Document Classification with Hierarchical Graph Neural Networks

Adrien Guille[0000−0002−1274−6040] and Hugo Attali

Université de Lyon, Lyon 2, ERIC UR3083
{adrien.guille,hugo.attali}@univ-lyon2.fr

Abstract. Various neural architectures have been explored for document classification, such as convolutional and recurrent networks or as of late, Transformers. In parallel, graph neural networks have vastly improved over the recent years. In this paper, we present preliminary results obtain with a novel, parameter-efficient, graph neural network that operates on documents encoded as directed graphs. These graphs describe document-wise word co-occurrence as well as the composition of sentences and thus make the neural network learn word, sentence and document representations in a hierarchical manner. Experiments conducted on various datasets show that it outperforms recent CNNs, RNNs and GNNs designed for document classification.

Keywords: Document Classification · Graph Neural Networks

1 Introduction

Document classification has many applications (*e.g.* automatic news categorization, automatic processing of medical documents, filtering of online Q&A threads). Several neural architectures have been explored to solve this task: convolutional neural networks [12][17], reccurrent neural networks [7][1] and lately, Transformers [25]. Pre-trained language models based on the Transformer [9] can be fine-tuned on downstream tasks such as document classification and generally yield state-of-the-art results. However, having hundreds of millions of parameters make them costly to operate or even incompatible with specific scenarios. Even though compressed language models can reduce the number of parameters to tens of millions, such as DistilBERT [22], their adoption remains limited for document classification, due to their inherent economic and environmental costs [24]. In parallel, graph neural networks have significantly improved and have become very efficient at processing graph-structured data [14][26] [5] [4]. In this paper, we build upon recent advances in the field of graph neural networks to design a novel parameter-efficient, architecture for document classification.

Proposal. Assuming that the exact ordering of the words isn't important but that capturing (i) in which sentence(s) they occur and (ii) how they co-occur together is essential, we encode documents as graphs. More precisely, we model a document as a directed graph made of one document vertex, connected to sentence vertices, themselves connected to word vertices (one vertex per unique word type) that are connected together to reflect document-wise co-occurrences. To solve the document classification task in terms of these graphs, we study a 4-layer neural network, based on the single-head graph attention mechanism (*i.e.* GAT) [26], that learns word, sentence and document embeddings in a hierarchical manner. In addition to this hierarchical behavior, the originality of the approach we investigate in this paper lies in that it explicitly models the document as a node in the graph, whereas existing GNNs for document classification such as MPAD [20] or TextING [30] require an additional readout function to aggregate all the embeddings to a graph-level representation for the document. Also, to mitigate the issue of over-smoothing that typically arises when stacking several GAT layers, we augment it with two kinds of residual connections: a residual connection on attention weights and a residual connection on embeddings. Lastly, we introduce a specific design for the last GAT layer, so that its lends itself to a potential interpretation, mainly by applying a mask on the last residual connection on attention weights.

Reproducibility. We open-source our code[1]. What is more, the datasets used in ours experiments and the code for all baselines are freely available online, making our results entirely reproducible.

2 Related Work

Convolutional Neural Networks. In a seminal paper, Kim proposes to perform 1D convolutions over sequences of word embeddings followed by global max pooling to solve document classification in terms of a CNN [12]. XML-CNN replaces the max-over-time pooling with a dynamic max pooling scheme in order to capture localized information in the documents [17]. Since this leads to much larger feature maps, it also includes an additional dense layer, a so-called bottleneck layer, to compress it before classification.

Recurrent Neural Networks. Even though efficient, CNNs are limited in the sense that convolution only captures short distance relations in the text. Recurrent neural networks based on the LSTM or GRU [7] cells can capture longer distance relations by maintaining and propagating hidden-states along the entire sequence of word embeddings. However, even though both GRU and LSTM cells address the exploding/vanishing gradient issue induced by the recurring calculations, their gating mechanism can still saturate, which limits their ability to classify long documents for instance. Yang *et al.* propose the HAN approach, that processes text in a hierarchical manner, encoding each sentence with a GRU and then passing the sentence embeddings to another GRU to learn a document embedding for classification [27]. Another way to improve RNNs for document classification is careful regularization. Adhikari *et al.* design the Reg-LSTM [1], a weight-dropped [19] bi-directionnal LSTM taylored for document classification, that incorporates an embedding dropout mechanism.

