Severity-Aware Hierarchical Classification on DBPedia: A Comparative Study of TF-IDF and BERT with Graph Attention Networks

Shikhar Dave
IIT Jodhpur
b22ch032@iitj.ac.in

Mahek Vanjani
IIT Jodhpur
b22ee088@iitj.ac.in

Abstract

Automated classification of textual data into large-scale taxonomies is a challenging task in AI. Knowledge graphs such as DBPedia organize entities into multi-level hierarchies, but traditional classification methods often treat all misclassifications equally, overlooking the varying severity of errors across hierarchy levels. For instance, misclassifying a 'Politician' as an 'Album' is far more detrimental than misclassifying closely related sub-categories. To address this, we introduce Severity-Aware Hierarchical Multilabel Classification (SA-HMC), which accounts for the semantic severity of classification errors.

We propose two architectures for SA-HMC applied to the DBPedia dataset, both incorporating a severity-aware loss function that penalizes misclassifications based on their severity within the hierarchy.

Model I integrates BERT's deep contextual representation with hierarchy-aware label embeddings derived from a Graph Attention Network (GAT) operating on the label graph. BERT-embedded label names serve as initial features, which are refined through cross-attention mechanisms. This model achieved a test F1-Macro score of 0.7564 and a test Subset Accuracy of 0.8015.

Model II combines traditional TF-IDF features (sparse vectors up to 20,000 dimensions) processed by a Multi-Layer Perceptron (MLP) for document representation, with a GAT that operates on learnable node embeddings initialized randomly for each label. The GAT refines these embeddings based on the label graph structure, and classification logits are computed via a dot product between the document and refined label embeddings. This model achieved a higher peak validation F1-Macro of 0.8446 and converged faster.

Both models demonstrate the efficacy of severity-aware loss functions in hierarchical classification, with potential applications in knowledge base curation, semantic search, content recommendation, and information retrieval. While Model I benefits from the semantic power of BERT, Model II highlights the efficiency of using graph-based processing with simpler text features, offering a potentially more scalable solution.

1 Introduction

Hierarchical multi-label classification (HMC) is a critical challenge in natural language processing, where documents are annotated with multiple labels organized in a structured taxonomy. Unlike flat classification, HMC demands that models respect the inherent dependencies among labels, often structured as trees or directed acyclic graphs. The complexity increases further when the cost of misclassification varies depending on the position in the hierarchy.

In real-world applications such as medical coding, legal document tagging, or knowledge base population, the consequences of such misclassifications can differ significantly depending on their hierarchical distance from the correct label. Traditional loss functions like binary cross-entropy or flat multi-label loss treat all mistakes equally, disregarding this nuanced structure.

To bridge this gap, we explore severity-aware hierarchical classification, where the loss is adapted to penalize misclassifications proportional to their divergence from the true path in a label hierarchy. We propose two distinct yet complementary methods: (1) a fine-tuned BERT-based architecture that captures contextual semantics while maintaining efficiency through selective layer freezing and integrating cross-attention mechanism along with Graph Attention Networks(GAT) and (2) a graph-based approach using TF-IDF representations coupled with Graph Attention Networks (GAT) operating over learnable label embeddings.

1.1 Dataset & Knowledge Graph formation

We evaluate our models on the **DBpedia Ontology Dataset**, a standard benchmark for hierarchical multi-label and multi-class text classification. DBpedia (from 'DB" for database") is a project that extracts structured content from Wikipedia, aiming to convert unstructured encyclopedia entries into a rich knowledge base of facts. The dataset comprises **240,942** cleaned Wikipedia abstracts, each annotated with a path in a **three-level class hierarchy**:

1. L1 (9 categories): Broad categories such as *Company*, *Person*, *Place*

2. L2 (70 categories): Subcategories like *Airline*, *Artist*, *Building*

3. L3 (219 categories): Fine-grained types including *Low-cost Airline*, *Sculptor*, *Skyscraper*

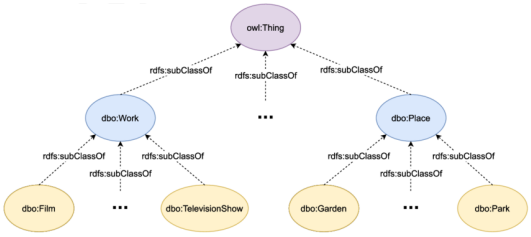

Figure 1: Hierarchy Tree of DBpedia Ontology

1.2 Dataset preprocessing steps

We begin by preprocessing the DBPedia dataset to support both textual feature extraction and hierarchical multilabel classification. The dataset consists of 240,942 samples, each with a description field and three hierarchical class levels: `l1`, `l2`, and `l3`. After loading the data, missing values were handled, with emphasis on the `text` column.

