Simplified Graph Learning for Inductive Short Text Classification

Kaixin Zheng[*1,2], **Yaqing Wang**[*1], **Quanming Yao**[3], **Dejing Dou**[1]
[1]Baidu Research, Baidu Inc., China
[2] Department of SIOE, Beihang University, China
[3] Department of EE, Tsinghua University, China
zhengkaixin@buaa.edu.cn
{wangyaqing01, doudejing}@baidu.com
qyaoaa@tsinghua.edu.cn

Abstract

Short text classification (STC) is hard as short texts lack context information and labeled data is not enough. Graph neural networks obtain the state-of-the-art on STC since they can merge various auxiliary information via the message passing framework. However, existing works conduct transductive learning, which requires retraining to accommodate new samples and takes large memory. In this paper, we present SimpleSTC which handles inductive STC problem but only leverages words. We construct word graph from an external large corpus to compensate for the lack of semantic information, and learn text graph to handle the lack of labeled data. Results show that SimpleSTC obtains state-of-the-art performance with lower memory consumption and faster inference speed. [1]

1 Introduction

Short text classification (STC) is a fundamental task in many applications, such as sentiment analysis (Chen et al., 2019) and query intent classification (Wang et al., 2017). Due to the limited length, short texts lack context and strict syntactic structure, which makes them hard to understand (Wang et al., 2017). Hence, existing STC methods strive to enrich short texts by incorporating various auxiliary information, including concepts of external knowledge bases (Wang et al., 2017; Chen et al., 2019; Wang et al., 2022), topics discovered in the corpus (Zeng et al., 2018; Yang et al., 2021), part-of-speech (POS) tags (Wang et al., 2021), and entities residing in knowledge graphs (Hu et al., 2019). Another challenge faced by STC is the shortage of labeled data in practice (Pang and Lee, 2005; Phan et al., 2008). Hence, recent STC methods (Hu et al., 2019; Wang et al., 2021; Yang et al., 2021) further consider semi-supervised STC problem where only a small portion of the corpus is labeled.

The state-of-the-art STC models are based on graph neural networks (GNNs) (Defferrard et al., 2016; Kipf and Welling, 2016). HGAT (Hu et al., 2019) applies a GNN with dual-level attention on a corpus-level graph containing topics, entities and texts. SHINE (Wang et al., 2021) designs a hierarchical GNN to operate on a corpus-level hierarchical graph of word-level components including words, part-of-speech (POS) tags and entities, and a learned text graph. In summary, they conduct transductive learning, modeling the whole dataset (including the training and testing samples) as a large heterogeneous graph of various semantic concepts. Hence, they both suffer from two problems: (i) *require model retraining* if new samples come and (ii) have *high memory consumption* to store the complete dataset. The recent HGAT-inductive (Yang et al., 2021) proposes to connect each new sample to training data and unlabeled data existing in the corpus, which shows potential of solving STC problem by inductive learning.

In this paper, we propose a simple STC method called SimpleSTC to handle inductive STC problem. Inspired by the observation that semantic information contained in large datasets can be helpful to understand small datasets (Krizhevsky et al., 2012; Devlin et al., 2019), we use external large corpus (which will be called global pool to avoid confusion with the dataset corpus) to construct the basic word graph of common words. Upon the word graph, we represent short text embedding by pooling over words which appear in the short text, and dynamically learns the text graph which only connects related short texts. In a nutshell, the word graph constructed from the global pool compensates for the lack of semantic information, while the text graph handles the lack of labeled data. Although existing STC methods leverage

[*] Equal contribution. K. Zheng did his work during internship at Baidu Research. Correspondence to: Y. Wang.
[1]Codes are available at https://github.com/tata1661/SimpleSTC-EMNLP22.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 10717 - 10724

multiple auxiliary information, we find SimpleSTC with word graph constructed using global pool is enough to provide semantic information and obtain state-of-the-art performance on benchmark STC datasets, using much lower memory consumption. Incorporating auxiliary information such as entities or tags cannot further improve the performance.

2 Proposed Method

Given a training set $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{I}$ where $\mathbf{x}_i$ is a short text sample and y_i is its label, we aim at learning a predictive model to classify new samples. We present SimpleSTC (Figure 1) to solve this inductive STC problem. We first construct word graph using an external large corpus to compensate for the lack of semantic information. Then, we learn text graph by connecting short texts to words appearing in them, and propagate the limited labeled information among connected texts.