[1] Code available from: `https://github.com/AdrienGuille/DocGAT`

Transformers. To cope with the saturation caused by the recurring architecture and also improve the computational efficiency on modern hardware, Vaswani *et al.* introduce the Transformer, a deep, attention-based, feed-forward network [25]. This has led to the development of pre-trained language models such as BERT [9], with hundreds of millions of parameters , followed by compressed versions with tens of millions parameters like DistilBERT [22]. They can be fine-tuned on downstream tasks such as document classification and generally perform very well. Even though they obtain state-of-the-art results on benchmarks, relying on them in practical applications can be costly both from an economic and environmental standpoint [24] (*e.g.* costs induced by computation time on TPUs/GPUs in the cloud, pollution induced by the important energy consumption).

Graph Neural Networks. In parallel, much progress has been made in the field of graph neural networks. Convolution over graph-structured data can be efficiently approximated with Chebyshev polynomials in the spectral domain (*i.e.* the Laplacian of the graph) [8]. In practice, we tend to favor the GCN, a simpler, linear approximation of convolution [14]. While the convolution weights over the edges are fixed according to the Laplacian in the GCN approach, the GAT approach suggests to learn them via an attention mechanism [26]. Yao *et al.* are among the firsts to propose a GNN-based method for document classification, named TextGCN [28]. It consists in representing the entire corpus (*i.e.* the documents for training/validation and the documents to classify) as a single graph, with both document and word vertices, so that words are connected to the documents they occur in and words are connected together according to co-occurrence. Then, document classification is performed in a transductive fashion by applying a 2-layer GCN on this graph. This restricts the usefulness of TextGCN to very specific scenarios and inductive approaches have since been proposed. TextING [30] encodes each document as an undirected graph that describes how words co-occur within a window of size 3. Word embeddings are propagated through these graphs via a Gated Graph neural network [16] with a gating mechanism akin to the GRU cell, and a document embedding is obtained via a readout function that performs an additional gating and eventually a pooling based on two MLPs. MPAD [20] considers similar graphs, except that they're directed and include a so-called master vertex connected to all the word vertices. The propagation of the word embeddings is also more involved as it requires applying a GRU and a MLP iteratively to update all the embeddings. The readout function is implemented by a global attention mechanism similar to the Directional Self-Attention Network (DiSAN) [23], that considers all the intermediate word embeddings (*i.e.* the updated embeddings after each iteration).

In this paper, we intend to build upon recent advances in the field of graph neural networks to design a novel, inductive, parameter-efficient architecture for document classification.

3 Proposal

We begin by explaining how we encode documents as graphs. Then, we describe the components of the graph neural network we propose to learn word, sentence and document embeddings in a hierarchical manner. Finally we show how to estimate the network parameters for document classification.

3.1 Document Encoding

Consider a document made of n_s sentences and n_w distinct word types. Assuming that preserving the exact word order isn't necessary but that word co-occurrence as well as the sentence composition should be preserved, we encode this document as a directed graph. There are $N = 1 + n_s + n_w$ vertices, namely one vertex for the document, one vertex for each sentence and one vertex for each unique word type. The document vertex (numbered 0) has outgoing edges towards all the sentence vertices (numbered 1 to n_s), while each sentence has outgoing edges towards all the words that occur in it. Words are connected together according to their document-wise co-occurrence within a symmetric window of size 1. This graph is characterized by its adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$. Fig. 1 illustrates the graph constructed on a toy document. We pair this graph with vertex features $\mathbf{X} \in \mathbb{N}^{N \times 1}$, integer ids that map them to their respective initial embedding in the neural network. The document vertex has a special id, 0, and all sentence vertices have the same special id, 1.

