*applied sciences*

MDPI

*Article*

# TC-HGAT: A Task-Conditioned Hierarchical Graph Attention Network for ESG Text Classification

**Yasser Elouargui *, Abdellatif Sassioui, Meriyem Chergui, Mohamed El Kamili and Mohammed Ouzzif**

C3S Laboratory, Higher School of Technology, Hassan II University, Casablanca 20010, Morocco; meriyem.chergui@ensem.ac.ma (M.C.)
* Correspondence: yasser.elouargui-etu@etu.univh2c.ma

**Abstract**

Classifying corporate environmental, social, and governance (ESG) disclosures is challenging because they combine technical terminology with subtle, domain-specific distinctions. Pretrained Transformers such as ClimateBERT perform well but require many parameters and substantial GPU memory. This paper introduces TC-HGAT, a lightweight hierarchical graph attention network with task-semantic conditioning. TC-HGAT uses frozen FastText-300D embeddings and builds, for each document, a graph of word nodes, virtual cluster nodes, and a task-semantic node obtained by mean-pooling the embeddings of a natural-language task description. On three ESG tasks (Claim, Specificity, and Commitment), TC-HGAT achieves macro-F1 scores of 94.6%, 77.8%, and 79.3%, with the highest mean scores on Claim and Commitment, although no difference from ClimateBERT is statistically significant. Most of the gain comes from the task-semantic node, which improves macro-F1 significantly on all three tasks (by 5.7, 2.4, and 4.5 points), whereas the cluster layer alone gives a modest improvement that is significant only on Claim. Random vectors in the same graph position, including a norm-matched one, do not reproduce this gain, indicating that it depends on the content of the task representation. TC-HGAT trains about 1.4 million parameters, far fewer than BERT-base and ClimateBERT, and needs substantially less peak GPU memory during training.

**Keywords:** Graph Neural Networks (GNNs); text classification; ESG disclosures; task-semantic conditioning; hierarchical graphs; machine learning; sustainability reporting

## 1. Introduction

Corporate environmental, social, and governance (ESG) information is shared through large amounts of reports, sustainability documents, and regulatory filings. Automatically classifying these texts is important for identifying climate-related claims, checking how clear and specific the statements are, and evaluating corporate sustainability commitments [1,2]. However, ESG text classification is challenging. Models need to understand specific meanings, handle technical terms, and work with different document structures. They also need to avoid simply learning superficial word patterns [3].

Pretrained Transformer models, such as BERT [4,5], and domain-specific models like ClimateBERT [6] and FinBERT [7], have achieved strong results in many natural language processing tasks by learning useful information from large text datasets. However, traditional document classification models usually treat each text as a separate sequence. They process the text without clearly representing the relationships between different parts of the text or allowing information to move between different levels of meaning. In

Check for updates

Academic Editor: Gabriele Maria Lozito

Received: 21 September 2026
Revised: 1 November 2026
Accepted: 4 November 2026
Published: 9 November 2026

**Copyright:** © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

addition, these models have many parameters and require substantial GPU memory to fine-tune, which can be a barrier for organizations that need to classify disclosures with limited computational resources.

Graph Neural Networks (GNNs) provide another useful approach because they can represent relationships between different elements using nodes and edges [8–10]. When text is represented as a graph, message-passing methods [11] allow each node to collect information from nearby words or documents. This helps the model learn better and more structured representations of the text [12]. Recent methods have also combined language models with GNNs to use both the meaning of the text and the relationships between different text elements [13].

However, the performance of a GNN classifier depends strongly on how the graph is built [14]. Common text graphs can have some important limitations. At the word level, a flat graph passes information only between directly connected tokens, so relating distant or semantically related words requires several propagation steps; in larger graphs this repeated compression of information through few edges is known as over-squashing [15,16]. Even for short texts, such as the sentence-level ESG statements studied here, a flat word graph provides no intermediate unit that groups semantically related words. At the document level, graphs usually contain no information about the classification task itself. Most existing methods mainly connect basic text units, such as words or documents. They do not include intermediate levels that group related semantic concepts, nor do they include a node that represents the classification task.

ESG classification adds three specific difficulties to this setting. First, the classes are separated by subtle cues: a vague sustainability statement and a specific, quantified target often share most of their vocabulary and differ only in a few tokens, such as numbers, dates, or modal verbs. Second, labeled ESG data are scarce; the benchmarks used in this study contain only 800 to 1040 training sentences per task, which favors compact models that can be trained reliably on small datasets. Third, the same text may need to be judged under different criteria (whether it is a claim, whether it is specific, whether it is a commitment), so a representation that is aware of the task objective can be useful. The novelty of this work lies in addressing the second and third difficulties within the graph structure itself: rather than adding parameters through a larger encoder, we add an intermediate layer of embedding-derived cluster nodes and a single task-semantic node, and we test with explicit control conditions whether the task-level representation, rather than the added structure, is responsible for the observed gains.

To address these limitations, we investigate a central research question: *Can graph representations be augmented with intermediate structural abstractions and task-semantic conditioning to improve classification performance while maintaining a low computational footprint?*

We address this question by introducing TC-HGAT, a task-conditioned hierarchical graph attention network. Operating on pretrained static 300-dimensional word embeddings, with FastText-300D used as the main configuration and GloVe-300D evaluated as an alternative embedding backbone, TC-HGAT constructs a multi-tier graph containing three distinct node types: low-level lexical word nodes, intermediate virtual cluster nodes, and a task-semantic node. Virtual cluster nodes aggregate semantically related words into intermediate representations, providing an additional level of abstraction between individual words and the document representation. The task-semantic node is obtained by mean-pooling the static embeddings of a natural-language description of the classification task. It is therefore a fixed task-level vector, shared by all documents of a task, rather than an instruction that the model interprets. It is connected to the intermediate cluster layer, where it participates in attention-based message passing.

We evaluate TC-HGAT across three benchmark ClimateBERT ESG tasks: claim detection, statement specificity, and commitment identification [3,6]. Through controlled ablation experiments, including correct, wrong-task, and unrelated task descriptions, and a fixed random vector placed in the same graph position, we examine whether the task representation contributes beyond the addition of graph structure and of an extra node alone. We further evaluate the sensitivity of the framework to different static embeddings, clustering algorithms, levels of task-description detail, and numbers of virtual clusters.

The main contributions of this work are as follows:

- We introduce TC-HGAT, a lightweight hierarchical GNN architecture that combines embedding-derived virtual cluster nodes with a task-semantic node for multi-scale message passing.
- We investigate task-semantic conditioning through a controlled multi-condition ablation protocol. The correct task description improves macro-F1 significantly on all three tasks over the configuration without a task node, whereas mismatched descriptions give much smaller gains, and unrelated descriptions and random vectors, including a norm-matched one, give none.
- We systematically evaluate the contribution of hierarchical graph structure by comparing flat word-level and hierarchical graph configurations. The cluster layer alone gives a modest, task-dependent improvement of up to 2.3 macro-F1 points over the flat Word-GCN configuration, which is statistically significant only on Claim.
- We evaluate the sensitivity of the proposed architecture to the embedding backbone using FastText-300D as the main configuration and GloVe-300D as an alternative, together with four clustering strategies, several levels of task-description detail, and different numbers of virtual clusters.
- We demonstrate that TC-HGAT achieves strong predictive performance with a substantially smaller computational footprint than the large Transformer-based baselines. In particular, the model uses approximately 1.4 million trainable parameters and substantially less peak GPU memory, with an end-to-end inference time of 8.4 ms per document, including per-document clustering (Section 5.14).

The remainder of this paper is organized as follows. Section 2 provides background and discusses related work. Section 3 presents the proposed TC-HGAT methodology and architectural design. Section 4 details the experimental setup and benchmark datasets. Section 5 presents the empirical results and efficiency profiling. Section 6 addresses limitations and potential threats to validity, and Section 7 concludes the paper.

## 2. Background and Related Work

### *2.1. Transformer-Based Text Representation*

Pretrained Transformer models are widely used for text representation and classification [5]. BERT introduced a bidirectional Transformer trained on large amounts of unlabeled text and showed that contextual word representations can be useful for many different tasks [4]. By considering the words around each token, Transformers can capture the meaning of words and some of the relationships between them more effectively than simple word-based features.

Despite their strong performance, Transformer models still represent text mainly as sequences. The relationships between different parts of the text are learned through self-attention rather than being explicitly represented as a graph. This can be a limitation for tasks where classification depends on relationships between words, phrases, or documents. These relationships can provide useful information and reveal patterns that are not directly represented in a standard sequence-based model.

### 2.2. Graph Neural Networks for Text Classification

GNNs are a useful way to represent data where the relationships between different elements are important [9,10], and a growing body of work has applied them to text classification [17]. TextGCN showed this by building a graph that includes both word and document nodes. It then uses graph convolutions to pass information between connected nodes [8]. This approach has also been extended to different types of graphs at the word and text levels [18].

An important distinction in this literature is between corpus-level and text-level graphs. Corpus-level methods such as TextGCN [8] and BertGCN [13] build a single graph over the whole corpus, including unlabeled test documents, and are therefore transductive. Text-level methods instead build one graph per document, which makes them inductive: a new document can be classified without rebuilding the corpus graph. Huang et al. [18] introduced text-level graphs with shared word-node parameters, and TextING [19] learned document-specific word graphs with gated graph networks.

Later studies have explored combining pretrained language models with GNNs. For example, BertGCN builds a graph that contains different types of text elements and uses BERT to create document representations [13]. Attention-based GNNs improve this process by giving different importance to neighboring nodes. Graph Attention Networks (GATs) assign different weights to neighboring nodes when passing information [20]. GATv2 further improves this method by using a more flexible attention mechanism that considers both the source and target node representations [21]. Beyond text, graph- and attention-based backbones are increasingly benchmarked as lightweight alternatives in other application domains, such as fault diagnosis and regression in building HVAC systems [22].

### 2.3. Graph Construction for Text Classification

The graph structure determines which nodes can share information with each other, which directly affects how the GNN understands the relationships in the data. Bugueño and de Melo compared different graph construction methods across several classification tasks and showed that choosing the right graph structure is an important part of the model, not just a simple preprocessing step [14].

Most existing methods mainly connect basic text elements. Although this allows information to move between connected elements, they do not clearly include intermediate nodes that group related concepts. They also do not directly include information about the classification task in the graph structure.

### 2.4. NLP for ESG and Climate-Related Disclosures

Several datasets and models have been developed specifically for climate and ESG text. ClimaText [23] provides sentence-level labels for climate-change topic detection. ClimateBERT [6] adapts a DistilRoBERTa model to climate-related text through continued pretraining and is evaluated on the claim, specificity, and commitment tasks used in this study [3]. ClimateBERT-NetZero [24] extends this line of work to the detection of net-zero and emission-reduction targets, and ESGBERT [25] provides domain-adapted models for environmental, social, and governance text. These approaches rely on fine-tuning Transformer encoders with tens of millions of parameters. The present work is complementary: it targets the same benchmark tasks with a graph model whose trainable parameters are restricted to two attention layers and a linear head.

### *2.5. Research Gap*

Previous studies show that Transformers are effective for representing text sequences [4], while GNNs provide a clear way to pass information between related elements [8,11]. They also show that the way a graph is built controls how information moves through it [14]. However, most graph-based classification models use only the original text elements to build their graphs.

More specifically, TextGCN [8] and Word-GCN mainly connect words and documents or words with other words. These flat structures provide no intermediate level that groups related meanings, and in larger graphs they are susceptible to over-squashing. BertGCN [13] combines BERT with graph convolutions, but it still requires many parameters and does not directly include the classification task in the graph. A hierarchical word–cluster configuration, which we use as the task-node-free counterpart of our proposed architecture, adds connections between words and virtual clusters but does not include a task-semantic node. We refer to this configuration as *Hierarchical-GAT* throughout the experiments.

We address this gap by adding two new types of nodes to the graph: virtual cluster nodes and a task-semantic node. Virtual cluster nodes create an intermediate layer that groups related text elements [26–28]. The task-semantic node is a fixed vector obtained from the static embeddings of a short task description. It adds task-level information to the graph and interacts with the cluster nodes during attention-based message passing. Unlike instruction-following in large language models, this node is not interpreted or reasoned over by the model; it acts only as a task-conditioning vector.

## 3. Proposed Method

This section presents TC-HGAT, a hierarchical graph attention network with task-semantic conditioning. The complete end-to-end architecture is illustrated in Figure 1. To capture semantic information at multiple levels of abstraction, each text instance is represented as a hierarchical graph composed of three node types: word nodes at the lexical level, virtual cluster nodes at an intermediate semantic level, and a task-semantic node at the task level. Graph attention layers then perform message passing across these levels to obtain a graph-level representation for classification.

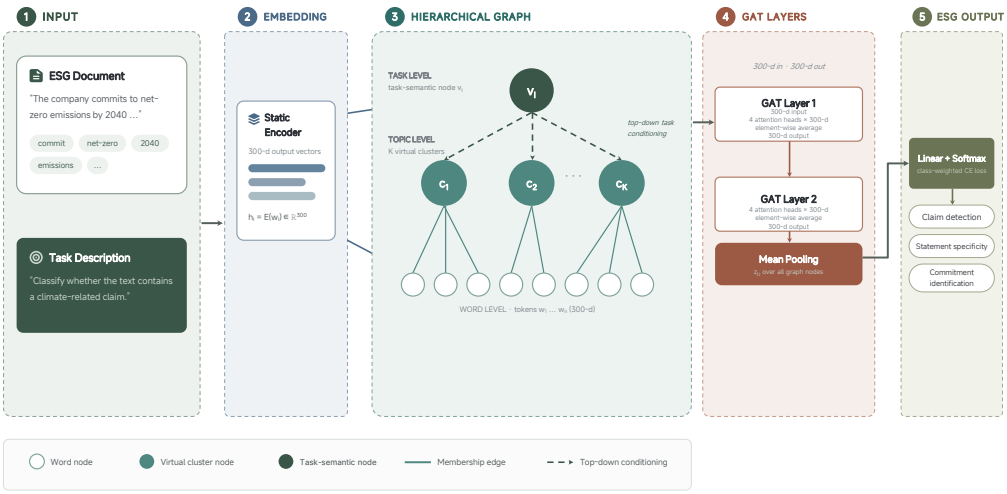



**Figure 1.** End-to-end architecture of TC-HGAT, the hierarchical graph attention network with task-semantic conditioning. Ellipses (...) indicate tokens, text, and nodes omitted for readability.