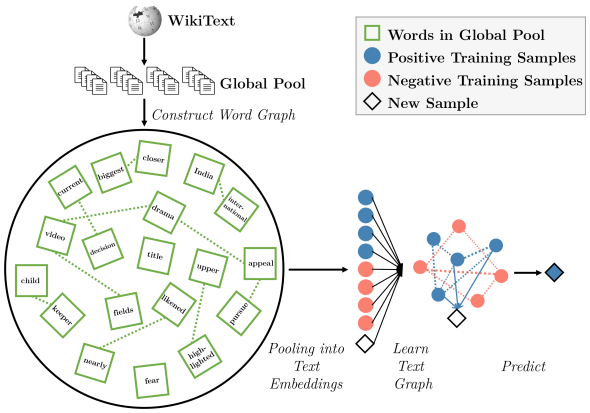


Figure 1: The architecture of the proposed SimpleSTC.

2.1 Word Graph Construction

To compensate for the lack of semantic information in the limited training data, we take an external large corpus as global pool which can provide abundant words and their statistics. Then, we construct word graph upon the global pool.

For global pool, we use WikiText (Merity et al., 2016), a famous large corpus containing millions of tokens extracted from Wikipedia articles. We only use abstracts, which summarize each article. We tokenize sentences, then remove stopping words and infrequent words which appear less than 10 times in the global pool. In this way, we wish global pool to retain more general and common information. The word set $\mathcal{V}_w$ contain the 24,867 words left in the global pool. We now can construct a word graph $\mathcal{G}_w$ which models the connection among these words. We connect words $v_m, v_n \in \mathcal{V}_w$ based on local co-occurrence statistics calculated by point-wise mutual information (PMI):

$$[\mathbf{C}]_{mn} = \max(\text{PMI}(v_m, v_n), 0). \quad (1)$$

We record node features in $\bar{\mathbf{E}} \in \mathbb{R}^{|\mathcal{V}_w| \times |\mathcal{V}_w|}$ where the ith row is the one-hot node feature $\bar{\mathbf{e}}_m \in \mathbb{R}^{|\mathcal{V}_w|}$ for $v_m \in \mathcal{V}_w$. This $\mathbf{C}$ then carries abundant semantic information discovered from the global pool.

2.2 Hierarchical Graph Learning

Here, we first obtain node embeddings in $\mathcal{G}_w$ by end-to-end training w.r.t each STC task to encode both general topology of $\mathcal{G}_w$ and dataset-specific information. We then leverage hierarchical graph learning to pooling over the word graph and dynamically learning a text graph to effectively propagate the limited label information.

Let $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}_w| \times d}$ denote node embeddings for words in $\mathcal{G}_w$, it is updated by 2-layer GNN as

$$\mathbf{E} = (\mathbf{C} + \mathbf{I})\text{ReLU}((\mathbf{C} + \mathbf{I})\bar{\mathbf{E}}\mathbf{W}_w^1)\mathbf{W}_w^2, \quad (2)$$

where $[\text{ReLU}(\mathbf{x})]_i = \max([\mathbf{x}]_i, 0)$, $\mathbf{C}$ is obtained by (1), $\mathbf{I}$ is identity matrix, and $\mathbf{W}_w^1, \mathbf{W}_w^2$ are parameters. We then concatenate $\mathbf{E}$ with pretrained word embeddings as $\hat{\mathbf{E}}$ following SHINE (Wang et al., 2021).

The short texts are encoded as the aggregated node embeddings in $\mathcal{G}_w$. Let $\mathcal{G}_s$ represent the text graph, where a node corresponds to a short text $\mathbf{x}_i$ in the training set. Each $\mathbf{x}_i \in \mathcal{G}_s$ is represented as

$$\bar{\mathbf{h}}_i = \hat{\mathbf{E}}^\top \mathbf{s}_i, \text{ with } [\mathbf{s}_i]_m = \text{TF-IDF}(v_m, \mathbf{x}_i), \quad (3)$$

where $(\cdot)^\top$ denotes the transpose operation, TF-IDF is the term frequency-inverse text frequency (Aggarwal and Zhai, 2012). Words in $\mathbf{x}_i$ but not in $\mathcal{V}_w$ are ignored.

