similar. Likewise, in PTC, the similarity between two product titles can be obtained via historical information such as co-views.

The similarity is best represented in form of a graph, where each node represents a short-text sample and the weight of an edge between two nodes indicate similarity score obtained from user behavioral data. Motivated by the graphical nature of the data, we propose to use Graph Convolutional Network (GCN) model to address the short-text classification as a node classification problem. We summarize the major contributions of the work below:

- We demonstrate the power of leveraging additional structural information between samples in the context of short-text classification problem.
- We perform extensive experiments on one proprietary, and two public datasets and show that our method outperforms several text classification baselines by 1.8 % - 5.8 %.
- We show that our approach is more robust to noise in the textual features, compared to the state-of-the-art methods that only rely only on text-based features.

1 algorithms. The success of these deep learning algorithms relies on their capacity to model complex and non-linear relationships within data [8]. Deep learning for text classification has two research direction. One research direction is to model better text representation that preserves word relationship while another research direction focuses on downstream model development.

2 Problem Formulation and Data Description

2.1 Problem Formulation

Assume we are given a dataset of n samples $\{x_i, y_i\}_{i=1}^n$. Each sample x_i is a short text, and $y_i \in \{1, 2, \cdots, L\}$ is a label assigned to the text. Furthermore, although we assume a multiclass classification setting, the method we propose can be trivially extended to the multi label setting: $y_i \subset \{1, 2, \cdots, L\}$ with a simple modification of the loss function. We assume we are given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with a corresponding adjacency matrix $A \in \mathbb{R}^{n \times n}$ on the samples. We abuse notation and denote by $x_i \in \mathbb{R}^d$ a vector representation of the short text. This representation can be obtained by, e.g. a bag-of-words or a more complex model such as a neural network. Let $X \in \mathbb{R}^{n \times d}$ be a matrix with each row being a sample x_i.

2.2 Data Description

2.2.1 Product Query Classification (PQC)

For this problem, we make use of a proprietary dataset from `Amazon.com` (which we will refer to as **Amazon Internal**). Details about the dataset are provided in Table 1. For each query, we obtain the FastText [9] embedding in $d = 128$ dimensions. We use anonymous user logs to construct the similarity graph between queries. Specifically, for two queries i and j, the adjacency matrix A is constructed as $A_{ij} =$ *# common purchases b/w query* i *and query* j. Note that the graph is symmetric.

2.2.2 Product Title Classification (PTC)

We make use of two publicly available datasets for this purpose, consisting of Amazon item titles and other side information [10] [2]. The datasets are broken down by categories, and we make use of **Electronics** and **Home** & **Kitchen**. Details are provided in Table 1. The dataset contains product titles, metadata for each product (also bought, also viewed, bought together, buy after viewing) and their categories. For each product, it's category is a path from a coarse-grained label to a fine-grained label (e.g.: `Electronics` ⇒ `Computers & Accessories` ⇒ `Cables & Accessories`). We use the product titles as inputs and the finest grained label from the above metadata as the item's label. We use FastText model to obtain the $d = 128$ dimensional feature representations of the item titles. To construct the graph between products, we use the 'also viewed' metadata of each product. The idea behind using co-views and not, for example, co-purchases, is that items viewed together tend to be substituted: a customer will view several similar items and then purchase one of them. Contrast this with co-purchases, which are complementary items: a customer will buy socks with shoes.

3 Short Text Graph Convolutional Network (ST-GCN)

Graph Convolution Network (GCN) [11] is a multilayer neural network that operates directly on a graph and induces node embedding based on properties of its connecting nodes. Formally, for a graph $\mathcal{G}$, let A be the adjacency matrix and let its degree matrix be D, where $D_{ii} = \sum_j A_{ij}$.

The graphs between queries (for PQC problem) and product titles (for PTC problem) are generated using the auxiliary information as described in Sections 2.2.1 and 2.2.2. After graph construction, we use a two-layer GCN and then apply following softmax layer $\sigma()$ and learn parameters optimizing cross entropy loss.

$$Z = \sigma(\tilde{A}ReLU(\tilde{A}XW_0)W_1) \tag{1}$$

where $\tilde{A} = D^{-1/2}\bar{A}D^{1/2}$ is the normalized symmetric adjacency matrix and W_0, W_1 are weight matrix, and $X \in \mathbb{R}^{n \times d}$ is the feature matrix.

4 Experiments and Results

Table 1: Summary statistics of datasets.

Dataset	No. of Text Samples	No. of Unique Words	No. of Edges in Graph G	No. of Classes	Average Length of text sample
Amazon Internal	200000	41880	3642910	2290	3.38
Electronics	188626	291,804	962,444	796	14.23
Home & Kitchen	279788	176,754	6549,740	1100	9.63

4.1 Quantitative Results

In this section, we compare ST-GCN with multiple text-classification baselines. Due to space constraints, we have provided descriptions of baselines and hyperparameter settings in supplementary

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

material. All pre-trained word embeddings are 128-d learned by training FastText [12, 13]. The results are summarized in Table 2. We use classification accuracy as evaluation metric.

ST-GCN significantly outperforms all baseline models on all 3 datasets which showcase the effectiveness of our approach for short text data. For the Amazon Internal dataset, where the graph is clean and the data contains a lot more labels we see a 5.86% relative improvement in accuracy compared to the second-best baseline model, SWEM. Similarly, for electronics and home & kitchen dataset we see a relative improvement of 1.8% and 2.4% from second-best baseline model respectively.