### 3.1. Problem Formulation

Let a text instance be represented as a sequence of tokens $D = (w_1, w_2, \ldots, w_n)$, where $n$ denotes the sequence length. Let $\mathcal{T}$ denote the classification task and $y$ its corresponding target label. The objective is to learn a predictive mapping $f(D, \mathcal{T}) \rightarrow y$ by constructing and learning over a task-aware hierarchical graph $G_D$.

### 3.2. Word Representation

Each token $w_i \in D$ is mapped to a 300-dimensional continuous vector using a pretrained static embedding function:

$$h_i = E(w_i), \qquad h_i \in \mathbb{R}^{300}, \tag{1}$$

where $E(\cdot)$ denotes the embedding lookup function. FastText-300D is used as the main embedding configuration, while GloVe-300D is evaluated as an alternative embedding backbone in a dedicated sensitivity experiment [29,30]. The embedding representations are kept fixed during graph model training, such that the trainable parameters are restricted to the graph neural network and classification components.

Tokens that are not present in the pretrained vocabulary are handled as follows. For FastText, an out-of-vocabulary (OOV) token is represented by the average of the vectors of its character $n$-grams, as provided by the pretrained FastText binary model, so that every retained token receives a 300-dimensional vector. For GloVe, which has no subword information, OOV tokens are mapped to a zero vector. OOV tokens are rare in the evaluated datasets: the token-level OOV rate is 2.28% (Claim) and 1.98% (Specificity and Commitment, which share the same sentences) for FastText, and 2.16% and 2.07% for GloVe.

### 3.3. Hierarchical Graph Construction

To construct the multi-level topology, the graph node set $V$ is partitioned into three subsets:

$$V = V_W \cup V_C \cup \{v_I\}, \tag{2}$$

where $V_W = \{v_1^w, \ldots, v_n^w\}$ represents the document word nodes, $V_C$ represents the virtual cluster nodes, and $v_I$ denotes the task-semantic node. The number of virtual cluster nodes is document-dependent and satisfies $|V_C| \leq K$, where $K$ denotes the requested maximum number of clusters.

For each document, the embeddings of its retained unique words are clustered independently in the embedding space. Let $C_k$ denote the set of word indices assigned to the $k$-th observed cluster in the document. If the number of retained document words is smaller than $K$, the effective number of clusters is reduced accordingly. Only observed, non-empty clusters are instantiated as virtual cluster nodes. Thus, for every instantiated cluster, $|C_k| > 0$.

For each observed cluster, the corresponding virtual node is initialized as the mean of the embeddings of its constituent words:

$$c_k = \frac{1}{|C_k|} \sum_{i \in C_k} h_i, \qquad C_k \neq \emptyset, \qquad c_k \in \mathbb{R}^{300}. \tag{3}$$

The resulting cluster nodes provide an intermediate semantic representation between individual lexical units and the task-level task-semantic node. During model training, the initial cluster assignments remain fixed, while the corresponding node representations are updated by the graph attention layers.

### 3.4. Task-Semantic Node

Let $t = (t_1, t_2, \ldots, t_m)$ denote a natural-language description of classification task $\mathcal{T}$, where $m$ is the number of tokens in the description. Because the proposed framework operates in the same 300-dimensional static embedding space as the word and cluster nodes, the task description is represented by mean-pooling the embeddings of its constituent tokens:

$$h_I = \frac{1}{m} \sum_{j=1}^{m} E(t_j), \qquad h_I \in \mathbb{R}^{300}. \tag{4}$$

Mean pooling places $h_I$ in the same space and at the same scale as the mean-initialized cluster nodes without introducing additional parameters or a separate encoder. Section 5.8 compares this choice with IDF-weighted and max pooling.

The resulting task-semantic representation is shared across documents belonging to the same classification task. Consequently, the task-semantic node provides task-level semantic information rather than document-specific contextual information. This design avoids introducing a separate contextual language encoder and preserves the lightweight computational structure of TC-HGAT. Because the representation is built from static word embeddings, the model does not interpret the task description or model interactions between its tokens; the node acts as a fixed task-conditioning vector rather than as an instruction in the sense used for large language models.

Because $h_I$ is identical for every document within a task, it cannot by itself separate the classes of a binary task. Any effect on classification must therefore come from its interaction with the document-specific cluster nodes during attention-based message passing, or from its contribution to the pooled graph representation. Section 5.6 separates these two routes.

The task-semantic node is connected to the virtual cluster nodes, allowing task-level information to participate in message passing at the intermediate semantic level. The exact task descriptions used in the experiments are reported in Appendix A.1.

### 3.5. Graph Topology and Edge Definition

The hierarchical graph is defined as $G_D = (V, E)$, where the edge set consists of word–cluster connections $E_{WC}$ and task–cluster connections $E_{TC}$:

$$E = E_{WC} \cup E_{TC}. \tag{5}$$

A word node $v_i^w$ is connected to a cluster node $v_k^c$ when the word belongs to cluster $k$, that is, when $i \in C_k$. The task-semantic node $v_I$ is connected to every instantiated virtual cluster node:

$$E_{WC} = \{(v_i^w, v_k^c) : i \in C_k\}, \tag{6}$$

and

$$E_{TC} = \{(v_I, v_k^c) : C_k \neq \emptyset\}. \tag{7}$$

The graph is treated as undirected, and self-loops are added to all nodes to preserve their own representations during attention-based message passing. No direct word–word or task-node–word edges are introduced. Consequently, the explicit hierarchical structure of the graph can be summarized as

$$\text{Task-semantic node} \iff \text{Virtual Clusters} \iff \text{Words}. \tag{8}$$

This topology establishes the virtual clusters as intermediate semantic bottlenecks through which information is exchanged between the lexical and task levels.

The task-semantic node is connected to the cluster layer rather than to the word layer for two reasons. First, the number of task edges is then bounded by $K$ and does not

grow with document length. Second, because $h_I$ is identical for all documents in a task, a direct edge to every word would add a largely document-independent term to every word representation; connecting it to cluster nodes lets it interact with representations that already aggregate document-specific content. With two attention layers, information from $v_I$ still reaches the word nodes through the clusters. A variant that connects $v_I$ to both the cluster and the word nodes is evaluated in Section 5.4.

### 3.6. GNN Message Passing

The hierarchical graph is processed using two successive Graph Attention Network v2 (GATv2) layers. GATv2 enables the model to assign different attention weights to neighboring nodes based on their current representations.

At layer $l$, the representation produced by attention head $m$ for node $i$ is expressed as

$$h_i^{(l+1,m)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l,m)} \mathbf{W}^{(l,m)} h_j^{(l)}\right), \tag{9}$$

where $\mathcal{N}(i)$ denotes the neighborhood of node $i$, $\mathbf{W}^{(l,m)}$ is a trainable linear transformation, and $\sigma(\cdot)$ denotes the nonlinear activation function.

The attention coefficient is computed as

$$\alpha_{ij}^{(l,m)} = \mathrm{Softmax}_j\left(\mathbf{a}^{(l,m)T}\,\mathrm{LeakyReLU}\left(\mathbf{W}^{(l,m)}\left[h_i^{(l)} \| h_j^{(l)}\right]\right)\right), \tag{10}$$

where $\mathbf{a}^{(l,m)}$ is a learnable attention parameter and $\|$ denotes concatenation of the source and target node representations.

Both graph attention layers use $M = 4$ attention heads. Each head produces a 300-dimensional representation. Rather than concatenating the outputs of the four heads, the head representations are averaged element-wise:

$$h_i^{(l+1)} = \frac{1}{M}\sum_{m=1}^{M} h_i^{(l+1,m)}. \tag{11}$$

Consequently, the multi-head operation preserves a 300-dimensional node representation throughout the network. This dimensionality-preserving aggregation maintains compatibility with the initial 300-dimensional word, cluster, and task-node representations.

Following graph message passing, a graph-level representation $z_D$ is obtained by mean pooling over all nodes:

$$z_D = \frac{1}{|V|}\sum_{i\in V} h_i^{(L)}, \tag{12}$$

where $L = 2$ denotes the number of graph attention layers.

In the default configuration, the pooling set $V$ includes the task-semantic node $v_I$ together with all word and cluster nodes. Because $v_I$ is a single node, its weight in $z_D$ is $1/|V|$ and decreases with document length. Part of its representation therefore reaches the classification head directly, and not only through message passing. Section 5.6 evaluates a variant in which $v_I$ is kept for message passing but excluded from the pooling, i.e., $z_D = \frac{1}{|V_W|+|V_C|}\sum_{i\in V_W\cup V_C} h_i^{(L)}$.

### 3.7. Classification and Training Objective

The pooled graph representation $z_D$ is passed through a linear classification head followed by a softmax function to obtain the predicted class probabilities:

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_o z_D + \mathbf{b}_o). \tag{13}$$

The framework is trained using a class-weighted cross-entropy objective to account for class imbalance:

$$\mathcal{L} = -\sum_{c=1}^{C} \omega_c y_c \log \hat{y}_c, \tag{14}$$

where $C$ denotes the number of target classes, $y_c$ is the ground-truth indicator for class $c$, $\hat{y}_c$ is the predicted probability for class $c$, and $\omega_c$ is the class-specific weighting factor.

The use of class-weighted cross-entropy ensures that classes with fewer training examples contribute proportionally to the optimization objective.

### 3.8. TC-HGAT Algorithm

Algorithm 1 summarizes the complete TC-HGAT procedure. Given a document $D$ and a task description $t$, the algorithm first obtains static word representations, assigns words to up to $K$ document-level clusters, and constructs the hierarchical graph. The task description is then represented in the same embedding space and connected to the virtual cluster layer. Graph attention message passing is subsequently applied, followed by graph-level mean pooling and classification.

The cluster assignments are determined during graph construction and remain fixed during model optimization. The graph neural network parameters and classification head are optimized using the class-weighted cross-entropy objective described above. The exact task descriptions are reported in Appendix A.1.

**Algorithm 1** TC-HGAT: hierarchical graph classification with task-semantic conditioning

**Require:** Document $D = (w_1, \ldots, w_n)$, task description $t = (t_1, \ldots, t_m)$, embedding function $E$, requested maximum number of clusters $K$
**Ensure:** Predicted class $\hat{y}$
1: Obtain word representations $h_i = E(w_i)$, where $h_i \in \mathbb{R}^{300}$.
2: Cluster the retained document word embeddings into up to $K$ clusters and retain only observed non-empty clusters.
3: Assign each document word to its observed cluster: $C_k = \{i : k_i = k\}$.
4: Initialize each virtual cluster node: $c_k = \frac{1}{|C_k|} \sum_{i \in C_k} h_i$, for $C_k \neq \emptyset$.
5: Encode the task description: $h_I = \frac{1}{m} \sum_{j=1}^{m} E(t_j)$.
6: Construct the hierarchical graph $V = V_W \cup V_C \cup \{v_I\}$ and $E = E_{WC} \cup E_{TC}$.
7: Connect each word to its assigned cluster and connect $v_I$ to every instantiated cluster node. Add self-loops.
8: Apply two successive GATv2 layers: $H^{(1)} = \text{GATv2}^{(1)}(G, H^{(0)})$ and $H^{(2)} = \text{GATv2}^{(2)}(G, H^{(1)})$.
9: Obtain the graph representation: $z_D = \frac{1}{|V|} \sum_{i \in V} h_i^{(2)}$ (including $v_I$).
10: Compute class probabilities: $\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_o z_D + \mathbf{b}_o)$.
11: During training, minimize the class-weighted cross-entropy loss and update the trainable graph and classification parameters using AdamW.
12: Return $\hat{y} = \arg\max_c \hat{y}_c$.

## 4. Experimental Setup

### 4.1. Datasets

We evaluate the proposed method on three ESG text classification tasks from the ClimateBERT benchmark: claim detection, statement specificity, and commitment identification. The claim detection dataset contains 1700 instances, while the specificity and

commitment datasets each contain 1320 instances. All three tasks are formulated as binary classification problems.

The test partitions correspond to the official test splits released with the ClimateBERT datasets, comprising 400 instances for claim detection and 320 instances each for specificity and commitment classification. For each task, the official training split was partitioned into training and validation sets using stratified random sampling, with 20% of the original training instances allocated to validation and a fixed random seed of 42. This procedure yielded 1040 training and 260 validation instances for claim detection, and 800 training and 200 validation instances for each of the other two tasks. The resulting partitions were kept fixed across all models to ensure a controlled and reproducible comparison. Exact duplicate documents and train–test overlaps were checked for and were absent from the resulting datasets.

Table 1 summarizes the main characteristics of the three datasets. The tasks exhibit different levels of class imbalance. The claim detection dataset is the most imbalanced, with the majority class accounting for 77.65% of the instances. The specificity and commitment datasets are more balanced, although their majority classes still represent 60.45% and 60.38% of the data, respectively. This class distribution motivates the use of macro-F1 as the primary evaluation metric and class-weighted cross-entropy during model training.