Let $\mathbf{X}$ denote all short text embeddings with $\mathbf{x}_i$ on the ith row. We then estimate $\mathcal{G}_s$ as

$$[\mathbf{A}]_{ij} = \text{ReLU}(\cos(\bar{\mathbf{h}}_i, \bar{\mathbf{h}}_j) - \delta), \quad (4)$$

$$\mathbf{H} = \mathbf{A}\text{ReLU}(\mathbf{A}\mathbf{X}\mathbf{W}_s^1)\mathbf{W}_s^2, \quad (5)$$

where $\cos(\cdot,\cdot)$ is cosine similarity, $\delta \geq 0$ is a threshold to prune edges between irrelevant short texts, and $\mathbf{W}_s^1, \mathbf{W}_s^2$ are parameters.

Finally, we obtain class prediction of each $\mathbf{x}_i$ and optimize SimpleSTC w.r.t to classification loss:

$$\hat{\mathbf{y}}_i = \exp(\mathbf{w}_{y_i}^\top \mathbf{h}_i) / \sum\nolimits_c \exp(\mathbf{w}_c^\top \mathbf{h}_i), \quad (6)$$

$$\mathcal{L} = -\sum\nolimits_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{train}}} (\mathbf{y}_i)^\top \log(\hat{\mathbf{y}}_i),$$

where $\mathbf{y}_i \in \mathbb{R}^C$ is a one-hot vector with all 0s but a single 1 denoting the index of ground truth class $y_i \in \{1, \ldots, C\}$, and $\mathbf{w}_c$ denotes classifier parameter for class c.

During inference, all parameters of SimpleSTC are fixed. A new sample $\mathbf{x}_k$ is classified as follows: (i) tokenize $\mathbf{x}_k$ and obtain its feature $\bar{\mathbf{h}}_k$ by (3); (ii) estimate its relevance $[\mathbf{a}_k]_i$ w.r.t each $\mathbf{h}_i$ in $\mathcal{D}_{\text{train}}$ by (4); (iii) obtain short text embedding $\mathbf{h}_k = \mathbf{a}_k\mathbf{H} + \text{ReLU}(\bar{\mathbf{h}}_k\mathbf{W}_s^1)\mathbf{W}_s^2$ where the training short text embedding matrix $\mathbf{H}$ is directly used; (iv) make prediction by (6).

3 Experiments

We use a 24GB NVIDIA GTX 3090 GPU. All results are averaged over five runs. Some results are put in Appendix B due to space limit.

3.1 Datasets

Experiments are performed on benchmark short text datasets (Table 1).

	# texts	avg. len	# classes	# words
Twitter	9,970	6.6	2	20,726
MR	10,661	11.2	2	18,447
Snippets	10,174	17.5	8	25,906
TagMyNews	31,279	6.5	7	23,218

Table 1: Summary of short text datasets used.

Following Hu et al. (2019); Wang et al. (2021), we remove the duplicate text to avoid unfair testing, then tokenize each sentence and remove stopping words. Then, we separately sample 20 labeled samples per class as training and validation sets. The rest samples form the testing set, which are excluded during training.

3.2 Experimental Setup

Baselines. We compare our **SimpleSTC** with: (i) traditional two-step feature extraction and classification methods including **TF-IDF+SVM**, **LDA+SVM** (Cortes and Vapnik, 1995), and **WideMLP** (Galke and Scherp, 2022); (ii) pre-trained BERT (Devlin et al., 2019) which represents each short text as the averaged word embeddings (**BERT-AVG**) or the embedding of the CLS token (**BERT-CLS**) and is fine-tuned together with a linear classifier; (iii) inductive GNN based text classification methods including **TLGNN** (Huang et al., 2019), **TextING** (Zhang et al., 2020), and **HyperGAT** (Ding et al., 2020); and (iv) inductive STC method **HGAT-inductive** (Yang et al., 2021). Implementation details are in Appendix A.

Metric. We report micro-averaged accuracy (ACC) and macro-averaged F1 score (F1) averaged over five runs obtained on the testing sets.

Hyperparameter Setting. For methods which have been applied on datasets in Table 1, we use their reported hyperparameters. For other methods, we find hyperparameters by grid search. In SimpleSTC, the sliding window size is 5 when calculating PMI, the embedding size d is 200, and the threshold δ in (4) is set as 0.6. We use Adam (Kingma and Ba, 2014) with the learning rate as $1 * 10^{-3}$ to train the model for a maximum number of 1000 epochs. Dropout rate is 0.9. In ablation study, we consider adding entity graph and POS tag graph, which are constructed as in SHINE.