A simple model, TF-IDF + LR performs well on short text datasets, sometimes beating deep learning baselines such as CNN-random. This validates our hypothesis that short text documents often lack the structure that complex models such as neural networks capture, and sometimes the latter might lead to overfitting.

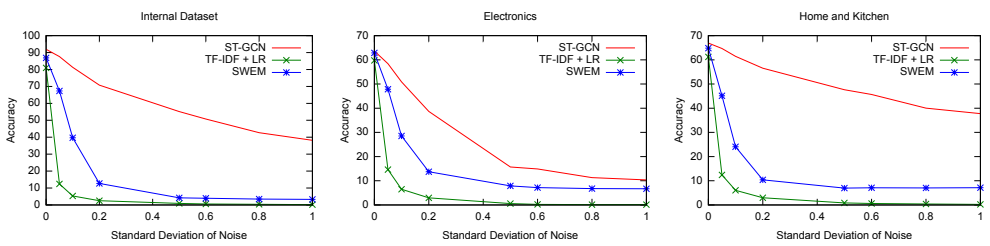
TextGCN, GCN based model for text classification, shows competitive performance on Electronics and Home & Kitchen dataset but performs very poorly on the Amazon Internal dataset. This could probably be due to customer queries being much shorter with an average length below 4 compared to product titles. On further examination, we noticed that due to spelling errors, TextGCN generates spurious edges in the graph, a limitation of learning the graph from the corpus.

Table 2: Summary of results in terms of classification accuracy (in percentage). Note that ST-GCN outperforms all baselines consistently.

Model	Amazon Internal	Electronics	Home & Kitchen
TF-IDF + LR [14]	80.9	59.70	61.2
CNN-rand [15]	79.45	55.87	61.42
CNN-non-static [15]	82.75	58.75	64.19
CharCNN [16]	80.36	61.72	63.18
fastText [13]	83.67	61.4	64.03
Graph-CNN-C [17]	80.08	58.60	59.25
Text GCN [18]	80.25	61.77	65.31
SWEM [19]	86.86	62.85	64.81
ST GCN	**91.95**	**63.99**	**66.9**

4.2 Robustness 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 ST-GCN to added noise. We tested best performance models by introducing additive white Gaussian noise with zero mean and varying standard deviations in our embedding[20]. Figure 1 reports test accuracy with varying standard deviation (σ) of the noise. We see test performance of TF-IDF + LR and SWEM drops immediately with very small noise, while ST-GCN is robust to noise. We note that Text GCN learn its own embedding via input corpus and consequently we were unable to introduce noise in their embeddings. On Amazon Internal dataset with $\sigma = 0.05$, TF-IDF performance drops from 80% to 12%, SWEM performance drops from 86% to 67%, while ST-GCN performance dropped from 91% to 87%.



(a) Internal Dataset (b) Electronics (c) Home and Kitchen

Figure 1: Test accuracy by varying standard deviation of noise (best seen in color)

.

5 Conclusion

In this paper, we propose ST-GCN, a GCN-based approach to solve short-text classification problem. It uses the side information present in the dataset to make a more informed decision of classify texts. We demonstrated its efficacy on two commercial applications with abundant short text i.e. product query classification and product title classification. Furthermore, we also show the robustness of ST-GCN to the noise. We note that this method is general and can be leverage in other domains wherever side information is available. A downside of our approach is that ST-GCN is transductive by design which could potentially be solved by introducing inductive representations [21] into the GCN framework and remains future work.

References

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

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

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

[4] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009, 2009.

[5] John P Pestian, Christopher Brew, Paweł Matykiewicz, Dj J Hovermale, Neil Johnson, K Bretonnel Cohen, and Włodzisław Duch. 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, 2007.

[6] Charu C Aggarwal and ChengXiang Zhai. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer, 2012.

[7] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842. ACM, 2010.

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[10] Ruining He and Julian McAuley. 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, 2016.

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

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

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

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

[15] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

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

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

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

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

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

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

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Supplementary Material

For ST-GCN, we set the embedding size of the first convolution layer as 128, learning rate 0.1, dropout as 0.1 and L2 regularization $1e-7$. We trained ST$-$GCN for a maximum of 200 epochs using Adam [22] and end training if the validation loss does not reduce for 15 continuous epochs. All the datasets were split into 70 % training, 10% validation and 20 % testing. For all baselines, we used default parameters as in their original paper/implementation.

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

- **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 [15] 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 [16]
- **fastText**: Pre-trained word embeddings as feature with linear classifier.
- **SWEM**: average pooling operation [19] using pre-trained embeddings as feature and feed forward network with layer architecture 256-512-1024-C as classifier, where C is the number of classes.

Graph only

- **Graph-CNN-C**: CNN model that performs convolutions across word embeddings relation graph using Chebyshev filter [17].

Text+Graph

- **Text GCN**: GCN model where we construct the input graph using documents and word as nodes [18] and use that for text classification.

For all baselines, we used default parameters as in their original paper/implementation. All pre-trained word embeddings are 128 dimensional vectors that were learned by training FastText embedding model [12, 13]. For PQC, fasttext model was trained on query log collected over one month on our search engine in United States. For PTC, fasttext model was trained on the product titles of external electronics and home & kitchen dataset respectively and no internal dataset was used for that purpose.