**Table 1.** Statistics of the ESG classification datasets.

| Statistic | Claim | Specificity | Commitment |
|---|---|---|---|
| Total documents | 1700 | 1320 | 1320 |
| Train | 1040 | 800 | 800 |
| Validation | 260 | 200 | 200 |
| Test | 400 | 320 | 320 |
| Classes | 2 | 2 | 2 |
| Majority class | 1 | 0 | 0 |
| Majority proportion | 77.65% | 60.45% | 60.38% |

### 4.2. Baselines

The evaluation compares TC-HGAT against both shallow and deep architectures:

- **Shallow Baseline:** TF-IDF + Logistic Regression (LR).
- **Transformer Baselines:** TinyBERT [31], BERT-base [4], and ClimateBERT [6].
- **Graph Baselines:** Word-GCN (flat token graph), TextGCN [8], BertGCN [13], and Hierarchical-GAT, the task-node-free hierarchical configuration developed as the structural counterpart of TC-HGAT.
- **Static-embedding Baselines:** A fastText classifier [32], a BiLSTM initialized with FastText-300D embeddings, and logistic regression over mean-pooled FastText-300D sentence embeddings.

*Baseline configurations.* The Transformer baselines were initialized from the pre-trained checkpoints `huawei-noah/TinyBERT_General_4L_312D` [31], `bert-base-uncased` [4], and `climatebert/distilroberta-base-climate-f` [6]. All three models were fine-tuned end-to-end with a linear classification head using the first-token representation and a maximum sequence length of 128 tokens. TF-IDF + Logistic Regression used unigram and bigram features, regularization strength $C = 1.0$, and class-balanced weights; because it is deterministic, its standard deviation across seeds is zero. Word-GCN is a two-layer GCN over a per-document word graph in which tokens are connected within a sliding window of 5 tokens, using the same static embeddings, hidden dimension, and mean-pooling readout as TC-HGAT. TextGCN and BertGCN followed the corpus-level graph construction principles of their respective original papers [8,13].

*Text-level versus corpus-level graphs.* TC-HGAT, Hierarchical-GAT, and Word-GCN build an independent graph for each document, whereas TextGCN and BertGCN build a single corpus-level graph with inter-document edges. We use text-level graphs because they are inductive: a new disclosure can be classified without rebuilding a corpus graph or retraining, and the graph of each instance depends only on that instance [18,19]. As a result, the comparison with TextGCN and BertGCN differs in both architecture and graph construction. These baselines also have access to word co-occurrence statistics across the whole corpus, including the unlabeled test documents, whereas TC-HGAT sees only the document being classified. This comparison should therefore be read as a system-level comparison between published graph approaches. The controlled comparisons that isolate the effect of our design choices are those between Word-GCN, Hierarchical-GAT, and TC-HGAT, which share the same per-document construction, embeddings, and training protocol.

### 4.3. Embedding Families and Clustering Strategies

We evaluate GloVe-300D and FastText-300D static embeddings independently. FastText-300D is used as the main embedding configuration, while GloVe-300D is evaluated as an alternative embedding backbone in a dedicated sensitivity experiment. Virtual cluster nodes are constructed using four clustering algorithms: K-Means ($K = 10$), Agglomerative Clustering, Spectral Clustering, and DBSCAN [33–35]. For each document, clustering is performed independently on the retained word embeddings of that document. Only observed, non-empty clusters are instantiated as virtual cluster nodes. The requested maximum number of clusters is selected as $K = 10$ for the main configuration, with the sensitivity to alternative values of $K$ evaluated separately in Section 5.12.

### 4.4. Task-Node Ablation Protocol

Six conditions are evaluated to test the contribution of the task-semantic node:

1. *None*: The task-node-free Hierarchical-GAT configuration, which uses the same word–cluster hierarchical topology as TC-HGAT but does not include a task-semantic node.
2. *Correct*: The description of the target task.
3. *Wrong ESG Task*: The description of a different ESG task (e.g., passing the commitment description to the claim task).
4. *Unrelated*: A non-ESG description (exact text in Appendix A.4).
5. *Fixed random*: A random 300-dimensional vector, drawn once and kept frozen during training, placed in the position of the task-semantic node. Its entries are drawn independently from the standard normal distribution $\mathcal{N}(0, 1)$ using PyTorch (version 2.1) with seed 42, and the same vector is used for all tasks and training seeds. Its L2 norm is 17.28, whereas the norms of the task vectors $h_I$ are 1.26, 1.32, and 1.20 for Claim, Specificity, and Commitment, respectively, so the unscaled random vector is about 13–14 times larger than the task vectors.
6. *Norm-matched random*: The same fixed random vector rescaled to the L2 norm of $h_I$ of the target task, so that it has the same scale as the task vector but no task content.

All conditions that contain a task node (2–6) use the same graph topology, GNN architecture, number of trainable parameters, data partitions, and training protocol; they differ only in the content of the node vector. The comparison between *Correct* and *Fixed random* therefore separates the effect of the task-description content from that of adding a node. Because GATv2 attention scores depend on the scale of the node representations, the norm-matched control additionally separates the content of the task vector from its scale.

### 4.5. Implementation Details and Training Configuration

#### 4.5.1. Hardware and Computational Environment

All experiments were conducted on a single laptop equipped with an NVIDIA RTX 4070 Laptop GPU (8 GB VRAM; NVIDIA Corporation, Santa Clara, CA, USA) and 60 GB of system memory. This configuration provides a constrained GPU-memory environment for evaluating the computational requirements of the proposed architecture. All models were implemented in Python (version 3.10) using PyTorch (version 2.1) and PyTorch Geometric (version 2.4). The Transformer baselines used the Hugging Face Transformers library (version 4.55.4); TF-IDF, logistic regression, and the clustering algorithms used scikit-learn (version 1.7.2); and the fastText classifier and FastText embeddings used the fastText library (version 0.9.3).

#### 4.5.2. Training Configuration

We used a consistent training pipeline across TC-HGAT and the evaluated neural baselines to ensure a controlled experimental comparison. Training was performed using mini-batches of 16 documents for 15 epochs. Optimization was carried out using the AdamW optimizer [36], with an initial learning rate of $2 \times 10^{-5}$. The loss function was a class-weighted cross-entropy loss to account for class imbalance within the three classification tasks.

In this shared protocol, no model-specific hyperparameter search was performed, so that all neural models received the same optimizer, learning rate, batch size, and training budget. A learning rate of $2 \times 10^{-5}$ is a standard choice for fine-tuning pretrained Transformers but is conservative for graph networks trained from random initialization; the shared setting is therefore not expected to favor the graph-based models. To verify that the comparison is not driven by this shared setting, every neural model was additionally tuned on the validation partitions over learning rates $\{2 \times 10^{-5}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$, dropout rates $\{0.1, 0.3, 0.5\}$, and 15 or 30 epochs. The configuration with the highest validation macro-F1 was selected for each model and task, and only then evaluated on the test partition. The selected configurations are reported in Appendix B, and the corresponding results in Section 5.3.

To account for stochastic variation in neural model training, the principal experiments were independently repeated using three random seeds: 13, 42, and 77. The same fixed train, validation, and test partitions were used across the corresponding experiments. Unless otherwise stated, reported results are presented as mean $\pm$ standard deviation across these three runs.

The graph hidden dimensionality was maintained at 300 dimensions across the two GATv2 layers. Each layer employed four attention heads, with the individual head outputs averaged element-wise so that the resulting node representation remained 300-dimensional rather than being concatenated. A dropout rate of 0.5 was used throughout the graph network. The default clustering configuration used a requested maximum of $K = 10$ clusters, as described in Section 4.3, unless otherwise specified in the corresponding sensitivity experiments.

All other hyperparameters were kept fixed across the corresponding experiments to isolate the effects of the architectural and representation choices under investigation. The complete configuration is summarized in Table 2.

All configuration choices, including the number of clusters $K$, clustering algorithm, embedding backbone, and task-description level, were selected using macro-F1 on the validation partitions only. Model selection followed a consistent checkpoint-selection protocol across all experiments. The official test partitions were not used for training, checkpoint selection, or hyperparameter tuning and were evaluated only after the experimental config-

urations had been fixed. Sensitivity analyses in Section 5 report test-set performance across configurations to characterize the effects of individual design choices.

**Table 2.** Summary of TC-HGAT implementation and hyperparameter settings.

| Parameter | Configured Value |
|---|---|
| Embedding backbone | FastText-300D (default) |
| Input embedding dimension | 300 |
| Graph hidden dimension | 300 |
| Attention heads ($M$) | 4 |
| GNN layers ($L$) | 2 |
| Dropout rate | 0.5 |
| Requested maximum clusters ($K$) | 10 |
| Cluster initialization | Mean of constituent word embeddings |
| Optimizer | AdamW |
| Learning rate | $2 \times 10^{-5}$ |
| Batch size | 16 |
| Training epochs | 15 |
| Loss function | Class-weighted cross-entropy |
| Train/Validation/Test split | Official test split; 20% of training split held out for validation |
| Checkpoint selection | Best validation macro-F1 across 15 epochs |
| Random seeds | 13, 42, 77 |
| GPU | NVIDIA RTX 4070 Laptop (8 GB) |
| System memory | 60 GB |
| Framework | PyTorch Geometric [37] (version 2.4) |

### 4.6. Evaluation Metrics

Model performance is evaluated using two complementary classification metrics: accuracy and macro-F1. Accuracy measures the overall proportion of correctly classified instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{15}$$

Because the ESG tasks exhibit class imbalance, macro-F1 is used as the primary evaluation metric. It computes the unweighted mean of the F1 scores obtained for the individual classes:

$$\text{Macro-F1} = \frac{1}{C} \sum_{c=1}^{C} \text{F1}_c, \tag{16}$$

where $C$ is the number of classes and $\text{F1}_c$ is the F1 score for class $c$. Macro-F1 gives equal importance to all classes and is therefore more informative than accuracy when the class distribution is imbalanced.

## 5. Experimental Results

This section presents the empirical evaluation of the proposed TC-HGAT framework, detailing the main comparative performance benchmarks, targeted ablation studies, hyperparameter sensitivities, and resource efficiency profile.

### 5.1. Main Performance Comparison

Table 3 compares accuracy and macro-F1 performance of the proposed TC-HGAT model against conventional text-classification algorithms, static-embedding baselines, Transformer-based architectures, and graph-based baselines across the three ESG classification tasks.

Unless otherwise stated, the TC-HGAT results reported in this section use the FastText-300D embedding backbone, the Detailed task description, and a requested maximum of $K = 10$ clusters.

**Table 3.** Accuracy and macro-F1 (%) across the three ESG classification tasks. Values are reported as mean ± standard deviation where available. Higher values indicate better performance. Params denotes trainable parameters. For TC-HGAT, Hierarchical-GAT, and Word-GCN it excludes the frozen static embeddings; for the BiLSTM it includes the fine-tuned embedding matrix, whose size depends on the task vocabulary. The fixed-random control uses the same topology and trainable parameters as TC-HGAT, with the task-node vector replaced by a frozen random vector.

| Model | Params | Claim | | Specificity | | Commitment | |
|---|---|---|---|---|---|---|---|
| | | Acc | Macro-F1 | Acc | Macro-F1 | Acc | Macro-F1 |
| TF-IDF + LR | N/A | 77.8 | 76.2 ± 0.0 | 74.1 | 73.0 ± 0.0 | 74.8 | 73.4 ± 0.0 |
| Mean-pooled FastText + LR | <1 K | 84.3 | 83.5 ± 0.0 | 72.8 | 71.9 ± 0.0 | 73.6 | 72.6 ± 0.0 |
| fastText classifier | N/A | 89.4 | 85.7 ± 0.0 | 77.2 | 74.6 ± 0.0 | 75.0 | 72.1 ± 0.0 |
| BiLSTM (FastText-300D init.) | ∼2.9–3.1 M | 87.9 | 83.9 ± 2.10 | 77.2 | 72.9 ± 1.63 | 77.5 | 73.8 ± 1.18 |
| TinyBERT | ∼15 M | 88.3 | 87.4 ± 0.7 | 76.1 | 74.8 ± 1.6 | 76.0 | 74.9 ± 1.6 |
| BERT-base | ∼110 M | 92.1 | 91.6 ± 1.2 | 80.2 | 79.4 ± 0.6 | 78.1 | 77.0 ± 0.3 |
| ClimateBERT | ∼82 M | 93.7 | 93.3 ± 0.6 | 79.5 | 78.6 ± 2.0 | 79.0 | 78.1 ± 0.1 |
| Word-GCN | ∼1 M | 87.0 | 86.6 ± 0.9 | 74.5 | 73.4 ± 2.1 | 75.4 | 74.7 ± 1.8 |
| TextGCN | ∼2 M | 85.4 | 85.0 ± 0.4 | 71.2 | 70.0 ± 0.7 | 71.5 | 70.0 ± 0.9 |
| BertGCN | ∼110 M | 93.2 | 93.0 ± 0.06 | 78.8 | 78.0 ± 0.53 | 78.2 | 77.0 ± 0.8 |
| Hierarchical-GAT (Ours, w/o task node) | ∼1.4 M | 89.4 | 88.9 ± 0.34 | 76.2 | 75.4 ± 0.84 | 75.8 | 74.8 ± 0.91 |
| Hierarchical-GAT + fixed random node | ∼1.4 M | 88.6 | 88.1 ± 0.6 | 75.5 | 74.6 ± 1.1 | 74.9 | 74.0 ± 1.2 |
| **TC-HGAT (Ours)** | **∼1.4 M** | **95.1** | **94.6 ± 0.46** | **78.5** | **77.8 ± 0.04** | **80.1** | **79.3 ± 1.4** |

Bold indicates the proposed model (TC-HGAT); it does not mark the best value in each column.

TC-HGAT achieves the highest overall macro-F1 scores among the evaluated models for Claim and Commitment classification, reaching 94.6% and 79.3%, respectively. For Specificity classification, TC-HGAT obtains a macro-F1 of 77.8%. Although this is below BERT-base (79.4%) and ClimateBERT (78.6%), TC-HGAT remains competitive with the Transformer-based and graph-based baselines while using substantially fewer trainable parameters.