3.3 Performance Comparison

Table 2 shows the classification performance. As can be seen, SimpleSTC consistently obtains the state-of-the-art. Given a small training set, LDA cannot capture the topics well, thus performs even worse than TF-IDF. WideMLP fails to perform well given a small training set. BERT-AVG explicitly preserves information of each word, and obtains good performance on Snippets and TagMyNews where more training samples are used to fine-tune the model. HGAT separates 1000 unlabeled data from each dataset as data-specific pool, which fails to supplement the information loss. SimpleSTC outperforms the second-best (i.e., TextING) in Table 2 by 6.36% and 7.47% in terms of average ACC and F1 respectively. We attribute this to the exploitation of global pool which carries abundant semantic information. Results in Appendix B also show that SimpleSTC consistently outperforms TextING and HGAT with varying training data percentage.

For memory size, we compare SimpleSTC with (i) the state-of-the-art transductive STC methods SHINE and HGAT-transductive, and (ii) inductive STC methods HGAT-inductive. Results in Table 3 show that SimpleSTC takes the smallest memory space except on Snippets. When considering large dataset (i.e., TagMyNews), SimpleSTC works well, while the others run out of memory.

For inference time, we compare SimpleSTC with (i) the two inductive STC methods HGAT-inductive and SimpleSTC and (ii) the inductive text classification method TextING. Table 4 shows that Sim-

	Twitter		MR		Snippets		TagMyNews	
Model	ACC	F1	ACC	F1	ACC	F1	ACC	F1
TFIDF+SVM	$57.76_{(1.59)}$	$56.53_{(1.95)}$	$54.66_{(0.68)}$	$54.06_{(0.44)}$	$64.21_{(1.17)}$	$63.81_{(0.89)}$	$34.16_{(1.80)}$	$32.87_{(1.26)}$
LDA+SVM	$52.71_{(1.72)}$	$49.08_{(3.36)}$	$51.86_{(1.28)}$	$50.98_{(1.58)}$	$30.16_{(2.01)}$	$28.71_{(1.85)}$	$21.45_{(4.67)}$	$18.19_{(1.81)}$
WideMLP	$57.60_{(2.49)}$	$56.51_{(3.53)}$	$53.12_{(1.97)}$	$51.41_{(4.28)}$	$49.55_{(1.28)}$	$48.69_{(1.25)}$	$24.79_{(0.78)}$	$23.97_{(0.95)}$
BERT-AVG	$50.52_{(3.61)}$	$47.33_{(4.17)}$	$50.46_{(1.68)}$	$48.10_{(2.95)}$	$66.35_{(0.46)}$	$65.83_{(0.88)}$	$\underline{62.27}_{(1.61)}$	$\underline{56.91}_{(1.00)}$
BERT-CLS	$50.29_{(0.38)}$	$36.32_{(4.62)}$	$50.16_{(0.33)}$	$35.61_{(1.63)}$	$42.08_{(10.05)}$	$38.37_{(10.91)}$	$38.14_{(5.42)}$	$29.13_{(4.41)}$
TLGNN	$54.40_{(3.02)}$	$45.29_{(8.23)}$	$52.44_{(1.68)}$	$46.88_{(7.14)}$	$59.88_{(2.03)}$	$59.21_{(2.16)}$	$34.70_{(1.16)}$	$31.25_{(1.17)}$
TextING	$\underline{61.82}_{(2.19)}$	$\underline{60.77}_{(2.44)}$	$\underline{58.73}_{(1.02)}$	$\underline{58.30}_{(1.26)}$	$\underline{76.26}_{(1.20)}$	$\underline{75.70}_{(1.41)}$	$60.76_{(1.35)}$	$57.22_{(1.27)}$
HyperGAT	$56.12_{(4.81)}$	$49.92_{(11.67)}$	$51.59_{(0.35)}$	$44.81_{(4.23)}$	$34.91_{(0.81)}$	$34.80_{(0.85)}$	$24.43_{(4.39)}$	$17.77_{(3.00)}$
HGAT-inductive	$54.88_{(1.74)}$	$52.51_{(2.23)}$	$52.21_{(2.10)}$	$48.48_{(7.11)}$	$62.56_{(1.33)}$	$61.98_{(1.36)}$	OOM	OOM
SimpleSTC	$\mathbf{62.19}_{(1.56)}$	$\mathbf{62.01}_{(1.59)}$	$\mathbf{62.27}_{(1.11)}$	$\mathbf{62.14}_{(1.12)}$	$\mathbf{80.96}_{(1.69)}$	$\mathbf{80.56}_{(2.01)}$	$\mathbf{67.17}_{(1.27)}$	$\mathbf{63.34}_{(1.38)}$

Table 2: Test performance (%) obtained on benchmark datasets. The best results are marked in bold, and the second-best results are underlined. OOM indicates out-of-memory.