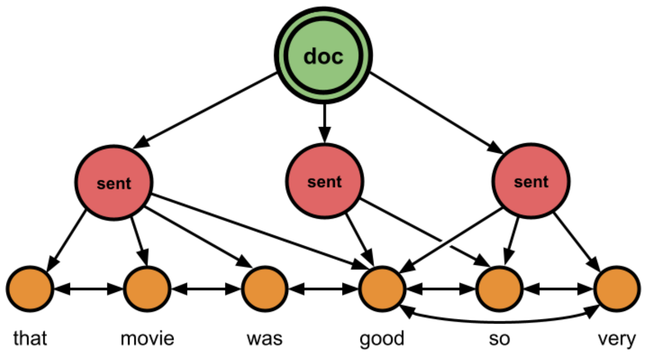

Fig. 1. Encoding of the document: "That movie was good. So good. So very good."

3.2 Model Architecture

The model operates on the two inputs we've just described, a directed adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$ and a feature matrix $\mathbf{X} \in \mathbb{N}^{N \times 1}$, and aims at learning embeddings for the words, the sentences and the document in a hierarchical manner. The architecture consists in an initial embedding layer followed by L single-head graph-attention layers (*i.e.* GAT layers) [26]. To mitigate the over-smoothing phenomenon that arises when stacking several GAT layers [6], we implement two kinds of residual connections: a residual connection on the attention weights and a residual connection on the embeddings. In the following we describe the general design of our GAT-like layers and conclude by describing the specific design of the last layer to ensure the output is straightforward to interpret.

Single-Head Graph Attention Layer. We consider the simplest form of attention, single-head attention [3], as we found no advantage in employing multi-head attention in any of our experiments. The l^{th} layer receives the adjacency matrix $\mathbf{A}$, with added self-loops on all vertices, and the input vertex embeddings $\mathbf{H}^{(l-1)}$; it outputs the updated vertex embeddings $\mathbf{H}^{(l)}$. When $l = 1$, the input $\mathbf{H}^{(0)}$ are the initial embeddings.

First, we compute the attention scores between vertices. If $a_{ij} = 0$, the attention score α_{ij} is set to 0, otherwise, it is computed as:

$$\alpha_{ij}^{(l)} = \frac{\exp\left(\text{LeakyReLU}(z^{(l)} \cdot [h_i^{(l-1)}\mathbf{W}^{(l)} || h_j^{(l-1)}\mathbf{W}^{(l)}])\right)}{\sum_{k\in\mathcal{N}_i} \exp\left(\text{LeakyReLU}(z^{(l)} \cdot [h_i^{(l-1)}\mathbf{W}^{(l)} || h_k^{(l-1)}\mathbf{W}^{(l)}])\right)}, \tag{1}$$

where $||$ stands for concatenation and $\mathcal{N}_i$ is the set of outgoing neighbors of the vertex i. This score is parameterized by $z^{(l)} \in \mathbb{R}^{2d_{\text{out}}}$ and $\mathbf{W}^{(l)} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, respectively a weight vector and a linear transformation. The LeakyReLU activation is calculated with a negative slope of 0.2 as in [26]. The candidate updated embeddings $\mathbf{H}'^{(l)} \in \mathbb{R}^{N \times d_{\text{out}}}$ are calculated according to the attention scores. For vertex i, it is calculated as:

$$h_i'^{(l)} = \sum_{j\in\mathcal{N}_i} \alpha_{ij}^{(l)} (h_j^{(l-1)}\mathbf{W}^{(l)}). \tag{2}$$

Residual Connection on Attention Weights. Following recent work by He *et al.* showing that residuals on the attention weights improves the stability of the Transformer [10], and work by Lv *et al.* showing it could also benefit attention-based GNNs [18], we replace Eq.2 with Eq. 3 below when $l > 1$ to compute the candidate output embeddings:

$$h_i'^{(l)} = \sum_{j\in\mathcal{N}_i} \Big(\frac{1}{2}\big(\alpha_{ij}^{(l)} + \alpha_{ij}^{(l-1)}\big) h_j^{(l-1)}\mathbf{W}^{(l)}\Big). \tag{3}$$

We take the arithmetic mean between the attention weights computed in this layer and the attention weights computed in the previous layer and mix the embeddings accordingly.