TC-HGAT obtains a higher mean macro-F1 than ClimateBERT on Claim and Commitment, although the differences (1.3 and 1.2 points) are not statistically significant (Section 5.2).

The static-embedding baselines use the same pretrained FastText vectors as TC-HGAT at a comparable or lower cost, and therefore test whether the graph structure adds value beyond the embeddings themselves. Relative to the strongest of them on each task, TC-HGAT improves macro-F1 by 8.9 points on Claim and 3.2 points on Specificity (over the fastText classifier) and by 5.5 points on Commitment (over the BiLSTM), despite having fewer trainable parameters than the BiLSTM. Even without the task node, Hierarchical-GAT outperforms all static-embedding baselines on the three tasks (88.9%, 75.4%, and 74.8% versus at most 85.7%, 74.6%, and 73.8%). The gains of TC-HGAT are therefore not explained by the static embeddings alone.

Compared with the standard Word-GCN, TC-HGAT yields absolute macro-F1 improvements of 8.0 percentage points on Claim, 4.4 percentage points on Specificity, and 4.6 percentage points on Commitment. Relative to Hierarchical-GAT, the proposed model achieves gains of 5.7, 2.4, and 4.5 percentage points, respectively.

Because the task-node vector is identical for every document within a task, it cannot by itself separate the two classes; any gain must arise from how the node interacts with the document-specific cluster representations. To separate the effect of the description content from that of merely adding a node, the fixed-random control places a frozen random vector in the same graph position as the task node. It has exactly the same topology and trainable parameters as TC-HGAT, so the only difference is the content of the node. This control does

not reproduce the gain: it reaches 88.1%, 74.6%, and 74.0% macro-F1 on Claim, Specificity, and Commitment, 0.8 points below Hierarchical-GAT on each task and 6.5, 3.2, and 5.3 points below TC-HGAT. The improvement of TC-HGAT therefore depends on the content of the task description rather than on the presence of an additional node. A norm-matched version of this control, which removes differences in scale, gives the same conclusion (Section 5.5).

TC-HGAT also obtains a higher mean macro-F1 than BertGCN on Claim (+1.6 points) and Commitment (+2.3 points). On Specificity, however, BertGCN and ClimateBERT remain competitive, indicating that this task is comparatively more difficult for the proposed lightweight architecture.

### 5.2. Statistical Reliability and Per-Class Results

To assess whether the main differences are reliable, we computed 95% paired-bootstrap confidence intervals of the macro-F1 difference over the test instances, with $p$ values Holm–Bonferroni-adjusted within each task (Table 4). Per-class F1 scores are reported in Table 5.

**Table 4.** Paired comparisons of macro-F1 (%). Δ is the mean difference across the three seeds; CI is the unadjusted 95% paired-bootstrap confidence interval; $p$ values are Holm–Bonferroni-adjusted within each task.

| | Claim | | | Specificity | | | Commitment | | |
|---|---|---|---|---|---|---|---|---|---|
| **Comparison** | **Δ** | **95% CI** | $p$ | **Δ** | **95% CI** | $p$ | **Δ** | **95% CI** | $p$ |
| TC-HGAT vs. Hierarchical-GAT | +5.7 | [4.1, 7.3] | <0.001 | +2.4 | [0.6, 4.2] | 0.012 | +4.5 | [2.6, 6.4] | <0.001 |
| TC-HGAT vs. ClimateBERT | +1.3 | [−0.2, 2.8] | 0.19 | −0.8 | [−2.9, 1.3] | 0.47 | +1.2 | [−1.1, 3.5] | 0.31 |
| Hierarchical-GAT vs. Word-GCN | +2.3 | [0.7, 3.9] | 0.009 | +2.0 | [−0.3, 4.3] | 0.088 | +0.1 | [−2.0, 2.2] | 0.93 |

**Table 5.** Per-class F1 (%) on the test partitions, averaged over the three seeds. Class 1 denotes the positive label (claim, specific, commitment).

| | Claim | | Specificity | | Commitment | |
|---|---|---|---|---|---|---|
| **Model** | **$F1_0$** | **$F1_1$** | **$F1_0$** | **$F1_1$** | **$F1_0$** | **$F1_1$** |
| ClimateBERT | 89.4 | 97.2 | 82.1 | 75.1 | 81.9 | 74.3 |
| BERT-base | 87.0 | 96.2 | 82.6 | 76.2 | 80.8 | 73.2 |
| Hierarchical-GAT | 82.5 | 95.3 | 79.8 | 71.0 | 79.0 | 70.6 |
| TC-HGAT | 91.4 | 97.8 | 82.0 | 73.6 | 82.4 | 76.2 |

The improvement of TC-HGAT over Hierarchical-GAT is significant on all three tasks (adjusted $p \leq 0.012$). None of the differences between TC-HGAT and ClimateBERT is statistically significant: the confidence intervals include zero on all three tasks, so TC-HGAT is not significantly different from ClimateBERT on any task. For the hierarchy ablation, only the Claim gain of Hierarchical-GAT over Word-GCN is significant, and the Commitment gain of 0.1 points is not. On Specificity, the per-class results show that the gap to the Transformer baselines comes mainly from the positive class: TC-HGAT obtains an F1 of 73.6% on specific statements, compared with 75.1% for ClimateBERT and 76.2% for BERT-base, whereas its F1 on non-specific statements (82.0%) is on par with both.

### 5.3. Effect of Tuning the Baselines

Table 6 reports the test results of each neural model after validation-based tuning (Section 4.5).

After tuning, the ranking in Table 3 is preserved. TC-HGAT retains the highest mean macro-F1 on Claim (95.0%) and Commitment (79.6%), and remains below BERT-base, ClimateBERT, and BertGCN on Specificity. Tuning improves the Transformer baselines by 0.3–0.9 points and the graph baselines by 0.2–1.6 points, with the largest gains for TextGCN,

consistent with the shared learning rate being conservative for randomly initialized graph networks. TC-HGAT itself gains only 0.3–0.4 points, so its margin over Hierarchical-GAT narrows slightly to 5.3, 2.2, and 4.2 points. The main conclusions are therefore not an artifact of the shared training protocol.

**Table 6.** Macro-F1 (%) on the test partitions after validation-based tuning of each neural model (mean ± standard deviation over seeds). Selected configurations are listed in Appendix B.

| Model | Claim | Specificity | Commitment |
|---|---|---|---|
| TinyBERT | 88.1 ± 0.6 | 75.6 ± 1.3 | 75.8 ± 1.2 |
| BERT-base | 92.0 ± 0.9 | 79.7 ± 0.7 | 77.6 ± 0.6 |
| ClimateBERT | 93.6 ± 0.5 | 79.1 ± 1.2 | 78.5 ± 0.4 |
| Word-GCN | 87.4 ± 0.8 | 74.2 ± 1.5 | 75.3 ± 1.3 |
| TextGCN | 86.2 ± 0.5 | 71.4 ± 0.8 | 71.6 ± 0.9 |
| BertGCN | 93.2 ± 0.3 | 78.4 ± 0.6 | 77.5 ± 0.7 |
| Hierarchical-GAT | 89.7 ± 0.4 | 76.0 ± 0.7 | 75.4 ± 0.8 |
| TC-HGAT | 95.0 ± 0.4 | 78.2 ± 0.5 | 79.6 ± 0.9 |

### *5.4. Structural Hierarchy Ablation*

To separate the contribution of the cluster layer from that of the task-semantic node, Table 7 compares flat and hierarchical graphs, each with and without the task node. In the flat graphs, no cluster layer exists, so the task node $v_I$ is connected directly to all word nodes. All GATv2 configurations use the same embeddings, readout, and training protocol. The effect of the content of the task description (correct, mismatched, or unrelated) is analyzed separately in Section 5.5.

**Table 7.** Effect of the cluster layer and of the task-semantic node. Macro-F1 (%), mean ± standard deviation. Accuracies of the main configurations are reported in Table 3.

| Graph | Task Node | Task-Node Edges | Claim | Specificity | Commitment |
|---|---|---|---|---|---|
| Flat (Word-GCN) | No | – | 86.6 ± 0.9 | 73.4 ± 2.1 | 74.7 ± 1.8 |
| Flat (GATv2) | No | – | 87.4 ± 0.7 | 74.0 ± 1.6 | 74.9 ± 1.4 |
| Flat (GATv2) | Yes | words | 89.8 ± 0.6 | 74.9 ± 1.2 | 76.8 ± 1.1 |
| Hierarchical-GAT | No | – | 88.9 ± 0.34 | 75.4 ± 0.84 | 74.8 ± 0.91 |
| **TC-HGAT (default)** | **Yes** | **clusters** | **94.6 ± 0.46** | **77.8 ± 0.04** | **79.3 ± 1.4** |
| Hierarchical | Yes | clusters + words | 94.1 ± 0.5 | 77.3 ± 0.6 | 78.8 ± 1.0 |

Bold indicates the default TC-HGAT configuration, which gives the highest values in all columns.

*Effect of the cluster layer.* Adding virtual cluster nodes to the flat Word-GCN increases macro-F1 by 2.3 points on Claim (86.6% to 88.9%) and 2.0 points on Specificity (73.4% to 75.4%), whereas the change on Commitment (+0.1 points) is negligible. Only the Claim gain is statistically significant (Section 5.2). Relative to the flat GATv2 graph, which uses the same layer type, the changes are +1.5, +1.4, and −0.1 points. On its own, the cluster layer therefore provides a modest and task-dependent improvement.

*Effect of the task-semantic node.* Adding the task node improves macro-F1 in both graph types: by 2.4, 0.9, and 1.9 points in the flat GATv2 graph, and by 5.7, 2.4, and 4.5 points in the hierarchical graph (Claim, Specificity, Commitment). The gain from task conditioning is therefore larger when the cluster layer is present, which is consistent with task information being integrated more effectively through cluster-level representations than through direct connections to individual words; a formal interaction test would be needed to establish this statistically. Finally, connecting the task node to both clusters and words gives 0.5 points less than the default cluster-only connection on each task, which supports the cluster-only design described in Section 3.5.

### *5.5. Task-Node Ablation*

To determine whether the performance gains depend on the content of the task description rather than on the addition of a node, Table 8 evaluates the six conditions defined in Section 4.4: no task node, the correct task description, the description of a different ESG task, an unrelated description, a fixed random vector, and its norm-matched version. Figure 2 summarizes the effect of these conditions.

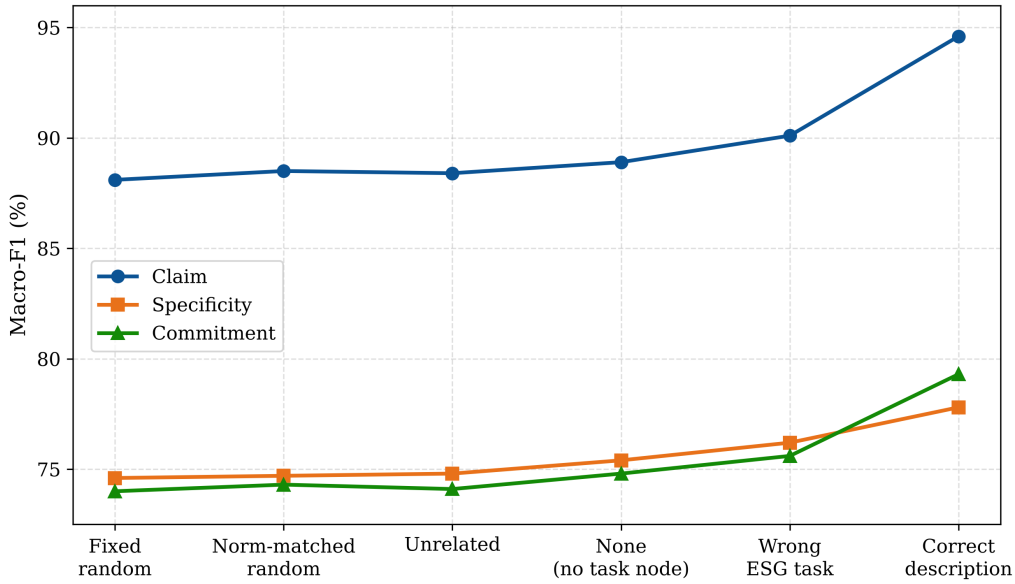



**Figure 2.** Effect of the task-node condition across the three ESG tasks.

**Table 8.** Ablation of the task-semantic node using accuracy and macro-F1 (%).

| Condition | Claim | | Specificity | | Commitment | |
|---|---|---|---|---|---|---|
| | Acc | Macro-F1 | Acc | Macro-F1 | Acc | Macro-F1 |
| None | 89.4 | 88.9 | 76.2 | 75.4 | 75.8 | 74.8 |
| **Correct** | **95.1** | **94.6** | **78.5** | **77.8** | **80.1** | **79.3** |
| Wrong ESG Task | 90.8 | 90.1 | 77.0 | 76.2 | 76.8 | 75.6 |
| Unrelated | 88.9 | 88.4 | 75.7 | 74.8 | 75.1 | 74.1 |
| Fixed random | 88.6 | 88.1 | 75.5 | 74.6 | 74.9 | 74.0 |
| Norm-matched random | 88.9 | 88.5 | 75.3 | 74.7 | 75.4 | 74.3 |

Bold indicates the condition used by TC-HGAT (correct task description), which gives the highest values in all columns.

The correct task description produces the strongest performance across all three classification tasks. Relative to the configuration without a task node, macro-F1 increases by 5.7 percentage points on Claim, 2.4 points on Specificity, and 4.5 points on Commitment.

The description of a different ESG task does not completely eliminate the benefit of conditioning. Instead, it produces intermediate performance, with macro-F1 values of 90.1%, 76.2%, and 75.6% for Claim, Specificity, and Commitment, respectively. A plausible explanation is that the ESG task descriptions share part of their vocabulary, so their mean-pooled vectors are partly similar to the correct one; the further the vector is from the correct description, the smaller the gain.