	Twitter	MR	Snippets	TagMyNews
SHINE	11.47	10.88	6.99	OOM
HGAT-transductive	18.88	14.14	OOM	OOM
HGAT-inductive	12.38	15.33	21.69	OOM
SimpleSTC	9.10	9.20	9.15	12.37

Table 3: Memory consumption (GB) of STC methods.

pleSTC is much faster than the others, which is more efficient in real deployment.

	Twitter	MR	Snippets	TagMyNews
TextING	2.88e-4	3.64e-4	2.58e-4	1.60e-4
HGAT-inductive	6.41e-6	7.33e-6	1.59e-5	OOM
SimpleSTC	3.53e-6	3.41e-6	3.64e-6	1.80e-6

Table 4: Inference time (seconds) per sample taken by inductive STC methods.

3.4 Ablation Study

Recall that the proposed SimpleSTC models word graph and then learns text graph. While the state-of-the-art SHINE constructs word graph, POS tag graph and entity graph as word-level component graphs, aggregates their node embeddings into short text embeddings and then learns text graph. Here, we consider multiple variants of SimpleSTC: (i) **SimpleSTC+$\mathcal{G}_\mathbf{p}$** adds POS tag graph $\mathcal{G}_p$ constructed from the 45 POS tags in global pool, **SimpleSTC+$\mathcal{G}_\mathbf{e}$** adds entity graph $\mathcal{G}_e$ constructed from 12,857 entities which appear more than 3 times in global pool, and **SimpleSTC+$\mathcal{G}_\mathbf{p}$+$\mathcal{G}_\mathbf{e}$** adds both $\mathcal{G}_p$ and $\mathcal{G}_e$ in par with $\mathcal{G}_w$. (ii) **SimpleSTC-$\mathcal{G}_\mathbf{s}$** removes $\mathcal{G}_s$ and directly takes $\mathbf{x}_i$ in (3) as $\mathbf{h}_i$ to be predicted in (6). (iii) **FeatureOnly** removes the hierarchical graph networks: It takes the input node features $\mathbf{X}_w$ as $\mathbf{H}_w$, obtains short text embedding $\mathbf{x}_i$ by (3) and then estimates $\mathbf{A}_s$ by (4), then directly uses label propagation (Zhou et al., 2004) to obtain class prediction.

Figure 2 shows the results. The following observations can be made: (i) When constructing word-level component graphs from a large global pool, SimpleSTC with $\mathcal{G}_w$ is enough to obtain satisfactory results, it does not need to use other auxiliary information captured in $\mathcal{G}_p$ and $\mathcal{G}_e$ like other STC models; (ii) The learning of $\mathcal{G}_s$ is necessary to obtain good performance, which is also observed in the transductive SHINE; (iii) Given the same amount of input, SimpleSTC outperforms FeatureOnly which validates the contribution of the hierarchical GNN designed in SimpleSTC.

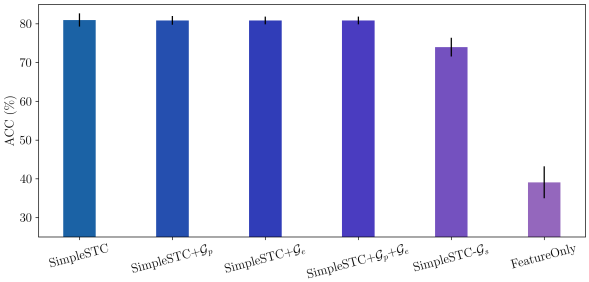


Figure 2: Ablation study on Snippets.

3.5 Study of Global Pool

In SimpleSTC, we use an external large corpus as the global pool to construct the work graph. This global pool is believed to contain abundant semantic information to represent short text datasets.

Table 5 shows how many of the words in each dataset which can be covered by the word graph constructed from the global pool. As shown, the word graph does not contain every word. We think the good performance of using word graph can be attributed to its ability of capturing common se-

mantic information, which is more important than seeing every word.

	Twitter	MR	Snippets	TagMyNews
# Words	20,726	18,447	25,906	23,218
# Covered Words	5,711	9,134	10,038	11,901
Ratio (%)	27.6	49.5	38.8	51.3

Table 5: Summary of words in each dataset which are covered by the global pool.