Residual Connection on Embeddings. We also implement a more conventional residual by summing the candidate output embeddings with the input embeddings, prior to activation, to obtain the actual output embeddings $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times d_{\text{out}}}$:

$$h_i^{(l)} = \sigma\big(h_i'^{(l)} + h_i^{(l-1)}\big), \tag{4}$$

where σ is the activation function, SELU [15] in our experiments. Note that this requires having $d_{\text{in}} = d_{\text{out}}$. In order to have different input and output dimensions, one could, for instance, linearly transform $\mathbf{H}^{(l-1)}$. We don't detail this modification as we found that keeping the same dimension in the $L-1$ first layers led to the best results (even though varying the dimension only seemed to have a minor effect).

Interpretable Last Layer. To make the predictions interpretable, we adopt a specific design for the last GAT layer. First, it operates on the adjacency matrix $\mathbf{A}$, without self-loops. Second, the attention weight matrix calculated in the penultimate layer is masked before applying the attention residuals, by setting its diagonal to 0. Third, it ignores the residual connection on the embeddings. By doing so, the document embedding computed in the last layer is purely a weighted average of the sentence embeddings, while the sentence embeddings are purely weighted averages of the word embeddings. Because, as detailed in the next subsection, the classification is a function of the sole embedding of the document vertex outputed by the last layer, the attention weights in the last layer allow to directly identify the important sentences and words.

3.3 Parameter Estimation

We extract the embedding for the document vertex calculated in the last layer, $h_0^{(L)}$, and pass it to a fully-connected layer with softmax activation for multiclass classification:

$$\hat{y} = \text{softmax}(h_0^{(L)}\mathbf{C} + b), \tag{5}$$

where $\mathbf{C} \in \mathbb{R}^{d \times c}$ and $b \in \mathbb{R}^{c}$ are the parameters of this linear classifier with c the number of classes. For the purpose of regularization we add Alpha-Dropout [15] with a rate of 0.5 before the classification layer, as well as Dropout in every GAT layer with a rate of 0.5. We learn all the parameters by minimizing the categorical cross-entropy via mini-batch stochastic gradient descent. For multilabel tasks, we replace the softmax activation with the sigmoid activation, and replace the categorical cross-entropy with the binary cross-entropy.

4 Experiments

In this section we describe the series of experiments we've conducted to thoroughly evaluate the performance of our method and demonstrate the usefulness of all of its components. Note that our experiments are entirely reproducible, as we provide links to retrieve all of the datasets as well as all of the source code needed.

4.1 Datasets and Protocol

We run experiments on 5 well-known multilabel and multiclass datasets – ranging from about 3,000 to 250,000 training documents – covering news articles, scientific abstracts, and online Q&A (see Tab. 1):

- **Reuters 90**[2]: A set of news stories from the Reuters agency. We keep the original train/test split [2]. The aim is to predict the category or categories of each story.
- **Reuters 8**[2]: A subsample of the above dataset, with only the 8 most frequent classes and only single label documents [20]. The aim is to predict the category.
- **Ohsumed**[3]: A set of medical abstracts from PubMed. We keep the original train/test split [11]. The aim is to predict the cardiovascular disease addressed in the papers.
- **AG News**[4]: A set of news headlines from more than 2000 news sources. We keep the original train/test split [29]. The aim is to predict the category.
- **Yahoo Q&A**[4]: A set of Yahoo! Questions & Answers from the Yahoo! Webscope program (question title, question content and best answer). We keep the original train/test split [29]. The aim is to predict the topic.

We randomly sample 10% out of the train set for validation, identical for all methods. We run each method 10 times on each dataset and report the average accuracy in the multiclass case or the average micro F1-score in the multilabel case.

Table 1. Dataset properties.