The unrelated description and the fixed random vector both perform 0.5–0.8 points below the configuration without a task node. Rescaling the random vector to the L2 norm of $h_I$ does not change this result: the norm-matched control reaches 88.5%, 74.7%, and 74.3% macro-F1 on Claim, Specificity, and Commitment, 0.4–0.7 points below the configuration without a task node and 6.1, 3.1, and 5.0 points below the correct description. It improves

on the unscaled random vector by only 0.1–0.4 points, so the scale of the added vector has little effect, and the gain of the correct description cannot be attributed to it. These controls show that adding a node to the graph, whether it carries an unrelated text representation or no semantic content at all, and whether or not its scale matches that of the task vector, does not improve classification. Rather, the relevance of the task description determines whether message passing through the task-semantic node is beneficial.

### *5.6. Ablation on Task-Node Readout*

In the default configuration, the task node $v_I$ is included in the mean-pooled graph representation $z_D$ (Section 3.6), so part of its representation reaches the linear classification head directly. To test whether the gains depend on this direct contribution, each configuration with a task node was retrained with $v_I$ excluded from the readout, while keeping it in the graph for message passing. The document representation then aggregates only the word and cluster nodes. The configuration without a task node is unaffected by this change. Results are reported in Table 9.

**Table 9.** Macro-F1 (%) with and without the task node $v_I$ in the graph-level readout (mean ± standard deviation over seeds). The "with $v_I$" values are those of Tables 3 and 8.

| | **Claim** | | **Specificity** | | **Commitment** | |
|---|---|---|---|---|---|---|
| **Condition** | **with $v_I$** | **without $v_I$** | **with $v_I$** | **without $v_I$** | **with $v_I$** | **without $v_I$** |
| None (no task node) | $88.9 \pm 0.34$ | | $75.4 \pm 0.84$ | | $74.8 \pm 0.91$ | |
| Correct | $94.6 \pm 0.46$ | $93.6 \pm 0.6$ | $77.8 \pm 0.04$ | $76.3 \pm 0.8$ | $79.3 \pm 1.4$ | $78.2 \pm 1.2$ |
| Wrong ESG Task | $90.1 \pm 0.6$ | $88.8 \pm 0.7$ | $76.2 \pm 0.9$ | $74.3 \pm 1.0$ | $75.6 \pm 1.0$ | $74.3 \pm 1.1$ |
| Unrelated | $88.4 \pm 0.5$ | $87.6 \pm 0.6$ | $74.8 \pm 1.0$ | $73.2 \pm 1.1$ | $74.1 \pm 0.9$ | $73.0 \pm 1.0$ |
| Fixed random | $88.1 \pm 0.6$ | $87.2 \pm 0.7$ | $74.6 \pm 1.1$ | $73.2 \pm 1.2$ | $74.0 \pm 1.2$ | $73.1 \pm 1.2$ |

Excluding $v_I$ from the readout lowers macro-F1 by 0.8–1.9 points across the conditions with a task node. For the correct description, the decrease is 1.0, 1.5, and 1.1 points, to 93.6%, 76.3%, and 78.2%. On Claim and Commitment, the correct condition then remains 4.7 and 3.4 points above the configuration without a task node, retaining most of its original advantage (5.7 and 4.5 points). On Specificity, however, the remaining margin is only 0.9 points (76.3% versus 75.4%), which is comparable to the standard deviations across seeds (0.8 and 0.84 points) and is therefore unlikely to be statistically reliable. We thus do not claim that the advantage of the correct description is retained on Specificity once $v_I$ is excluded from the readout. Every other condition, including the fixed random vector, falls below the configuration without a task node once $v_I$ is excluded. These results support the interpretation that, on Claim and Commitment, the task-description vector acts mainly by modulating the cluster and word representations through message passing, whereas on Specificity a larger share of its benefit depends on the readout contribution. The ablation does not, however, establish message passing as the only mechanism.

*Why the readout contribution is not a linear injection of the task vector.* Because $h_I$ is identical for all documents of a task, concatenating the input vector $h_I$ to $z_D$ before the linear head would give $\mathbf{W}_o[z_D; h_I] + \mathbf{b}_o = \mathbf{W}_o^{(z)} z_D + (\mathbf{W}_o^{(I)} h_I + \mathbf{b}_o)$, where the term in parentheses is the same for every document and can already be represented by the learned bias $\mathbf{b}_o$; adding $h_I$ to $z_D$ leads to the same conclusion. Such a direct injection is therefore a reparameterization of the bias and does not increase the expressiveness of the classifier, so we did not evaluate it as a separate control. The vector that enters the readout in TC-HGAT is instead $h_I^{(L)}$, the state of $v_I$ after two GATv2 layers, which differs between documents because $v_I$ attends to and aggregates the document-specific cluster nodes, and which is weighted by $1/|V|$, a quantity that depends on document length. The 1.0–1.5-point readout

contribution of the correct description therefore arises from this document-dependent state produced by message passing, not from a constant shift of the logits.

### 5.7. Robustness to Task-Description Paraphrases

To test whether the improvement depends on the specific wording of the Detailed task descriptions, each Detailed description was rewritten in three semantically equivalent paraphrases (Appendix A.3) that preserve the task definition but change word choice and sentence structure. The paraphrases were written by the authors and fixed before any test evaluation. Each paraphrase was used in place of the original description under the same training protocol, with all other settings unchanged. The results are reported in Table 10.

**Table 10.** Macro-F1 (%) with the original Detailed task description and with three paraphrases (P1–P3). The mean and standard deviation are computed across the four formulations. *None* is the task-node-free Hierarchical-GAT configuration.

| Task Description | Claim | Specificity | Commitment |
|---|---|---|---|
| Original (Detailed) | 94.6 | 77.8 | 79.3 |
| Paraphrase P1 | 94.2 | 77.4 | 79.0 |
| Paraphrase P2 | 93.9 | 77.6 | 78.6 |
| Paraphrase P3 | 94.4 | 77.1 | 79.1 |
| Mean ± SD across formulations | 94.3 ± 0.3 | 77.5 ± 0.3 | 79.0 ± 0.3 |
| None (no task node) | 88.9 | 75.4 | 74.8 |

Across the four formulations, macro-F1 varies by at most 0.7 points on each task, and every paraphrase remains at least 5.0, 1.7, and 3.8 points above the configuration without a task node on Claim, Specificity, and Commitment, respectively. The improvement therefore does not depend on one particular wording of the description. The mean-pooled embeddings of the original and paraphrased descriptions have an average cosine similarity of 0.94. This high similarity is expected, because averaging static embeddings of near-synonyms produces similar vectors, and it means that this experiment tests robustness to rewording rather than to substantially different formulations of the task.

### 5.8. Sensitivity to Task-Description Pooling

To test whether the results depend on mean pooling, Table 11 compares it with two alternatives that remain in the 300-dimensional static embedding space and add no trainable parameters: IDF-weighted averaging, using IDF values computed on the training partition, and element-wise max pooling.

**Table 11.** Macro-F1 (%) for different pooling operators used to build $h_I$ from the Detailed description (mean ± standard deviation over the three seeds).

| Pooling | Claim | Specificity | Commitment |
|---|---|---|---|
| Mean (default) | 94.6 ± 0.46 | 77.8 ± 0.04 | 79.3 ± 1.4 |
| IDF-weighted mean | 94.5 ± 0.5 | 78.0 ± 0.7 | 79.2 ± 1.0 |
| Max | 92.7 ± 0.7 | 76.6 ± 0.9 | 77.9 ± 1.1 |
| No task node | 88.9 ± 0.34 | 75.4 ± 0.84 | 74.8 ± 0.91 |

IDF-weighted averaging performs within 0.2 points of mean pooling on all three tasks, a difference smaller than the standard deviations across seeds, whereas max pooling lowers macro-F1 by 1.9, 1.2, and 1.4 points. All pooling variants nevertheless remain above the configuration without a task node on every task, so the benefit of the task-semantic node does not depend on the choice of pooling operator, although averaging operators are

preferable. Max pooling likely performs worse because it produces a vector that lies outside the region spanned by the mean-initialized cluster nodes, which weakens the compatibility between the task node and its neighbors.

### 5.9. Sensitivity to Task-Description Detail

We further investigate the sensitivity of TC-HGAT to the level of detail provided in the task description. Four levels of detail are considered: Minimal, Standard, Detailed, and Expert. The Minimal description provides only the essential task definition, while the Standard description introduces additional contextual information. The Detailed description provides explicit ESG-related terms and corresponds to the configuration used in the main TC-HGAT model. The Expert description further increases the level of descriptive detail. The results are reported in Table 12.

**Table 12.** Sensitivity of TC-HGAT to task-description detail. Values are reported as Accuracy/Macro-F1 (%).

| Description Level | Claim | Specificity | Commitment |
|---|---|---|---|
| Minimal | 91.8/90.9 ± 0.5 | 76.8/76.0 ± 0.4 | 77.6/76.7 ± 0.5 |
| Standard | 93.7/92.9 ± 0.4 | 77.6/76.9 ± 0.3 | 79.1/78.3 ± 0.4 |
| **Detailed (Base)** | **95.1/94.6 ± 0.46** | **78.5/77.8 ± 0.04** | **80.1/79.3 ± 1.4** |
| Expert | 94.6/94.0 ± 0.5 | 78.2/77.5 ± 0.4 | 79.7/78.9 ± 0.5 |

Bold indicates the default description level (Detailed), which gives the highest values in all columns.

The results show that performance improves as the description becomes more detailed, up to the Detailed configuration. For Claim, macro-F1 increases from 90.9% with the Minimal description to 92.9% with the Standard description and 94.6% with the Detailed description. For Specificity, the values are 76.0%, 76.9%, and 77.8% for the Minimal, Standard, and Detailed descriptions, respectively. For Commitment, the corresponding values are 76.7%, 78.3%, and 79.3%.

The Detailed description, which was selected as the default on the validation partitions, also gives the best test results. However, making the description more detailed with the Expert configuration does not improve the results. Instead, performance decreases slightly for all three tasks. Because the task vector is a mean over all description tokens, a longer description can dilute the most task-specific words with more general ones.

Overall, the results show that the content of the description is more important than simply making it longer. A detailed description gives a more informative task vector for conditioning the cluster representations, while adding too much information does not lead to better results.

### 5.10. Static Embedding Sensitivity

Table 13 compares the main FastText-300D configuration with the alternative GloVe-300D embedding backbone.

**Table 13.** Performance comparison between GloVe and FastText static backbones in TC-HGAT (%).

| Embedding | Claim | | Specificity | | Commitment | |
|---|---|---|---|---|---|---|
| | Acc | Macro-F1 | Acc | Macro-F1 | Acc | Macro-F1 |
| GloVe | 94.6 | 94.1 | 77.8 | 77.0 | 79.2 | 78.4 |
| **FastText** | **95.1** | **94.6** | **78.5** | **77.8** | **80.1** | **79.3** |

Bold indicates the default embedding backbone (FastText), which gives the highest values in all columns.

The results indicate a consistent but moderate advantage for FastText across all three downstream tasks. FastText improves macro-F1 by 0.5 percentage points on Claim, 0.8 points on Specificity, and 0.9 points on Commitment.

The improvement is accompanied by small gains in both evaluation metrics, rather than a large shift toward a single metric. This pattern suggests that subword-level representations may provide useful coverage for technical ESG terminology and domain-specific vocabulary.

Nevertheless, the magnitude of the improvement remains modest, indicating that the principal performance gains of TC-HGAT are unlikely to be explained solely by the choice of static embedding backbone.

### *5.11. Clustering Strategy Evaluation*

Table 14 evaluates the sensitivity of TC-HGAT to the algorithm used for virtual cluster construction. K-Means, Agglomerative, and Spectral clustering are evaluated with a predefined number of clusters, whereas DBSCAN determines the number of clusters from the local density of token representations. Consequently, the requested cluster count $K = 10$ applies to the fixed-cardinality clustering methods and does not constrain the number of clusters produced by DBSCAN.

**Table 14.** Impact of clustering algorithms on TC-HGAT classification performance (%).

| Clustering | Claim | | Specificity | | Commitment | |
|---|---|---|---|---|---|---|
| | Acc | Macro-F1 | Acc | Macro-F1 | Acc | Macro-F1 |
| **K-Means** | **95.1** | **94.6** | **78.5** | **77.8** | **80.1** | **79.3** |
| Agglomerative | 94.4 | 93.9 | 77.9 | 77.1 | 79.3 | 78.5 |
| Spectral | 94.0 | 93.5 | 77.6 | 76.8 | 79.8 | 79.1 |
| DBSCAN | 91.8 | 91.2 | 78.0 | 77.4 | 77.3 | 76.8 |

Bold indicates the default clustering algorithm (K-Means), which gives the highest values in all columns.

K-Means, selected as the default configuration using the validation partitions, also achieves the strongest overall test performance across the three tasks.

Agglomerative and Spectral clustering remain competitive with K-Means. Relative to K-Means, Agglomerative clustering yields macro-F1 scores lower by 0.7, 0.7, and 0.8 percentage points for Claim, Specificity, and Commitment, respectively. The corresponding differences for Spectral clustering are 1.1, 1.0, and 0.2 percentage points. These results indicate that the hierarchical representation can accommodate alternative partitioning strategies, although K-Means achieves the strongest overall performance in this evaluation.

DBSCAN exhibits a more task-dependent performance pattern. Its hyperparameters were tuned using the validation partitions over a grid of 15 configurations, with `eps` $\in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and `min_samples` $\in \{2, 3, 5\}$, using cosine distance throughout. The results reported in Table 14 correspond to the selected configuration, `eps` $= 0.4$ and `min_samples` $= 5$. Relative to K-Means, DBSCAN achieves macro-F1 scores lower by 3.4 percentage points for Claim and 2.5 percentage points for Commitment, whereas the difference for Specificity is only 0.4 percentage points. Thus, DBSCAN remains competitive for Specificity but performs less effectively on Claim and Commitment under the evaluated configuration. These differences may reflect the sensitivity of density-based clustering to the distribution and local density of token representations, which can affect the resulting hierarchical graph structure.