Recall that both HGAT and SHINE construct their graphs based on the corpus statistics, which is data-specific. Here, we further separate some samples from the corpus as the data-specific pool to construct the word graph, and examine the effect of using global pool vs data-specific pool. In addition, we add both POS tag graph $\mathcal{G}_p$ and entity graph $\mathcal{G}_e$ used by SHINE into SimpleSTC to analyze their contribution in different pools. Figure 3 shows that the inclusion of $\mathcal{G}_p$ and $\mathcal{G}_e$ indeed helps improve the performance of using data-specific pool, while they do not help when a global pool is used. We suspect that the global pool already provides enough semantic information, which covers the information offered by entities and POS tags. Although using a larger data-specific pool can increase the performance, data-specific pool which contains 50% samples of the corpus still cannot reach the performance of using global pool. Besides, leaving such a large amount of samples as the pool is impractical. While in SimpleSTC, one only needs to construct the word graph from global pool once, then can freely use it in different corpora.

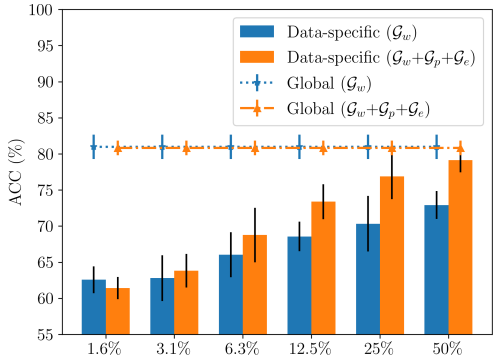


Figure 3: Varying $\frac{\text{pool size}}{\text{corpus size}}$ ratio (%) on Snippets.

3.6 Visualization of Learned Text Graph

Finally, we visualize the text graph $\mathcal{G}_s$ learned by SimpleSTC. As shown in Figure 4, SimpleSTC can learn a $\mathcal{G}_s$ which captures the correct relationship between samples, then the label information can be naturally forwarded to the new sample.

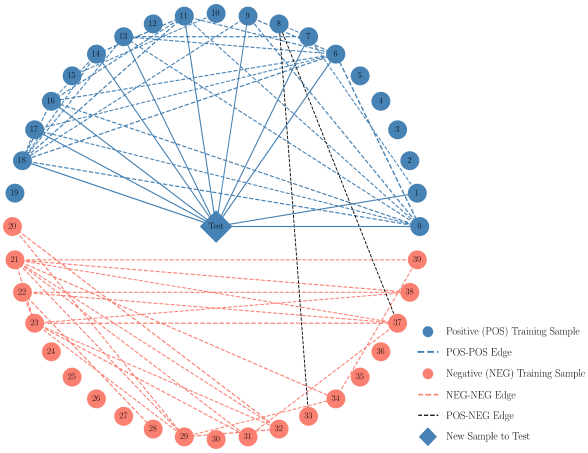


Figure 4: A text graph learned by SimpleSTC on MR.

4 Conclusion

STC problem is particularly hard due to the lack of context information. In the past, existing STC models leverage multiple auxiliary information such as words, topics and entities to provide semantic information. While in this paper, we find that using word graph of common words in the global pool alone can provide enough semantic information, incorporating other information cannot further improve the performance. Based on this insight, we propose SimpleSTC, which allows inductive learning, obtains state-of-the-art performance, and takes lower memory consumption and faster inference speed. We believe that we make a focused contribution to STC problem.

5 Limitations

We consider solving inductive STC problem on a hierarchically organized word graph and text graph. We manage to supplement semantic information by word graph constructed from an external large corpus. Nonetheless, how to dig out the most informative words instead of the common words from this external large corpus can be further explored.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (No. 2021ZD0110303).

References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining Text Data*, pages 163–222. Springer.

Jindong Chen, Yizhou Hu, Jingping Liu, Yanghua Xiao, and Haiyun Jiang. 2019. Deep short text classification with knowledge powered attention. In *AAAI Conference on Artificial Intelligence*, pages 6252–6259.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. 2020. Be more with less: Hypergraph attention networks for inductive text classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 4927–4936, Online. Association for Computational Linguistics.

Lukas Galke and Ansgar Scherp. 2022. Bag-of-words vs. graph vs. sequence in text classification: Questioning the necessity of text-graphs and the surprising strength of a wide MLP. In *Annual Meeting of the Association for Computational Linguistics*, pages 4038–4051, Dublin, Ireland. Association for Computational Linguistics.