	Reuters 90	**Reuters 8**	**Ohsumed**	**AG News**	**Yahoo Q&A**
# doc (train)	7,770	5,584	3,357	120,000	250,000
# doc (test)	3,019	2,208	4,043	7,600	60,000
# labels	90	8	23	4	10
type	multilabel	multiclass	multiclass	multiclass	multiclass

4.2 Compared Methods

We compare our method with 2 CNNs, 2 RNNs and 2 GNNs designed for document classification:

[2] Retrieved from: `https://git.uwaterloo.ca/jimmylin/hedwig-data/`
[3] Retrieved from: `https://github.com/CRIPAC-DIG/TextING`
[4] Retrieved from: `http://goo.gl/JyCnZq`

- **Our method**: 4 layers, with $d_{\text{out}} = 300$ for the first 3 layers and $d_{\text{out}} = 300$ in the multilabel case or $d_{\text{out}} = 150$ for the last layer in the multiclass case (allowed by the absence of residual connection on the embeddings).
- **CNN**[5] [12]: filters of width 3, 4 and 5, with 100 filters for each width as in the original paper [12].
- **XML-CNN**[5] [17]: filters of width 2, 4 and 8, with 100 filters for each width and a dynamic pooling window length of 8 as in [1].
- **HAN**[5] [27]: bidirectionnal GRUs for the sentence and document encoders, 100 hidden units in total, as in the original paper [27].
- **Reg-LSTM**[5] [1]: LSTM following modern best practices for document classification, 512 hidden units, regularized via embedding dropout and weight dropping, with dropout rates of 0.1 and 0.2 respectively, as in the original paper [1].
- **TextING**[6] [30]: 2-layer network with co-occurrences within a window of size 3 as recommended in the original paper [30].
- **MPAD**[7] [20]: 2 message passing layers, each followed by a 2-layer MLP, that reduces the dimension to 64 as in the original paper [20].

We initialize the embedding layer of all the networks with the English GloVe embeddings [21] in dimension 300, trained on Common Crawl data[8]. For our method, we add two randomly initialized vectors corresponding to the initial sentence and document embeddings. All neural networks are trained end-to-end with the Adam variant of the mini-batch stochastic gradient descent [13], with an initial learning rate of 0.001.

4.3 Main Results

Table 2 reports the mean scores on all datasets. Our approach achieves the best scores on four out of five datasets and the second best score on the toughest dataset, Ohsumed. This shows that the proposed method can efficiently learn from datasets of various sizes, made of documents of different natures. In contrast, CNNs are competitive on smaller datasets mostly, while RNNs need larger datasets to reach interesting scores. Regarding GNNs, MPAD has a good overall performance even though it is always outperformed by the proposed method whereas results from TextING are more contrasted, with a good score on the toughest dataset but a rather modest performance otherwise.

4.4 Detailed Analysis

Ablation Study. First, we analyze the impact of the two kinds of residual connections on the performance. Fig. 2 shows that both the attention residuals and

[5] Code available from: `https://github.com/castorini/hedwig`
[6] Code available from: `https://github.com/CRIPAC-DIG/TextING`
[7] Code available from: `https://github.com/giannisnik/mpad`
[8] Retrieved from: `https://nlp.stanford.edu/data/glove.42B.300d.zip`

Table 2. Mean test scores over 10 runs (standard deviation in parentheses). Best scores in bold and second-best scores in italic.

	Reuters 90	**Reuters 8**	**Oshumed**	**AG News**	**Yahoo Q&A**
CNN	*85.1* (0.3)	96.9 (0.2)	**70.3** (0.3)	*92.7* (0.1)	71 (0.1)
XML-CNN	83.9 (0.3)	96.8 (0.2)	66.7 (0.4)	92.5 (0.2)	71.8 (0.1)
HAN	76.4 (0.4)	95 (0.5)	61.5 (3.2)	92.6 (0.1)	72.7 (0.1)
Reg-LSTM	80.1 (0.3)	*97.2* (0.4)	69.1 (0.8)	*92.7* (0.1)	*73* (0.1)
TextING	75.1 (0.9)	96.5 (0.3)	69.5 (0.7)	90.4 (0.2)	OOM (192GB)
MPAD	85.0 (0.4)	96.6 (0.5)	67.6 (1.8)	92.8 (0.2)	72.5 (0.2)
Our Method	**87.8** (0.3)	**97.4** (0.2)	*69.9* (0.3)	**93.2** (0.1)	**73.1** (0.1)

the embedding residuals are essential. Removing either of them harms the performance and removing both of them leads to notably worse scores. Second, we study the impact of the number of GAT layers. Overall, it appears that varying the number of layers by one doesn't have a great impact on the performance. On average, removing one GAT layer decreases the scores by 0.23 points and adding one more layer decreases the scores by 0.15 points. In some cases, adding or removing one layer leads to a better score, but the improvement doesn't exceed 0.1 on any of the five datasets. Thus, 4 layers is the best performing network size overall.