Unlike fixed-cardinality clustering methods, DBSCAN can produce a variable number of clusters and may classify individual tokens as noise. Noise tokens are retained as word-token nodes in the graph but are not assigned to virtual cluster nodes. Virtual cluster nodes are created only for non-empty clusters. If DBSCAN produces no clusters for a document,

the graph retains the task-semantic node and word-token nodes without virtual cluster nodes. Because all edges pass through the cluster layer, these nodes then have no edges other than self-loops, and the prediction depends only on the readout of their individually transformed representations; this occurred for nearly 2% of documents. This handling preserves the token-level representation and allows documents for which density-based clustering identifies no clusters to be processed within the TC-HGAT framework.

### 5.12. Sensitivity to the Number of Semantic Clusters

The requested maximum number of clusters $K$ determines the granularity of the intermediate representation in the hierarchical graph. A small value of $K$ produces a coarse semantic organization, whereas larger values allow a finer partition of the retained word embeddings within each document. For the K-Means clustering configuration, we therefore evaluate TC-HGAT with $K \in \{2, 5, 10, 15, 20\}$ while keeping the remaining model configuration fixed. Because clustering is performed independently for each document, the realized number of virtual cluster nodes may be smaller than the requested value of $K$. The results are reported in Table 15.

**Table 15.** Sensitivity of TC-HGAT to the number of semantic clusters $K$. Values are reported as Accuracy/Macro-F1 (%).

| $K$ | Claim | Specificity | Commitment |
|---|---|---|---|
| 2 | 93.2/92.5 | 77.0/76.3 | 78.6/77.8 |
| 5 | 94.3/93.7 | 77.8/77.1 | 79.5/78.7 |
| **10 (Base)** | **95.1/94.6 ± 0.46** | **78.5/77.8 ± 0.04** | **80.1/79.3 ± 1.4** |
| 15 | 94.8/94.2 | 78.2/77.5 | 79.8/79.0 |
| 20 | 94.5/93.9 | 77.9/77.2 | 79.5/78.7 |

Bold indicates the default number of clusters (K = 10), which gives the highest values in all columns.

The results show that the choice of $K$ has a measurable effect on classification performance across the three ESG tasks. Using only two clusters produces a relatively coarse semantic partition and results in lower performance, with macro-F1 scores of 92.5%, 76.3%, and 77.8% for Claim, Specificity, and Commitment, respectively. Increasing the number of clusters to $K = 5$ improves these scores to 93.7%, 77.1%, and 78.7%.

Within the K-Means configuration, the best overall performance is obtained with $K = 10$, which corresponds to the base TC-HGAT configuration. At this setting, the model achieves macro-F1 scores of 94.6% for Claim, 77.8% for Specificity, and 79.3% for Commitment. Increasing the number of clusters beyond this point does not provide additional gains. With $K = 15$, macro-F1 decreases slightly to 94.2%, 77.5%, and 79.0%, while $K = 20$ produces 93.9%, 77.2%, and 78.7%, respectively.

This behavior suggests that an intermediate level of semantic granularity is beneficial for the hierarchical representation. When $K$ is too small, semantically distinct words may be grouped into overly broad clusters, limiting the discriminative capacity of the cluster-level representation. Conversely, increasing $K$ beyond 10 may produce increasingly fine-grained clusters without providing sufficient additional semantic structure to improve classification.

Overall, $K = 10$, which was selected on the validation partitions, also gives the best test results, providing a balance between coarse and overly fragmented semantic organization. The relatively small performance variations around this value also indicate that TC-HGAT is not excessively sensitive to moderate changes in the number of clusters.

### *5.13. Training Dynamics*

To examine optimization behavior and model convergence, Figure 3 plots the training and validation loss and macro-F1 convergence of TC-HGAT on the Claim detection task over 15 epochs.

The observed training behavior is characterized by a rapid reduction in training loss during the first six epochs, decreasing from above 0.90 to approximately 0.45, followed by a more gradual convergence toward 0.20 by epoch 15. Validation loss decreases concurrently and stabilizes around 0.28. The relatively small and stable difference between the training and validation curves does not indicate substantial overfitting during the observed 15-epoch training period.

The corresponding training and validation macro-F1 scores increase rapidly during the initial epochs, reaching approximately 85–89% by epoch 7, before entering a more stable region. The validation macro-F1 remains close to the training trajectory throughout the later epochs and reaches slightly above 94% by epoch 15. The overall alignment between training and validation performance is consistent with stable optimization and suggests that the model maintains similar performance on the validation data throughout the observed training period.

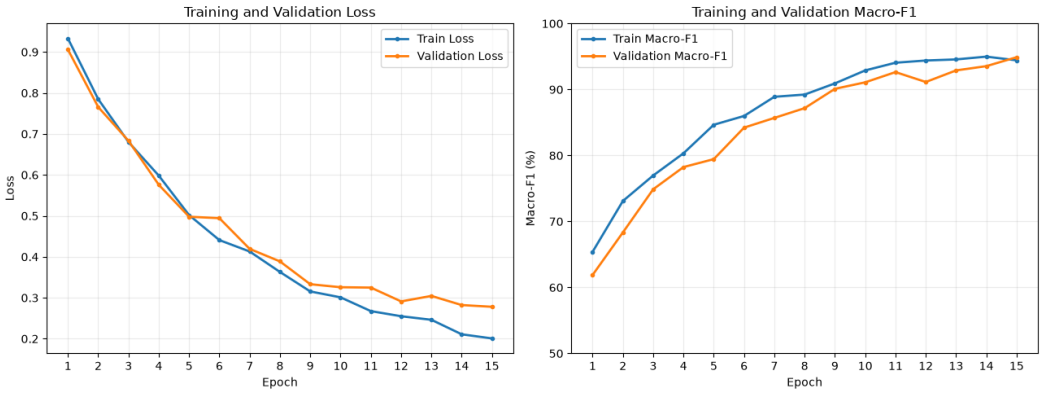



**Figure 3.** Training and validation loss (**left**) and macro-F1 (**right**) for TC-HGAT on the Claim detection task over 15 epochs.

### *5.14. Computational Efficiency Profiling*

We assess computational cost through four distinct quantities: the number of trainable parameters, the storage required by the frozen static embeddings, peak allocated GPU memory during training, and amortized inference time per document. These quantities capture different aspects of cost and are not interchangeable. Peak GPU memory and inference time are shown in Figures 4 and 5, respectively.

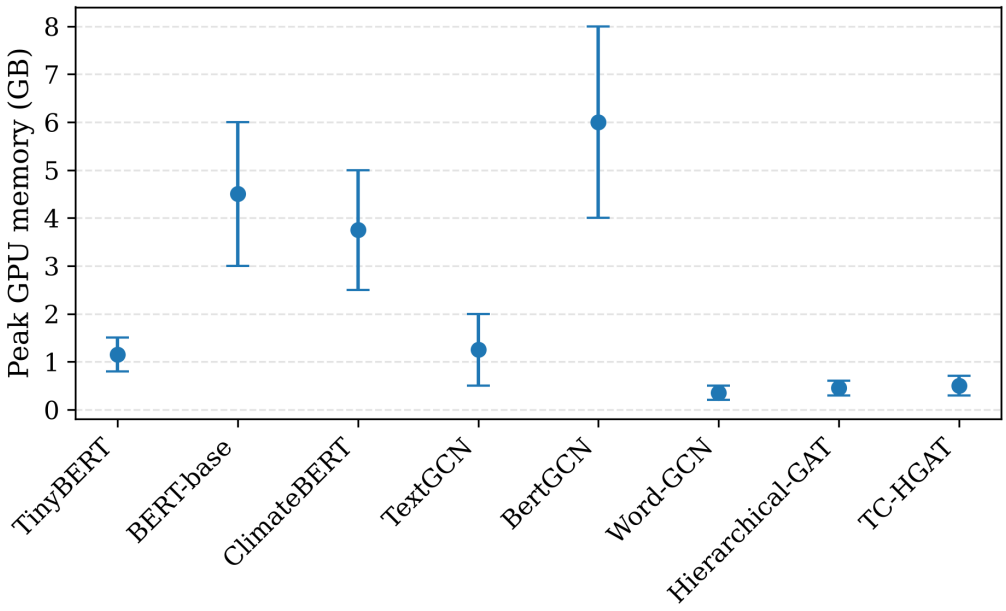



**Figure 4.** Peak allocated GPU memory during training across the evaluated models. Markers represent the midpoints of the observed ranges, and vertical lines indicate the minimum and maximum values. For TC-HGAT, the frozen embedding table and the clustering step reside on the CPU and are not included. Values are rounded to the nearest 0.1 GB.

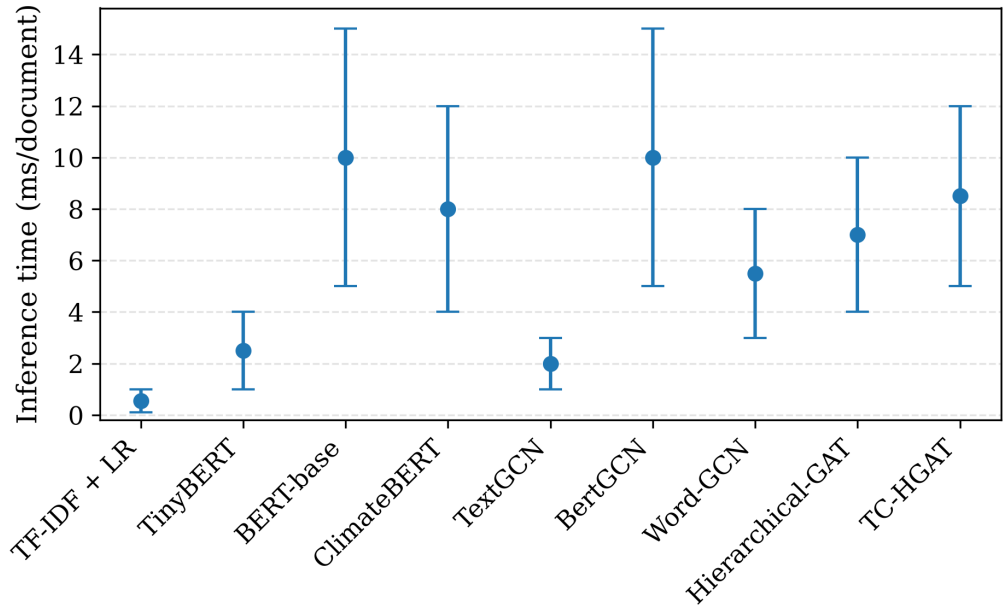



**Figure 5.** Amortized inference time per document across the evaluated models, using mini-batches of 16 documents. Markers represent the midpoints of the observed ranges, and vertical lines indicate the minimum and maximum values. For TC-HGAT, the values include embedding lookup, per-document clustering, and graph construction (Table 16).

*Measurement protocol.* All neural models were trained and evaluated with mini-batches of 16 documents. Peak allocated GPU memory during training was measured using PyTorch's CUDA memory statistics, with peak statistics reset at the beginning of each training run. The reported ranges span the three tasks and the random seeds. Inference time is the wall-clock time required to classify the full test partition after warm-up, divided by the number of test documents. It is therefore an amortized time per document, not a single-

document response latency. The timing scope depends on the preprocessing required by each model family. For the Transformer baselines, the measurement covers tokenization and the forward pass. For TC-HGAT, it covers the complete end-to-end pipeline: tokenization, CPU embedding lookup, per-document clustering, graph construction, and the GATv2 forward pass with classification.

Word-GCN and Hierarchical-GAT were timed with the same end-to-end procedure as TC-HGAT. For TextGCN and BertGCN, which classify the test documents as nodes of a precomputed corpus-level graph, the measurement covers the forward pass over that graph; the time needed to build the corpus graph is not included.

These figures describe the evaluated implementations on the hardware described in Section 4.5, rather than architecture-only costs.

*Trainable parameters and embedding storage.* TC-HGAT has approximately 1.4 million trainable parameters in its GATv2 layers and classification head. This count excludes the frozen 300-dimensional FastText embeddings, so it measures the number of parameters optimized during training, not the total size of the system. A direct comparison with the parameter counts of fine-tuned Transformers, which include their own embedding matrices, is therefore not a like-for-like comparison of model size.

The frozen embeddings instead incur a CPU-memory cost. In single precision, each word vector occupies $300 \times 4 = 1200$ bytes, so a table of two million vectors requires 2.4 GB of raw vector storage, plus indexing overhead and, if loaded, FastText subword information. Restricting the table to the vocabulary of the data reduces this cost substantially, but a pruned word-level table loses the subword support for unseen words. We therefore do not claim that TC-HGAT has lower total memory requirements than the Transformer baselines.

*Peak GPU memory during training.* TC-HGAT used approximately 0.3–0.7 GB of peak allocated GPU memory during training, compared with approximately 2.5–5 GB for ClimateBERT and 3–6 GB for BERT-base (Figure 4), and was successfully trained on the 8 GB laptop GPU used in this study. This advantage concerns GPU memory during training only; CPU-resident embeddings and preprocessing structures are not included.

*Amortized inference time.* End-to-end, TC-HGAT required approximately 5–12 ms per document (Figure 5). This is comparable to ClimateBERT (4–12 ms) and BERT-base (5–15 ms) and higher than TinyBERT (1–4 ms) and TF-IDF with logistic regression, so TC-HGAT offers no general inference-speed advantage. Table 16 decomposes this time on the Claim test set.