Linmei Hu, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 4821–4830, Hong Kong, China. Association for Computational Linguistics.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2019. Text level graph neural network for text classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 3444–3450, Hong Kong, China. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Annual Meeting of the Association for Computational Linguistics*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

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

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *International Conference on World Wide Web*, pages 91–100.

Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining knowledge with deep convolutional neural networks for short text classification. In *International Joint Conference on Artificial Intelligence*, pages 2915–2921.

Yaqing Wang, Song Wang, Yanyan Li, and Dejing Dou. 2022. Recognizing medical search query intent by few-shot learning. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 502–512.

Yaqing Wang, Song Wang, Quanming Yao, and Dejing Dou. 2021. Hierarchical heterogeneous graph representation learning for short text classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 3091–3101, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tianchi Yang, Linmei Hu, Chuan Shi, Houye Ji, Xiaoli Li, and Liqiang Nie. 2021. HGAT: Heterogeneous graph attention networks for semi-supervised short text classification. *ACM Transactions on Information Systems*, 39(3):1–29.

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

Jichuan Zeng, Jing Li, Yan Song, Cuiyun Gao, Michael R. Lyu, and Irwin King. 2018. Topic memory networks for short text classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 3120–3131, Brussels, Belgium. Association for Computational Linguistics.

Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020. Every document owns its structure: Inductive text classification via graph neural networks. In *Annual Meeting of the Association for Computational Linguistics*, pages 334–339, Online. Association for Computational Linguistics.

Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328.

A Implementation Details

We provide the URLs to download the datasets and codes of baselines.

A.1 Datasets

We use the following benchmark datasets: (i) **Twitter**[2] is a collection of tweets; (ii) **MR**[3] is a collection of movie reviews for sentiment analysis (Pang and Lee, 2005); (iii) **Snippets**[4] is a collection of web search snippets of Google Search (Phan et al., 2008); and (iv) **TagMyNews** is a collection of English news titles.

A.2 Baselines

We compare the proposed **SimpleSTC** with:

- Traditional two-step feature extraction and classification methods including **TF-IDF+SVM**, **LDA+SVM** (Cortes and Vapnik, 1995) which use support vector machine (SVM) to classify texts with TF-IDF feature and LDA feature respectively, and **WideMLP**[5] (Galke and Scherp, 2022) which applies MLP one pretrained word embeddings. We obtain TF-IDF or LDA features based on training set rather than the whole corpus.
- Pretrained BERT[6] (Devlin et al., 2019) is fine-tuned by 5 epochs to the short text classification task. Each text is represented as the averaged word embeddings (denote as **-AVG**) or the embedding of the CLS token (denote as **-CLS**).
- Inductive GNN based text classification methods including **TLGNN**[7] (Huang et al., 2019) which operates on text-level word graphs where node embeddings and edge weights are shared for words appearing in the training set, **TextING**[8] (Zhang et al., 2020) which constructs text-specific graphs, while **HyperGAT**[9] (Ding et al., 2020) which construct hypergraphs using training set.
- Inductive STC method **HGAT-inductive**[10] (Yang et al., 2021) which discovers topics from training set, learns a GNN with dual-level attention from a heterogeneous graph of entities, topics and texts, then connects new samples to training samples.

For methods which require pretrained word embeddings, we use the 300-dimensional GloVe6B[11] (Pennington et al., 2014).

B More Experimental Results

Figures Reporting F1. In the main text, we report performance in ACC (%) in Figure 2 and Figure 3. Here, we provide the corresponding Figure 5 and Figure 6 which take F1 (%) as metric. As can be seen, the observation still holds. SimpleSTC with $\mathcal{G}_w$ constructed from the global pool outperforms the others consistently.

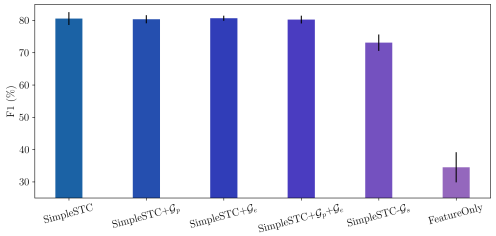


Figure 5: Ablation study on Snippets, reporting F1 (%).