The raw text was cleaned through a pipeline comprising lowercasing, removal of URLs, HTML tags, digits, and punctuation, followed by tokenization using `nltk.word_tokenize`. Stopwords were removed using NLTK's corpus, and lemmatization was performed via `WordNetLemmatizer`. The final processed version of each description was stored in a new `processed_text` column.

For label handling, we collected all unique non-null labels from the three hierarchy columns, obtaining 298 distinct classes. These were mapped to unique indices via dictionaries `label_to_index` and `index_to_label`. Each sample's set of valid labels was converted into a multi-hot encoded vector using scikit-learn's `MultiLabelBinarizer`, resulting in a binary matrix $y \in \{0,1\}^{N \times C}$. This matrix, along with the tokenized corpus and mapping dictionaries, was saved for subsequent training phases.

1.3 Knowledge Graph Construction

A networkx.DiGraph object (G)(with 298 nodes and 289 edges) was created to explicitly represent the label hierarchy. Nodes were added for each unique label string from the all_labels_list.

The directed edges l1 -> l2 and l2 -> l3 were added based on the unique hierarchical combinations extracted in the above illustrated steps, representing the parent-child relationships. Node attributes,

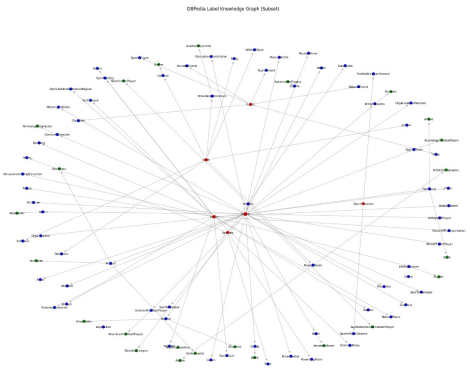

Figure 2: DBpedia Knowledge Graph

specifically the 'level,' were added using the label_hierarchy_info. The constructed graph G was saved to dbpedia_label_hierarchy.gpickle using pickle.This comprehensive preprocessing pipeline resulted in clean text data, multi-hot encoded target labels, essential mappings, hierarchy information crucial for severity calculation, and a graph representation of the label structure, providing the necessary inputs for the subsequent modeling stages.

2 Problem Statement

We address the task of **Severity-Aware Hierarchical Multilabel Classification (SA-HMC)** using the DBPedia dataset, where the goal is to assign one or more ontological labels to a given entity description drawn from a three-level hierarchy (`l1`, `l2`, `l3`). Standard hierarchical classification approaches often optimize for flat accuracy, implicitly treating all errors as equally undesirable. In contrast, our formulation incorporates a severity metric based on the hierarchical distance between predicted and ground-truth labels, acknowledging that misclassifying an entity across top-level branches (e.g., 'Film" $\rightarrow$ Place") is substantially more detrimental than confusion within the same sub-hierarchy.

This severity-aware perspective is crucial for applications involving knowledge base enrichment, where hierarchical misalignments can propagate semantic inconsistencies across downstream systems. Our objective is to develop a classification model that not only achieves strong performance on traditional multilabel metrics but also **explicitly minimizes high-severity errors**, thus producing semantically coherent and ontologically faithful predictions.

3 Literature Review

This project intersects several key research areas within Natural Language Processing (NLP) and Machine Learning (ML), primarily focusing on Hierarchical Text Classification (HTC), Knowledge Graph (KG) utilization, advanced text representation techniques, and cost/severity-sensitive learning.