**Table 16.** Amortized inference time of TC-HGAT by processing stage on the Claim test set (ms per document, mini-batches of 16 documents).

| Processing Stage | Time (ms/Document) |
|---|---|
| Tokenization and embedding lookup (CPU) | 0.6 |
| Per-document clustering (K-Means, CPU) | 5.1 |
| Graph construction (CPU) | 0.9 |
| GATv2 forward pass and classification (GPU) | 1.8 |
| End-to-end total | 8.4 |

Per-document clustering accounts for approximately 61% of the end-to-end time, whereas the GATv2 forward pass and classification account for approximately 21%. Preprocessing, not the graph network, therefore dominates the inference cost of TC-HGAT. Because the clusters depend only on the document and not on the task or the trained parameters, they could be computed once and reused, for example when the same documents are classified for several tasks. We did not measure such a cached pipeline, however, and the 1.8 ms of the forward-pass stage should not be read as its end-to-end latency, since graph retrieval, batching, and CPU–GPU transfer would add to it.

*Interpretation.* Overall, TC-HGAT offers a favorable trade-off between accuracy and GPU memory during training: it reaches 94.6% macro-F1 on Claim and 79.3% on Commitment while optimizing approximately 1.4 million parameters and using less peak allocated GPU memory during training than the Transformer baselines. This comes at the cost of CPU storage for the frozen embeddings and of clustering and graph construction at inference time. TC-HGAT is therefore best suited to settings where GPU memory for training is constrained and these CPU-side costs are acceptable; the results do not support a general claim of lower total memory use or faster inference.

### *5.15. Discussion*

The empirical results support the view that hierarchical semantic aggregation and task-semantic conditioning provide complementary information for ESG text classification.

The structural ablation shows that the cluster layer on its own has a modest and task-dependent effect. Relative to the flat Word-GCN configuration, it increases macro-F1 by 2.3 points on Claim (86.6% to 88.9%), 2.0 points on Specificity, and 0.1 points on Commitment, and only the Claim gain is statistically significant ($p = 0.009$; Specificity $p = 0.088$; Commitment $p = 0.93$). The larger and consistently significant gains of TC-HGAT come from the task-semantic node, whose effect is larger in the hierarchical graph than in the flat one (Section 5.4). This is consistent with the cluster layer acting mainly as the structure through which task conditioning operates, rather than as an independent source of large gains.

The task-node ablation provides further evidence that a relevant task representation contributes to the final document representation. Relative to the hierarchical configuration without a task node, the correct description produces gains of 5.7, 2.4, and 4.5 macro-F1 points on Claim, Specificity, and Commitment, respectively. A mismatched ESG description produces much smaller gains, while an unrelated description and a fixed random vector, including its norm-matched version, perform slightly below the configuration without a task node. Together, these controls indicate that the benefit of the task-semantic node depends on the content of the task representation rather than on the presence of an additional graph node. Because this representation is a fixed mean of static embeddings, the effect should be understood as task conditioning rather than as instruction following.

The embedding experiment shows that the choice of static embedding also affects performance. FastText-300D, used as the main embedding configuration, improves macro-F1 over GloVe-300D by 0.5, 0.8, and 0.9 percentage points for Claim, Specificity, and Commitment, respectively. Although this effect is consistent, it is smaller than the improvements observed from the principal architectural ablations, suggesting that the hierarchical and task-conditioning components play an important role in the observed performance.

The clustering experiments further indicate that the model is reasonably robust to the choice of clustering method. K-Means provides the strongest overall results, while Agglomerative and Spectral clustering remain competitive. The lower performance of DBSCAN on Claim and Commitment suggests that density-based grouping may be less suitable for constructing stable semantic clusters in the present embedding space.

The main comparison also highlights an important limitation. TC-HGAT obtains the highest mean macro-F1 on Claim and Commitment but not on Specificity, where it is 1.6 points below BERT-base, 0.8 points below ClimateBERT, and 0.2 points below BertGCN. Several results indicate that Specificity is the task on which the proposed components help least: the task-semantic node improves macro-F1 by only 2.4 points, compared with 5.7 and 4.5 points on Claim and Commitment, and once the node is excluded from the readout, the remaining margin over the configuration without a task node is only 0.9 points. Several properties of the current design plausibly contribute to this gap. First, specificity

depends heavily on numerals, dates, and named programs or entities (for example, a 30% reduction by 2030 versus a commitment to significant reductions). Static embeddings represent such tokens poorly: GloVe has no vectors for most unseen numeric strings and names, and FastText composes them from character *n*-grams, which encode surface form rather than meaning or role in the sentence. Second, clustering is performed in this static embedding space, so numeric and temporal tokens may be grouped with one another rather than with the words they qualify, and mean pooling at the cluster and graph levels further dilutes the few tokens that make a statement specific. Third, the specific–vague distinction often depends on the relation between a quantity and its referent (what is reduced, by how much, and by when), which contextual encoders model directly but which our graph, having no word–word edges, represents only through shared cluster membership. Finally, this task has the most heterogeneous notion of the positive class: the Detailed description lists actions, targets, quantities, dates, and measurements, whereas the Claim and Commitment descriptions are organized around a single linguistic act. The per-class results in Table 5 confirm that the gap comes mainly from missed specific statements: TC-HGAT matches the Transformer baselines on non-specific statements (82.0% F1) but trails them on specific ones (73.6% versus 75.1% and 76.2%). Contextual representations may therefore remain advantageous when classification depends on fine-grained distinctions between otherwise similar statements.

A qualitative inspection of the Specificity test predictions supports this interpretation. Several specific statements that TC-HGAT misclassifies as non-specific are classified correctly by ClimateBERT, but only with low confidence ($p < 0.70$), which indicates that they are borderline cases even for a contextual model. Most of these statements contain no explicit ESG quantity; their specificity comes mainly from the names of concrete programs and entities, such as Pronaf and the ABC Low Carbon Agriculture Program, a named farm project, or a regulatory authority and its decision dates, or from the description of a concrete industrial process (battery recycling to recover nickel, cobalt, and lithium). Such names are rare or absent from the FastText vocabulary, so their vectors are composed from character *n*-grams. These vectors reflect the surface form of the name rather than what the program is, and, being static, they do not change with the surrounding context. Mean pooling at the cluster and graph levels further dilutes these few informative tokens. A contextual encoder such as ClimateBERT can instead infer from the surrounding words (e.g., *financial facilities*, *program*, *approved*) that a specific, identifiable initiative is being described, which allows it to classify these statements correctly, albeit with low confidence.

Overall, the experiments support the combination of hierarchical graph aggregation and task-semantic conditioning as a lightweight approach to ESG text classification. At the same time, the results indicate that the proposed architecture should not be viewed as a universal replacement for contextual Transformer models, particularly for tasks requiring fine-grained semantic discrimination.

## 6. Limitations

Although the proposed TC-HGAT framework shows strong results and a relatively small computational footprint, the current study has several limitations.

First, TC-HGAT relies on static 300-dimensional embeddings, namely GloVe and FastText [29,30]. Although FastText provides subword information that can improve the representation of unseen or rare words, both embedding families lack the contextual adaptation of Transformer-based language models. Consequently, the representation of a word does not change according to its surrounding document context before graph message passing.

Second, the task-semantic representation is constructed by mean-pooling static word embeddings and is fixed for all documents belonging to the same classification task. The task-semantic node therefore captures task-level semantic information but does not model contextual interactions between individual description tokens and document tokens. For this reason, the mechanism should be understood as task conditioning rather than as instruction following in the sense used for large language models. The paraphrase experiment shows that the gain is stable under the evaluated rewordings (within 0.7 macro-F1 points). However, it does not cover contextual encoders of the task description, and because mean-pooled paraphrases remain highly similar (average cosine similarity 0.94), it tests robustness to wording more than to substantially different formulations of the task.

Third, the usefulness of task conditioning depends on the task description itself. The task-node ablation shows that the correct description provides stronger performance than mismatched or unrelated descriptions. Poorly designed or ambiguous descriptions may therefore reduce the benefit of the task-semantic node.

Fourth, the clustering strategy is document-level. The word embeddings of each document are clustered independently, with a requested maximum of $K$ clusters and only observed, non-empty clusters instantiated in the graph. While this design allows the intermediate semantic structure to adapt to individual documents, it may produce different cluster organizations across documents and may be less stable for very short or highly heterogeneous texts. Future work could investigate globally consistent, adaptive, or hierarchical clustering strategies.

Fifth, the evaluation is based on three ESG classification tasks derived from corporate sustainability text. These datasets are small (1320–1700 sentence-level instances per task). Specificity, and Commitment classification, the extent to which the observed results generalize to larger datasets, longer documents, other ESG datasets, financial domains, or general NLP tasks remains uncertain.

Sixth, the comparison with TextGCN and BertGCN combines differences in architecture with differences in graph construction, because these baselines use corpus-level graphs whereas TC-HGAT uses per-document graphs. The controlled evidence for our design choices comes from the comparisons with Word-GCN and Hierarchical-GAT, which share the same per-document construction.

Finally, the principal experiments use three random seeds (13, 42, and 77). The paired-bootstrap tests in Section 5.2 account for the sampling variation of the test instances, and the standard deviations indicate variability across seeds, but additional seeds would provide stronger evidence regarding the reliability of the observed differences between models. Future work should therefore evaluate the framework across a larger number of independent runs.

## 7. Conclusions and Future Work

In this paper, we introduced TC-HGAT, a task-conditioned hierarchical graph attention network for ESG text classification. TC-HGAT combines static word embeddings with a hierarchical graph representation containing word nodes, virtual semantic cluster nodes, and a task-semantic node obtained by mean-pooling the static embeddings of a task description. This design introduces an intermediate level of semantic aggregation while incorporating a fixed task-level representation into the graph.

Extensive experiments across Claim, Specificity, and Commitment classification show that TC-HGAT achieves strong predictive performance with a substantially smaller number of trainable parameters than the large Transformer-based baselines. Using the main FastText-300D configuration, the model obtains macro-F1 scores of 94.6%, 77.8%, and 79.3% on Claim, Specificity, and Commitment, respectively. TC-HGAT obtains the highest mean

macro-F1 among the evaluated models on Claim and Commitment, although its differences from ClimateBERT are not statistically significant, while BERT-base and ClimateBERT achieve higher macro-F1 on Specificity. These results indicate that the proposed lightweight architecture can provide competitive performance without requiring a large contextual Transformer encoder.

The ablation experiments further clarify the contribution of the individual components. The task-semantic node is the main source of the improvement: the correct task description improves macro-F1 over the hierarchical configuration without a task node by 5.7, 2.4, and 4.5 points, significantly on all three tasks. Virtual cluster nodes alone give a modest and task-dependent improvement over the flat graph that is statistically significant only on Claim. The control conditions, including fixed and norm-matched random vectors placed in the same graph position, show that performance depends on the content of the task representation rather than on the presence or scale of an additional graph node. Sensitivity experiments across embedding families, clustering strategies, levels of task-description detail, and cluster numbers further demonstrate the behavior of the proposed architecture under different configurations.

From a computational perspective, TC-HGAT uses approximately 1.4 million trainable parameters, excluding the frozen embedding table, and approximately 0.3–0.7 GB of peak GPU memory during training under the reported experimental setup. Its end-to-end inference time, including per-document clustering, is approximately 5–12 ms per document, which is comparable to the Transformer baselines rather than faster. The proposed architecture therefore provides a favorable trade-off between accuracy and training-time GPU memory, at the cost of CPU-side embedding storage and clustering.

Overall, the results suggest that hierarchical semantic organization and task-semantic conditioning provide a useful lightweight alternative for ESG text classification. At the same time, the Specificity results and the limitations of static embeddings and fixed task-level representations indicate that contextual and more adaptive representations remain important directions for future research.

*Future Work*

Several directions can be explored in future work. First, the document-level clustering mechanism could be extended with adaptive or hierarchical strategies that adjust the semantic organization to the content and length of individual documents while improving the stability of the resulting clusters.

Second, the task representation could be extended beyond mean-pooled static embeddings. Learned task embeddings, contextual encoders of the task description, or document-adaptive task representations could provide richer task representations while preserving the lightweight graph architecture.

Third, future experiments should evaluate parameter-efficient Transformer baselines, such as adapters or LoRA-based fine-tuning, to provide a stronger comparison between lightweight graph architectures and computationally efficient contextual models.

Fourth, robustness to the formulation of the task description could be studied with more substantially different formulations of the same task, such as question-style or keyword-based descriptions, and with contextual encoders.

Fifth, the framework should be evaluated on longer documents, such as full sustainability reports, where the intermediate cluster layer and the over-squashing motivation are expected to matter more than for sentence-level instances.

Finally, broader evaluations across additional ESG datasets, domains, and larger numbers of random seeds would provide stronger evidence regarding the generalizability and statistical reliability of the proposed framework.

**Author Contributions:** Conceptualization, Y.E.; Methodology, Y.E. and A.S.; Software, Y.E. and A.S.; Formal Analysis, M.C.; Investigation, M.C.; Data Curation, Y.E., A.S., M.C., M.E.K., and M.O.; Writing—Original Draft Preparation, Y.E., A.S., M.C., M.E.K., and M.O.; Writing—Review and Editing, Y.E. and M.C.; Supervision, M.C.; Project Administration, M.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The ClimateBERT ESG datasets used in this study are publicly available from their original authors under their respective license terms. The procedure used to derive the training and validation partitions from the official training splits is described in Section 4.1, and the preprocessing steps, clustering settings, hyperparameters, baseline configurations, and software versions required to reproduce the experiments are reported in Sections 3.2, 4.2, 4.3, and 4.5 and in Table 2 and Appendix B. The implementation code is not publicly available.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A. Task Descriptions

This appendix reports the task descriptions used by TC-HGAT and in the corresponding task-node ablation experiments. They are provided to clarify the semantic information supplied to the task-semantic node and to facilitate reproducibility. Each description is converted into a single fixed vector by mean-pooling the static embeddings of its tokens (Section 3.4); the imperative wording is therefore not interpreted as an instruction by the model.