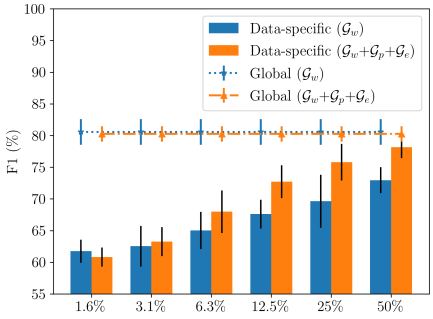


Figure 6: F1 (%) vs $\frac{\text{pool size}}{\text{corpus size}}$ ratio (%) on Snippets.

[2] http://www.nltk.org/howto/twitter.html#corpus_reader

[3] http://www.cs.cornell.edu/people/pabo/movie-review-data/

[4] Snippets and TagMyNews are downloaded from http://acube.di.unipi.it:80/tmn-dataset/.

[5] https://github.com/lgalke/text-clf-baselines

[6] https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4

[7] https://github.com/LindgeW/TextLevelGNN

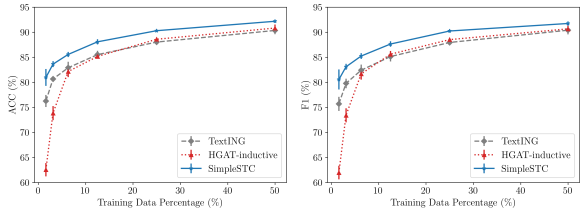
[8] https://github.com/CRIPAC-DIG/TextING

[9] https://github.com/kaize0409/HyperGAT

[10] https://github.com/ytc272098215/HGAT

[11] http://nlp.stanford.edu/data/glove.6B.zip

Effect of Training Data Percentage. We evaluate the effect of varying training data percentage in SimpleSTC, HGAT-inductive and TextING. As shown in Figure 7, SimpleSTC consistently outperforms the others. We attribute this to the exploitation of global pool which carries abundant semantic information.



(a) ACC. (b) F1.

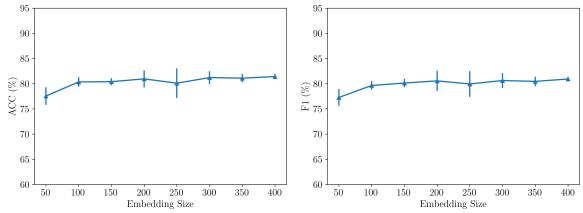
Figure 7: Varying training data percentage on Snippets.

Effect of Global Pool Size. In the main text, we keep common words which appear at least 10 times in the global pool. Here, we consider varying the size of global pool by using different word frequency. Table 6 shows that larger pool size leads to performance gain at the cost of larger memory space. One can balance the space and performance according to the needs.

Word Frequency	# word	memory	ACC	F1
>25	13343	5.08GB	$79.55_{(1.25)}$	$78.35_{(1.39)}$
>15	18997	7.03GB	$80.83_{(1.13)}$	$80.37_{(1.23)}$
>10	24867	9.08GB	$80.96_{(1.69)}$	$80.56_{(2.01)}$

Table 6: Varying the size of global pool on Snippets.

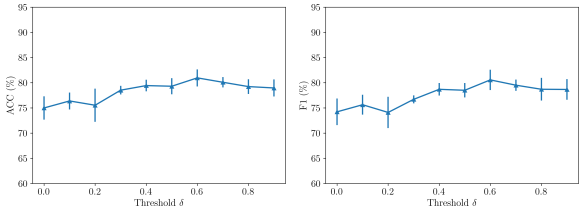
Effect of Embedding Size. Figure 8 plots testing performance vs embedding size. As shown, when embedding size is equal to 200 or 300, the performance is the best. Hence, we set embedding size as 200 to reduce parameter size.



(a) ACC. (b) F1.

Figure 8: Varying embedding size on Snippets.

Effect of Threshold δ in (4). Figure 9 examines the effect of δ in (4). Note that $\mathcal{G}_s$ is fully connected when $\delta = 0$, and $\mathcal{G}_s$ is removed when $\delta = 1$. As shown, $\delta = 0.6$ obtains the best performance, which validates the necessity of learning $\mathcal{G}_s$. Recall that we sample 20 texts per class, after pruning by δ, the average number of edges per text node is summarized in Table 7 below.



(a) ACC. (b) F1.

Figure 9: Varying δ in (4) on Snippets.

	Twitter	MR	Snippets	TagMyNews
Training Graph	9.5	8.9	14.7	12.0
Testing Graph	6.2	12.5	17.0	14.9

Table 7: Average number of edges per text node after pruning by δ.