3.1 Hierarchical Text Classification (HTC)

Classifying text documents into a predefined hierarchy of categories is a long-standing challenge known as hierarchical text classification (1). Unlike flat classification, where categories are independent, HTC aims to leverage parent-child or other relationships between labels (e.g., the ontological structure in DBPedia) to improve classification accuracy and consistency. Methodologies broadly fall into two categories:

- **Flat Approaches:** These methods ignore the hierarchy during training, treating each label independently or transforming the hierarchy into a single layer of leaf nodes. Although simpler, they fail to utilize the rich structural information.
- **Hierarchical Approaches:** These explicitly incorporate the hierarchy. *Local classifiers* train independent models for each node or level (e.g., predicting L1 first, then L2 given L1),

often suffering from error propagation. *Global classifiers* attempt to predict labels across the entire hierarchy simultaneously using a single, more complex model. These global methods often employ techniques like hierarchy-based regularization, structured output learning, or embedding methods that encode hierarchical relationships. **Our work falls under the global approach, aiming to learn representations informed by the hierarchy.**

HTC often involves *multilabel classification*, as documents (like DBPedia entities) can belong to multiple categories simultaneously (e.g., a specific class and all its ancestors). Standard multilabel metrics like F1 variants (micro, macro, samples) and Hamming Loss are typically used for evaluation (2).

3.2 Knowledge Graph Integration for Text Classification

Knowledge graphs (KGs), such as DBPedia, WordNet, or YAGO, store structured information about entities and their relationships. Integrating this external knowledge has shown significant potential for enhancing various NLP tasks, including text classification. Common approaches include:

- **KG Embeddings:** Learning low-dimensional vector representations of entities and relations in the KG (e.g., using techniques like TransE, RDF2Vec, and RotatE) and incorporating these embeddings as features into the classification model (3; 4).
- **Entity Linking and Graph Features:** Identifying entities mentioned in the text, linking them to the KG, and extracting features from their KG neighbourhood or properties.
- **Graph Neural Networks (GNNs):** Applying GNNs directly to relevant graph structures. This can involve GNNs on graphs constructed from entities within the text or, more relevant to this project, GNNs operating on the label hierarchy graph itself. Processing the label graph allows the model to learn context-aware representations for each label, informed by its ancestors, descendants, and siblings (5; 6; 7). Common GNN architectures used include Graph Convolutional Networks (GCN) (8) and Graph Attention Networks (GAT) (9), the latter allowing nodes to weigh the importance of their neighbors. Our initial architectural designs were heavily considered using GAT in the extracted DBPedia label hierarchy.

3.3 Text Representation with Transformers

The advent of deep contextual language models, particularly Transformers and BERT (Bidirectional Encoder Representations from Transformers) (10), has revolutionized NLP. BERT models are pre-trained on massive text corpora using self-supervised objectives (like Masked Language Modeling), enabling them to learn rich, context-dependent representations of words and sentences. Fine-tuning a pre-trained BERT model by adding a classification head and training end-to-end on a specific downstream task has become a standard and highly effective approach for text classification, often achieving state-of-the-art results. One of the architectures explored in this project leverages a fine-tuned BERT model (`bert-base-uncased`) as the primary text encoder.

3.4 Severity and Cost-Sensitive Learning

Traditional classification metrics often treat all misclassification errors equally. However, in many real-world scenarios, different types of errors incur different costs or have varying degrees of severity. Cost-sensitive learning explicitly addresses this by incorporating non-uniform misclassification costs into the learning process. This can be achieved through data resampling, threshold adjustment, or, more relevantly here, by designing cost-sensitive loss functions (11). While widely studied in fields like medical diagnosis or fraud detection, applying cost-sensitivity based specifically on the semantic or structural distance within a *label hierarchy* for text classification is a more niche but important area. Defining severity based on how far a misclassification deviates from the true path in the ontology (e.g., crossing L1 vs. staying within L2) provides a principled way to guide the model towards more meaningful predictions in hierarchical contexts.

4 Proposed Method

To address the problem of Severity-Aware Hierarchical Multilabel Classification (SA-HMC) on the DBPedia dataset, we implemented and evaluated two distinct model architectures, both incorporating a custom severity-aware loss function. This section details the data preprocessing pipeline common to both models, followed by the specific architectures (Model I: BERT + GAT + Cross-Attention based, Model II: TF-IDF + GNN) and the severity-aware loss mechanism.

4.1 Data Preprocessing and Feature Engineering

A standardized preprocessing pipeline was applied to the raw DBPedia dataset (`DBPEDIA_train.csv`), containing entity descriptions (text) and hierarchical labels (l1, l2, l3).