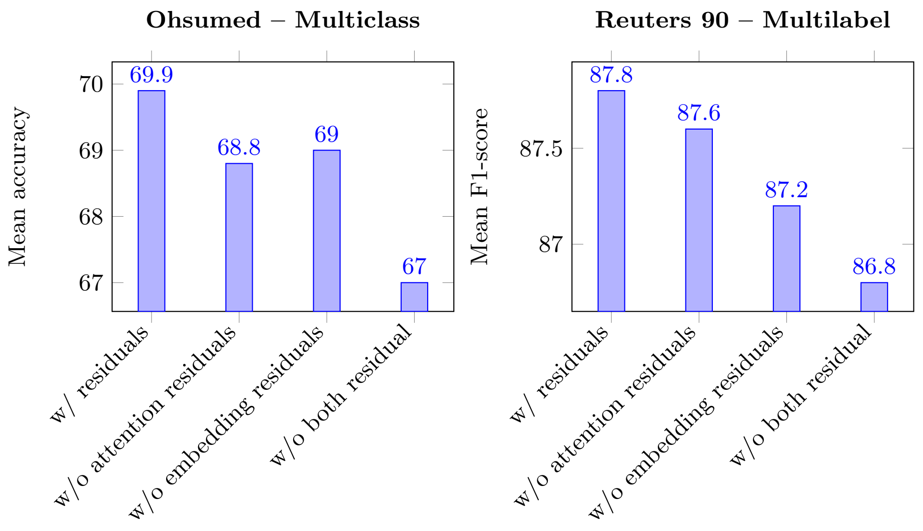

Fig. 2. Impact of the residual connections on the performance.

Parameter Efficiency. Not accounting for the embedding layer, because it is the same for all methods and it doesn't influence the comparison of the FLOPs implied by each method, our approach involves 315k parameters for an output dimension of 150 or 360k for an output dimension of 300, which means it has a model size comparable to the simplest CNN baseline. In contrast, XML-CNN (because of the larger feature maps engendered by dynamic pooling and the additional bottleneck layer) and Reg-LSTM (because of the involved gating mechanism) each have about 1,6M parameters. Combined with the fact that it is a strictly feed-forward network – as opposed to RNNs – this makes our approach computationally more affordable and easier to deploy in real-world scenarios.

5 Interpretability

Fig. 3 and 4 show documents taken from the test sample of the AG News dataset. In each document, the sentence receiving most attention from the last layer is underlined, and the two words receiving the most attention in them are double-underlined. The first example is mostly a quote, which, taken out of context, would be difficult to classify into the "Sports" category. Yet, the network focuses its attention on the introductory sentence, mostly attending the phrase "Athens Olympics", skipping the quote to predict the correct category. The second one is a longer document, related to business and start-ups selling shares for the first time on exchanges. The sentence that receives the most attention is actually key in understanding the article's topic, and allows the reader to quickly confirm that the prediction makes sense.

Notable quotes Tuesday at the Athens Olympics. "It hurt like hell. I could see (Thorpe) coming up. But when I was breathing, I saw my team going crazy – and that really kept me going."

Fig. 3. Predicted category: Sports.

Start-ups are discovering they might have to wait until the timing is right. Is the market for initial public offerings open or closed? Few questions loom larger for venture capital firms, which risk money on entrepreneurial companies and look for "liquidity events" that will help them recoup their investments. But more than at any other time in the recent past, the answer may depend on your vantage point.

Fig. 4. Predicted category: Business.

Though it seems that the attention weights in the last layer might quantify the importance of sentences and words, actual experiments are required to objectively assess this.