### *Appendix A.1. Task-Specific Descriptions*

TC-HGAT uses a task-specific textual description to construct the task-semantic node. The description is fixed for all documents within the same classification task. The Detailed level constitutes the default configuration used for the main TC-HGAT experiments reported in the paper. The descriptions are listed in Table A1.

**Table A1.** Task descriptions used by the main TC-HGAT configuration.

| Task | Detailed Task Description |
| --- | --- |
| Claim | "Determine whether the text contains a claim related to climate change, environmental impact, greenhouse gas emissions, sustainability, or climate-related action. Focus on whether the organization makes an explicit statement or assertion concerning an ESG-related issue." |
| Specificity | "Determine whether the text provides specific and concrete information about an ESG-related action, target, quantity, date, measurement, or measurable outcome. Look for concrete details such as numerical values, dates, quantified targets, measurable indicators, or clearly defined actions." |
| Commitment | "Determine whether the text expresses an ESG-related commitment, intention, target, planned action, or future-oriented objective by an organization. Focus on statements describing what the organization intends, plans, targets, or commits to do." |

### *Appendix A.2. Levels of Task-Description Detail*

To evaluate the effect of the level of detail, four levels of task description were considered: Minimal, Standard, Detailed, and Expert. The underlying classification objective remains unchanged across all conditions. The Detailed condition is the default configuration used by TC-HGAT in the main experiments, whereas the other levels serve as

controlled alternatives (Section 5.9). The four levels are listed in Tables A2–A4 for the three tasks.

### Appendix A.2.1. Claim Classification

**Table A2.** Task-description levels for Claim classification.

| Level | Task Description |
|---|---|
| Minimal | "Classify whether the text contains a climate-related claim." |
| Standard | "Determine whether the text contains a claim related to climate change, environmental impact, greenhouse gas emissions, sustainability, or climate-related action." |
| **Detailed (default)** | "Determine whether the text contains a claim related to climate change, environmental impact, greenhouse gas emissions, sustainability, or climate-related action. Focus on whether the organization makes an explicit statement or assertion concerning an ESG-related issue." |
| Expert | "Determine whether the text contains an explicit claim related to climate change, environmental impact, greenhouse gas emissions, sustainability, or climate-related action. Consider statements describing environmental impacts, emissions, sustainability performance, climate-related actions, or organizational practices. Distinguish explicit claims from simple mentions of ESG-related topics without an actual assertion." |

Bold indicates the default level used in the main experiments.

### Appendix A.2.2. Specificity Classification

**Table A3.** Task-description levels for Specificity classification.

| Level | Task Description |
|---|---|
| Minimal | "Determine whether the text contains specific information." |
| Standard | "Determine whether the text provides specific and concrete information about an ESG-related action, target, quantity, date, measurement, or measurable outcome." |
| **Detailed (default)** | "Determine whether the text provides specific and concrete information about an ESG-related action, target, quantity, date, measurement, or measurable outcome. Look for concrete details such as numerical values, dates, quantified targets, measurable indicators, or clearly defined actions." |
| Expert | "Determine whether the text provides specific and concrete information about an ESG-related action, target, quantity, date, measurement, or measurable outcome. Identify verifiable details such as quantities, dates, numerical targets, measurements, indicators, or clearly defined actions rather than general or vague statements about sustainability." |

Bold indicates the default level used in the main experiments.

### Appendix A.2.3. Commitment Classification

**Table A4.** Task-description levels for Commitment classification.

| Level | Task Description |
|---|---|
| Minimal | "Determine whether the text expresses an ESG commitment." |
| Standard | "Determine whether the text expresses an ESG-related commitment, intention, target, planned action, or future-oriented objective by an organization." |
| **Detailed (default)** | "Determine whether the text expresses an ESG-related commitment, intention, target, planned action, or future-oriented objective by an organization. Focus on statements describing what the organization intends, plans, targets, or commits to do." |
| Expert | "Determine whether the text expresses an ESG-related commitment, intention, target, planned action, or future-oriented objective by an organization. Identify forward-looking statements indicating an intention or commitment to undertake an ESG-related action, achieve a target, improve performance, or change organizational practices. Distinguish future-oriented commitments from descriptions of actions that have already occurred." |

Bold indicates the default level used in the main experiments.

## *Appendix A.3. Paraphrased Task Descriptions*

Each Detailed description was rewritten in three paraphrases that keep the task definition and the listed criteria while changing word choice and sentence structure. These paraphrases, listed in Table A5, are used in Section 5.7.

**Table A5.** Paraphrases of the Detailed task descriptions.

| Task | | Paraphrase |
|---|---|---|
| Claim | P1 | "Decide if the passage asserts something about climate change, environmental effects, greenhouse gas emissions, sustainability, or actions taken on climate. Check whether the company explicitly states a position or fact on an ESG topic." |
| | P2 | "Identify whether this statement makes an explicit assertion by the organization about an ESG matter, such as climate change, its environmental footprint, emissions of greenhouse gases, sustainability, or climate action." |

**Table A5.** *Cont.*

| Task | | Paraphrase |
|---|---|---|
| | P3 | "Judge whether the company in this text claims something regarding the climate, the environment, greenhouse gas emissions, sustainability, or measures against climate change, as opposed to only mentioning these subjects." |
| Specificity | P1 | "Decide if the passage gives precise and concrete details about an ESG action, goal, amount, deadline, metric, or measurable result, such as numbers, dates, quantified goals, indicators, or clearly described measures." |
| | P2 | "Identify whether this statement contains concrete ESG information, for example figures, time frames, quantified objectives, measurable indicators, or well-defined actions, rather than general wording." |
| | P3 | "Judge whether the text reports specific facts about an ESG-related activity or target, including values, years, quantities, measurements, outcomes that can be measured, or clearly defined steps." |
| Commitment | P1 | "Decide if the passage states that the organization commits to, intends, plans, or targets an ESG-related action or objective in the future." |
| | P2 | "Identify whether this statement describes what the company will do, aims to achieve, or has pledged regarding an ESG goal, target, or planned measure." |
| | P3 | "Judge whether the text expresses a forward-looking ESG intention of the organization, such as a pledge, a planned initiative, a target to reach, or an objective it has committed to pursue." |

### *Appendix A.4. Task-Node Ablation Conditions*

The task-node ablation evaluates whether performance gains depend on providing task-relevant semantic information. Six conditions are considered: *None*, *Correct*, *Wrong ESG Task*, *Unrelated*, *Fixed random*, and *Norm-matched random*. The corresponding experimental results are reported in Section 5.5, and the conditions are described in Table A6.

**Table A6.** Conditions used in the task-node ablation.

| Condition | Task-Node Configuration |
|---|---|
| None | No task-semantic node is included in the graph. |
| Correct | The Detailed task description corresponding to the target classification task is used. These descriptions are reported in Table A1. |
| Wrong ESG Task | The description of a different ESG classification task is provided instead of the description of the target task. For example, the Commitment description is supplied when performing Claim classification. |
| Unrelated | A non-ESG financial-news description is provided instead of the task-specific ESG description: "The central bank announced that it would maintain its benchmark interest rate, citing persistent inflationary pressures and uncertainty surrounding the global economic outlook." |
| Fixed random | A random 300-dimensional vector, drawn once and kept frozen, is placed in the position of the task-semantic node; no text is used. Its distribution, seed, and norm are given in Section 4.4. |
| Norm-matched random | The fixed random vector rescaled to the L2 norm of $h_I$ of the target task. |

The purpose of these controls is to distinguish task-relevant semantic information from the mere presence of an additional node. In the main experiments, the *Correct* condition corresponds to the Detailed descriptions reported in Table A1.

## Appendix B. Validation-Tuned Hyperparameters

Table A7 lists the hyperparameters selected on the validation partitions for each neural model.

**Table A7.** Hyperparameters selected on the validation partitions for each neural model (Section 5.3). The same configuration was selected for all three tasks.

| Model | Learning Rate | Dropout | Epochs |
|---|---|---|---|
| TinyBERT | $5 \times 10^{-5}$ | 0.1 | 30 |
| BERT-base | $2 \times 10^{-5}$ | 0.1 | 15 |
| ClimateBERT | $2 \times 10^{-5}$ | 0.1 | 15 |
| Word-GCN | $10^{-3}$ | 0.5 | 30 |
| TextGCN | $10^{-3}$ | 0.5 | 30 |
| BertGCN | $5 \times 10^{-5}$ | 0.3 | 15 |
| Hierarchical-GAT | $5 \times 10^{-4}$ | 0.5 | 30 |
| TC-HGAT | $5 \times 10^{-4}$ | 0.5 | 30 |

## References


1. Luccioni, A.; Baylor, E.; Duchene, N. Analyzing sustainability reports using natural language processing. *arXiv* **2020**, arXiv:2011.08073.
2. Friederich, D.; Kaack, L.H.; Luccioni, A.; Steffen, B. Automated identification of climate risk disclosures in annual corporate reports. *arXiv* **2021**, arXiv:2108.01415.
3. Bingler, J.A.; Kraus, M.; Leippold, M.; Webersinke, N. How cheap talk in climate disclosures relates to climate initiatives, corporate emissions, and reputation risk. *J. Bank. Finance* **2024**, *164*, 107191.
4. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the NAACL-HLT, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
5. 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 (NeurIPS)*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 5998–6008.
6. Webersinke, N.; Kraus, M.; Bingler, J.A.; Leippold, M. ClimateBERT: A pretrained language model for climate-related text. *arXiv* **2021**, arXiv:2110.12010.
7. Araci, D. FinBERT: Financial sentiment analysis with pre-trained language models. *arXiv* **2019**, arXiv:1908.10063.
8. Yao, L.; Mao, C.; Luo, Y. Graph convolutional networks for text classification. *Proc. AAAI Conf. Artif. Intell.* **2019**, *33*, 7370–7377.
9. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
10. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 1024–1034.
11. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural message passing for quantum chemistry. In Proceedings of the International Conference on Machine Learning (ICML), Sydney, NSW, Australia, 6–11 August 2017; pp. 1263–1272.
12. Skianis, K.; Malliaros, F.D.; Vazirgiannis, M. Fusing document, collection and label graph-based representations with word embeddings for text classification. In Proceedings of the 12th Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12), New Orleans, LA, USA, 6 June 2018; pp. 49–58.
13. Lin, Y.; Meng, Y.; Sun, X.; Han, Q.; Kuang, K.; Li, J.; Wu, F. BertGCN: Transductive text classification by combining GNN and BERT. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 1456–1462.
14. Bugueño, M.; de Melo, G. Connecting the dots: What graph-based text representations work best for text classification using graph neural networks? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Association for Computational Linguistics: Stroudsburg, PA, USA, 2023; pp. 8943–8960.
15. Alon, U.; Yahav, E. On the bottleneck of graph neural networks and its practical implications. In Proceedings of the International Conference on Learning Representations (ICLR), Online, 3–7 May 2021.

16. Topping, J.; Giovanni, F.D.; Chamberlain, B.P.; Dong, X.; Bronstein, M.M. Understanding over-squashing and bottlenecks on graphs via curvature. In Proceedings of the International Conference on Learning Representations (ICLR), Online, 25–29 April 2022.
17. Wang, K.; Ding, Y.; Han, S.C. Graph neural networks for text classification: A survey. *Artif. Intell. Rev.* **2024**, *57*, 190.
18. Huang, L.; Ma, D.; Li, S.; Zhang, X.; Wang, H. Text level graph neural network for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 3444–3450.
19. 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 58th Annual Meeting of the Association for Computational Linguistics (ACL)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 334–339.
20. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph attention networks. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
21. Brody, S.; Alon, U.; Yahav, E. How attentive are graph attention networks? In Proceedings of the International Conference on Learning Representations (ICLR), Online, 25–29 April 2022.
22. Sun, Z.; Yang, P.; Shi, L.; Yao, Q.; Li, L.; Xiao, H.; Zheng, X.; Shen, X. Benchmarking backbone neural networks for HVAC fault diagnosis and regression: Insights into spatiotemporal, graph, and attention-based architectures. *Energy Build.* **2026**, *366*, 117718. https://doi.org/10.1016/j.enbuild.2026.117718.
23. Varini, F.S.; Boyd-Graber, J.; Ciaramita, M.; Leippold, M. ClimaText: A dataset for climate change topic detection. *arXiv* **2020**, arXiv:2012.00483.
24. Schimanski, T.; Bingler, J.; Kraus, M.; Hyslop, C.; Leippold, M. ClimateBERT-NetZero: Detecting and assessing net zero and reduction targets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2023; pp. 15745–15756.
25. Schimanski, T.; Reding, A.; Reding, N.; Bingler, J.; Kraus, M.; Leippold, M. Bridging the gap in ESG measurement: Using NLP to quantify environmental, social, and governance communication. *Financ. Res. Lett.* **2024**, *61*, 104979.
26. Ying, R.; You, J.; Morris, C.; Ren, X.; Hamilton, W.L.; Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*; Curran Associates Inc.: Red Hook, NY, USA, 2018; pp. 4800–4810.
27. Lee, J.; Lee, I.; Kang, J. Self-attention graph pooling. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019; pp. 3734–3743.
28. Zhang, Z.; Bu, J.; Ester, M.; Zhang, J.; Yao, C.; Yu, Z.; Wang, C. Hierarchical graph pooling with structure learning. *arXiv* **2019**, arXiv:1911.05954.
29. Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2014; pp. 1532–1543.
30. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146.
31. Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; Liu, Q. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 4163–4174.
32. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2017; pp. 427–431.
33. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf.Theory* **1982**, *28*, 129–137.
34. Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR, USA, 2–4 August 1996; pp. 226–231.
35. von Luxburg, U. A tutorial on spectral clustering. *Stat. Comput.* **2007**, *17*, 395–416.
36. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
37. Fey, M.; Lenssen, J.E. Fast graph representation learning with PyTorch Geometric. In Proceedings of the ICLR Workshop on Representation Learning on Graphs and Manifolds, New Orleans, LA, USA, 6–9 May 2019.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.