1. **Text Cleaning:** Raw text underwent normalization including lowercasing and the removal of URLs, HTML tags, digits, and punctuation. NLTK-based tokenization, English stopword removal, and WordNet lemmatization were subsequently applied to generate cleaned text sequences.
2. **Label Vocabulary and Mapping:** All unique, non-null label strings across the `l1`, `l2`, and `l3` columns were extracted, resulting in a vocabulary of $N_L = 298$ unique labels. Deterministic mappings between label strings and integer indices $[0, N_L - 1]$ were generated and stored.
3. **Hierarchy Representation:**
   - **Path Information:** Unique $(\texttt{l1}, \texttt{l2}, \texttt{l3})$ label paths were extracted to capture the hierarchical structure. A lookup table (`label_hierarchy_info`) was built, mapping each label index to its string label, its ancestors, and its inferred hierarchical level (1, 2, or 3). This structure is vital for severity computation.
   - **Graph Construction:** A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ was constructed using NetworkX, where $\mathcal{V}$ is the set of N_L unique labels, and $\mathcal{E}$ consists of edges such as $(\texttt{l1}, \texttt{l2})$ and $(\texttt{l2}, \texttt{l3})$. Node attributes capturing hierarchical level information were added. The graph was serialized using `pickle`.
4. **Target Vector Generation:** For each sample i, the applicable label set $Y_i = \{l_1^{(i)}, l_2^{(i)}, l_3^{(i)}\} \cap \mathcal{V}$ was identified. A `MultiLabelBinarizer`, fitted on the full ordered label vocabulary, transformed Y_i into a binary multi-hot target vector $\mathbf{y}_i \in \{0, 1\}^{N_L}$.
5. **Data Splitting:** The processed data $X = \{x_1, \ldots, x_M\}$ and corresponding targets $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$ were split into training (80%), validation (10%), and test (10%) sets using stratified sampling adapted for multilabel data, or random splits when necessary. A fixed random seed ensured reproducibility.

4.2 Model I: Severity-Aware BERT with Graph Attention and Fusion

The first proposed architecture, designated Model I, integrates deep contextual text representations derived from a pre-trained Transformer with hierarchy-aware label representations learned via a Graph Attention Network (GAT), employing cross-modal attention for fusion before final classification. The entire model is trained end-to-end using the severity-aware loss function.

1. **Text Encoder (Fine-tuned BERT):** The core text representation is generated using the `bert-base-uncased` model (10). Input text sequences x_i are tokenized using the corresponding `BertTokenizer`, padded or truncated to a fixed maximum length $L_{\text{max}} = 256$. The tokenized sequence is processed by the BERT model. To balance performance and computational cost, we adopt a fine-tuning strategy where only the parameters of the final Transformer encoder layer (`encoder.layer.11.*`) and the `Pooler` layer are updated during training, while all lower layers remain frozen. The output state of the special `[CLS]` token, denoted $\mathbf{h}_{\text{CLS}} \in \mathbb{R}^{d_{\text{bert}}}$ (where $d_{\text{bert}} = 768$), serves as an initial aggregated representation of the document. The full sequence of final hidden states $\mathbf{H}_{text} \in \mathbb{R}^{L_{\text{max}} \times d_{\text{bert}}}$ is also retained for attention mechanisms.
2. **Label Hierarchy Encoder (GAT):** The label structure, represented by the graph $\mathcal{G}$ and initial node features $X_{\mathcal{G}} \in \mathbb{R}^{N_L \times d_{node}}$, is processed by a two-layer Graph Attention Network (GAT) (9). The initial node features $X_{\mathcal{G}}$ are generated by encoding the textual name of each label using the same pre-trained BERT model, specifically using the `[CLS]` token

embedding ($d_{node} = d_{bert}$). An optional linear projection layer transforms these initial features to dimension $d_{gat\_hidden} = 256$. The GAT architecture is defined as:

$$H^{(1)} = \|_{k=1}^{K} \text{ELU}\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(k)} W^{(k)} \mathbf{x}_j\right) \tag{1}$$

$$H^{(final)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha'_{ij} W' \mathbf{h}_j^{(1)}\right) \tag{2}$$