6 Perspectives

We've studied a simple but efficient hierarchical graph neural network that performs inductive classification on documents encoded as directed graphs, which preserve word co-occurrence and sentence composition. The promising experimental results motivate us to explore ways of improving that approach. First, we'd like to reformulate the neural network so that it operates in the hyperbolic space (instead of the euclidean space), which seems well suited to the hierarchical graphs that encode the documents. Second, we'd like to assess the ability of our approach to distillate knowledge from fine-tuned deep language models to improve performance while retaining the same low computational cost at prediction time. Third, we'd like to investigate how the attention weights in the last layer could contribute to the interpretability of the model.

References

1. Adhikari, A., Ram, A., Tang, R., Lin, J.: Rethinking complex neural network architectures for document classification. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics. pp. 4046–4051. NAACL (2019)
2. Apté, C., Damerau, F., Weiss, S.M.: Automated learning of decision rules for text categorization. ACM Trans. Inf. Syst. **12**(3), 233–251 (1994)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of the International Conference on Learning Representations. ICLR (2015)
4. Brochier, R., Guille, A., Velcin, J.: Global vectors for node representations. In: Proceedings of The World Wide Web Conference. pp. 2587–2593. WWW (2019)
5. Brochier, R., Guille, A., Velcin, J.: Inductive document network embedding with topic-word attention. In: Advances in Information Retrieval. pp. 326–340. ECIR (2020)
6. Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., Sun, X.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 3438–3445. AAAI (2020)
7. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. EMNLP (2014)
8. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of the International Conference on Neural Information Processing Systems. pp. 3844–3852. NeurIPS (2016)

9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics. pp. 4171–4186. NAACL (2019)
10. He, R., Ravula, A., Kanagal, B., Ainslie, J.: Realformer: Transformer likes residual attention. In: Proceedings of the Joint Conference of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing. ACL-IJCNLP (2021)
11. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Proceeding of the European Conference on Machine Learning. pp. 137–142. ECML (1998)
12. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 1746–1751. EMNLP (2014)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations. ICLR (2014)
14. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. In: Proceedings of the International Conference on Learning Representations. ICLR (2017)
15. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Advances in neural information processing systems. pp. 971–980. NeurIPS (2017)
16. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.S.: Gated graph sequence neural networks. In: Proceedings of the International Conference on Learning Representations. ICLR (2016)
17. Liu, J., Chang, W.C., Wu, Y., Yang, Y.: Deep learning for extreme multi-label text classification. In: Proceedings of the International ACM Conference on Research and Development in Information Retrieval. pp. 115–124. SIGIR (2017)
18. Lv, Q., Ding, M., Liu, Q., Chen, Y., Feng, W., He, S., Zhou, C., Jiang, J., Dong, Y., Tang, J.: Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In: Proceedings of the ACM Conference on Knowledge Discovery amp; Data Mining. pp. 1150–1160. KDD (2021)
19. Merity, S., Keskar, N.S., Socher, R.: Regularizing and optimizing lstm language models. in international conference on learning representations. In: Proceedings of the International Conference on Learning Representation. ICLR (2018)
20. Nikolentzos, G., Tixier, A., Vazirgiannis, M.: Message passing attention networks for document understanding. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 8544–8551. AAAI (2020)
21. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceeding of the International Conference on Empirical Methods in Natural Language Processing. pp. 1532–1543. EMNLP (2014)
22. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. In: Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing. pp. 1–5. EMC2 @ NeurIPS (2019)
23. Shen, T., Jiang, J., Zhou, T., Pan, S., Long, G., Zhang, C.: Disan: Directional self-attention network for rnn/cnn-free language understanding. In: Proceedings of the AAAI Conference on Artificial Intelligence. AAAI (2018)
24. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in NLP. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 3645–3650. EMNLP (2019)

25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5998–6008. NeurIPS (2017)
26. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. ICLR (2018)
27. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics. pp. 1480–1489. NAACL (2016)
28. Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 7370–7377. AAAI (2019)
29. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in neural information processing systems. pp. 649–657. NeurIPS (2015)
30. Zhang, Y., Yu, X., Cui, Z., Wu, S., Wen, Z., Wang, L.: Every document owns its structure: Inductive text classification via graph neural networks. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics. pp. 334–339. ACL (2020)