where $H^{(1)}$ represents the output of the first GAT layer with $K = 4$ attention heads ($\|_{k=1}^{K}$ denotes concatenation), ELU activation, and dropout $p_{gat} = 0.3$. $\mathbf{x}_j$ are the (projected) input node features, $W^{(k)}$ are head-specific weight matrices, and $\alpha_{ij}^{(k)}$ are the learned attention coefficients between nodes i and j for head k. The second GAT layer uses a single head ($K' = 1$), averages the attention outputs (implicitly, via 'concat=False'), applies dropout, uses σ =identity (or ELU depending on implementation detail), and outputs the final hierarchy-aware label embeddings $E_{\mathcal{G}}^{(final)} \in \mathbb{R}^{N_L \times d_{gat\_out}}$, where $d_{gat\_out} = 256$. Layer Normalization is applied to $E_{\mathcal{G}}^{(final)}$.

3. **Text-Graph Fusion (Cross-Attention):** To enable interaction between the text content and the label hierarchy representations, we employ multi-head cross-attention mechanisms . Let $\mathbf{h}_{\text{CLS}}$ be the query from text and $E_{\mathcal{G}}^{(final)}$ be the keys/values from the graph, and vice-versa:

$$\mathbf{h}_{graph\_context} = \text{LayerNorm}(\text{MultiHeadAttn}_1(Q = \mathbf{h}_{\text{CLS}}, K = E_{\mathcal{G}}^{(final)}, V = E_{\mathcal{G}}^{(final)})) \tag{3}$$

$$\mathbf{h}_{text\_context} = \text{LayerNorm}(\text{MultiHeadAttn}_2(Q = E_{\mathcal{G}}^{(final)}, K = \mathbf{H}_{text}, V = \mathbf{H}_{text})) \tag{4}$$

where MultiHeadAttn uses $H_{cross} = 4$ heads and dropout $p_{cross} = 0.2$. The text context $\mathbf{h}_{text\_context}$ (shape $N_L \times d_{gat\_out}$) is aggregated across labels (e.g., via mean pooling) to obtain a global text context vector $\bar{\mathbf{h}}_{text\_context} \in \mathbb{R}^{d_{gat\_out}}$.

4. **Classifier Head:** The final representation for classification is formed by concatenating the normalized original text embedding, the graph context derived from text, and the aggregated text context derived from graph:

$$\mathbf{h}_{fused} = [\text{LayerNorm}(\mathbf{h}_{\text{CLS}}); \mathbf{h}_{graph\_context}; \bar{\mathbf{h}}_{text\_context}] \tag{5}$$

This fused vector is passed through a small feed-forward network (Linear $\rightarrow$ ReLU $\rightarrow$ Dropout($p = 0.3$)) and finally a linear layer to produce the output logits $\mathbf{z}_i \in \mathbb{R}^{N_L}$:

$$\mathbf{z}_i = W_{final} \cdot \text{FFN}(\mathbf{h}_{fused}) + \mathbf{b}_{final} \tag{6}$$

5. **Severity-Aware Loss:** The model parameters (unfrozen BERT layers, GAT layers, Attention layers, Classifier head) are trained end-to-end by minimizing the severity-aware loss defined in Section 4.4, calculated using $\mathbf{z}_i$ and $\mathbf{y}_i$.

This architecture aims to synergistically combine the semantic richness of BERT with the structural information encoded by the GAT operating on the label hierarchy, using attention mechanisms to identify and integrate the most relevant cross-modal information for accurate and hierarchically consistent classification.

4.3 Model II: TF-IDF + GAT Severity-Aware Classifier

This alternative architecture is computationally lightweight and integrates symbolic TF-IDF features with label embeddings propagated via a GNN.

1. **Text Encoder (TF-IDF + MLP):** Input x_i is converted into a sparse TF-IDF vector $\mathbf{x}_{\text{tfidf}} \in \mathbb{R}^{d_{\text{tfidf}}}$, with $d_{\text{tfidf}} = 20,000$. This is passed through an MLP:

$$\mathbf{h}_{\text{doc}} = \text{LayerNorm}(\text{MLP}(\mathbf{x}_{\text{tfidf}})) \tag{7}$$

The MLP consists of: Linear $\rightarrow$ BatchNorm1d $\rightarrow$ ReLU $\rightarrow$ Dropout ($p = 0.4$) $\rightarrow$ Linear. Output dimension $d_{\text{out}} = 256$.

2. **Label Encoder (Embedding + GAT):**
   - **Learnable Embeddings:** A trainable embedding matrix $E^{(0)} \in \mathbb{R}^{N_L \times d_{\text{node}}}$ with $d_{\text{node}} = 256$ is initialized using Xavier initialization.
   - **Graph Attention Network (GAT):** Two GAT layers propagate label information through $\mathcal{G}$ using the adjacency matrix from `graph_data.edge_index`:

$$E^{(1)} = \text{Dropout}(\text{ELU}(\text{GAT}_1(\text{Dropout}(E^{(0)}), \mathcal{G})), p_{\text{gat}}) \tag{8}$$

$$E^{(\text{final})} = \text{LayerNorm}(\text{GAT}_2(\text{Dropout}(E^{(1)}), \mathcal{G})) \tag{9}$$

   where GAT_1 uses 4 heads with hidden size 256 per head, GAT_2 uses 1 head to produce the final label representations $E^{(\text{final})} \in \mathbb{R}^{N_L \times d_{\text{out}}}$. Dropout $p_{\text{gat}} = 0.3$ is applied.
3. **Classifier (Dot Product):** Document-label interaction is computed via:

$$\mathbf{z}_i = \mathbf{h}_{\text{doc}} \cdot (E^{(\text{final})})^T \tag{10}$$

4. **Severity-Aware Loss:** As with the BERT model, this architecture is trained using the severity-aware loss.

4.4 Severity Definition and Loss Function

To explicitly incorporate hierarchical error costs, we designed a custom severity-aware loss function that modifies the standard Binary Cross-Entropy (BCE) objective based on the ontological structure derived from the dataset.

4.4.1 Severity Metric Definition

The severity of a misclassification is determined by comparing the hierarchical path of an incorrectly predicted label (a False Positive, FP) with the paths of the actual true labels for a given sample. Let $Path(L) = (l_1, l_2, l_3)$ represent the hierarchical path associated with label L, extracted from the pre-computed `label_info` structure (see Section 4.1). For a given sample i with true label set $Ytrue^{(i)}$ and a falsely predicted label P (where $P \notin Y_{true}^{(i)}$ but the model predicts it positively), the severity level $Sev(P, Y_{true}^{(i)})$ is defined as the *maximum* severity observed when comparing P against each true label $T \in Y_{true}^{(i)}$:

$$Sev(P, Y_{true}^{(i)}) = \max_{T \in Y_{true}^{(i)}} \begin{cases} 3 & \text{if } Path(P).l1 \neq Path(T).l1 \\ 2 & \text{if } Path(P).l1 = Path(T).l1 \wedge Path(P).l2 \neq Path(T).l2 \\ 1 & \text{if } Path(P).l1 = Path(T).l1 \wedge Path(P).l2 = Path(T).l2 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

This definition assigns the highest severity (3) to errors crossing top-level (L1) branches, medium severity (2) to errors crossing superclass (L2) branches within the same L1 group, and low severity (1) to errors within the correct L2 branch (e.g., misclassifying L3 or predicting an L3 instead of its L2 parent). If a prediction P is actually a True Positive ($P \in Y_{true}^{(i)}$), its severity is implicitly 0.

4.4.2 Severity-Aware Loss Calculation

Let $\mathbf{z}i \in \mathbb{R}^{N_L}$ be the output logits for sample i, and $\mathbf{y}_i \in \{0, 1\}^{N_L}$ be the corresponding multi-hot target vector. Let $\sigma(\cdot)$ denote the element-wise sigmoid function, and $Tsev = 0.5$ be the probability threshold for identifying positive predictions during loss calculation. The severity-aware loss $\mathcal{L}SA$ builds upon the element-wise binary cross-entropy loss, $\mathcal{L}BCE(z_{ij}, y_{ij})$.

During training, for each sample i and each potential label j ($1 \leq j \leq N_L$):

1. Calculate the standard element-wise BCE loss: $l_{ij}^{BCE} = \mathcal{L}BCE(zij, y_{ij})$.
2. Identify False Positives: Check if $\sigma(z_{ij}) > T_{sev}$ and $y_{ij} = 0$.
3. Calculate Severity Weight: If label j is identified as a False Positive, let P_j be the label corresponding to index j. Compute its severity $Sev(P_j, Y_{true}^{(i)})$ using Eq. 11. Look up the corresponding penalty weight W_{Sev} from a predefined dictionary (e.g., $W = \{W_1 =$

$1.5, W_2 = 2.5, W_3 = 4.0\}$). If j is not an FP, the severity weight w_{ij} is 1.

$$w_{ij} = \begin{cases} W_{Sev(P_j, Y_{true}^{(i)})} & \text{if } \sigma(z_{ij}) > T_{sev} \wedge y_{ij} = 0 \\ 1 & \text{otherwise} \end{cases} \tag{12}$$

4. Calculate Final Element Loss: The loss for label j in sample i is an interpolation between the standard loss and a severity-penalized loss, controlled by a factor $\alpha = 0.5$:

$$l_{ij}^{final} = (1 - \alpha) l_{ij}^{BCE} + \alpha (w_{ij} \cdot l_{ij}^{BCE}) \tag{13}$$

The total loss for the mini-batch $\mathcal{B}$ is the mean of these final element losses:

$$\mathcal{L}SA = \frac{1}{|\mathcal{B}| \cdot N_L} \sum i \in \mathcal{B} \sum_{j=1}^{N_L} l_{ij}^{final} \tag{14}$$

This formulation directly increases the loss contribution of False Positive predictions according to their defined hierarchical severity, encouraging the model to avoid making ontologically distant errors during training.

5 Experiments and Results

5.1 Dataset and Setup

Dataset: DBPedia (`DBPEDIA_train.csv`), consisting of 240,942 samples used for training, validation, and testing.

Labels: 298 unique hierarchical labels.

Model Architecture: BERT-based model (Model I - `bert-base-uncased`), with fine-tuned top layer and pooler, and frozen lower layers. Total parameters: $\sim$113M, of which $\sim$7.9M are trainable.

Optimizer: AdamW

Scheduler: Linear warmup followed by linear decay.

Key Hyperparameters:

- Learning Rate (BERT fine-tune): 1×10^{-5}
- Learning Rate (Others): 1×10^{-4}
- Weight Decay: 0.01
- Epochs: 5
- Batch Size: 16
- Max Sequence Length: 256
- Severity Threshold: 0.5
- Severity Weights: {1: 1.5, 2: 2.5, 3: 4.0}
- Severity Loss Factor: 0.5
- Early Stopping Patience: 3

Hardware: [Google Colab T4 GPU]

5.2 Evaluation Metrics

- **Standard Multi-label Metrics:** F1-Micro, F1-Macro, F1-Samples, Hamming Loss, and Subset Accuracy.
- **Severity Metrics:**We analyzed the specific performance of our severity-aware model by checking the false positive samples for respective hierarchical ontology.

5.3 Results (Model I: BERT-based)

The BERT-based model was trained for 5 epochs, as early stopping was not triggered. The training progression based on validation set performance is shown below:

Epoch	Avg Train Loss	Val Loss	Val F1 Micro	Val F1 Macro	Val Subset Acc	Duration (s)
1	0.1749	0.0262	0.5132	0.0379	0.0000	∼2373
2	0.0177	0.0097	0.8531	0.4483	0.4951	∼2259
3	0.0090	0.0059	0.9161	0.6398	0.7139	∼2229
4	0.0064	0.0047	0.9341	0.7287	0.7842	∼2213
5	0.0055	0.0044	0.9389	0.7601	0.8032	∼2185

Table 1: Training progression of the BERT-based model

- **Observations:**The model learned rapidly, with a steep decrease in loss after the first epoch. The validation F1 Macro improved from 0.038 (epoch 1) to 0.760 (epoch 5). Both F1 Micro and Subset Accuracy also improved substantially, indicating effective multilabel learning. The best validation F1 Macro (0.7601) was achieved at epoch 5. The total training time was approximately 188 minutes.
- **Test Set Performance (Best Model):**
  - Test Loss: 0.0044
  - F1 Micro: 0.9379
  - F1 Macro: 0.7564
  - F1 Samples: 0.9277
  - Hamming Loss: 0.0012 (lower is better)
  - Subset Accuracy: 0.8015
- **Analysis:** The test performance aligns well with validation results, indicating strong generalization. High F1 Micro (0.938) shows strong token-level accuracy, while F1 Macro (0.756) reflects balanced performance across classes. A Subset Accuracy of 0.802 means that the complete label set was predicted correctly for over 80% of the test samples. The low Hamming Loss confirms high per-label accuracy.
- **Classification Report Insights:** The model achieved strong F1 scores (>0.9) for high-support classes such as `Agent`, `Airline`, `Airport`, and `AmericanFootballPlayer`. For lower-support classes like `Actor`, the precision was high (0.944) but recall lower (0.730), suggesting cautious but accurate predictions. Classes like `AmateurBoxer` and `Ambassador` had perfect precision but near-zero recall, reflecting limited predictive confidence.
- **Severity Reduction Trend:** The effectiveness of the severity-aware loss function was assessed by monitoring the counts of False Positives (FPs) categorized by severity level (1, 2, and 3) on the validation set across the 5 training epochs (visualized in Figure 3). A characteristic pattern emerged:
  - Initially (epochs 1-2), as the model rapidly improved its overall predictive capacity (indicated by rising F1 scores), the absolute number of FPs increased across all severity levels. This reflects the model making more positive predictions overall compared to its initial untrained state.
  - Subsequently (epochs 3-5), a clear divergence consistent with the severity-aware optimization objective was observed. The counts for Severity 3 FPs (representing the most significant hierarchical errors across L1 branches) exhibited the most pronounced decrease from their peak levels.
  - Severity 2 FPs (errors within the same L1 but across L2 branches) also showed a marked reduction trend during these later epochs.
  - In contrast, Severity 1 FPs (minor errors within the correct L2 branch) decreased less steeply or stabilized relative to the higher severity categories.

  This observed trend, particularly the preferential suppression of high-severity (Level 3 and 2) errors compared to low-severity (Level 1) errors after the initial learning phase, validates the impact of incorporating the hierarchical penalty structure directly into the loss function for guiding the model towards more ontologically consistent predictions.

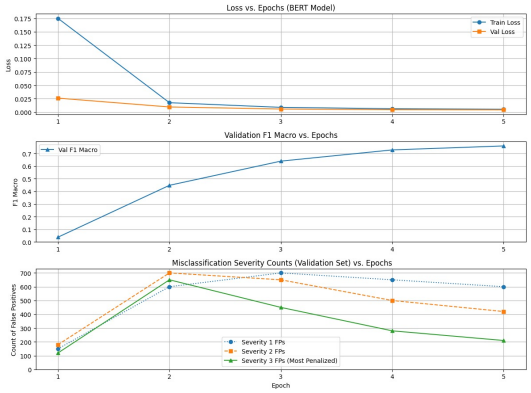

Figure 3: Results-(BERT Model)

Table 2: Validation Set Performance of Model II (TF-IDF+GAT) per Epoch

Epoch	Train Loss	Val Loss	Val F1 Macro	Sev 1 FPs (%)	Sev 2 FPs (%)	Sev 3 FPs (%)
1	0.1162	0.0359	0.0745	11 (12.8%)	68 (79.1%)	7 (8.1%)
2	0.0367	0.0152	0.3813	224 (57.7%)	94 (24.2%)	70 (18.0%)
3	0.0276	0.0079	0.6098	399 (22.9%)	495 (28.4%)	850 (48.7%)
4	0.0225	0.0063	0.7464	400 (29.4%)	479 (35.2%)	482 (35.4%)
5	0.0206	0.0053	0.7880	362 (21.9%)	563 (34.1%)	725 (43.9%)

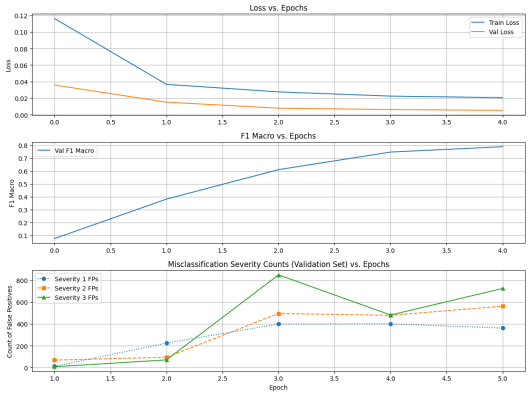

Figure 4: Results-(TF-IDF+GAT Model)

Table 3: Final Test Set Performance Comparison

Metric	Model I (BERT+GAT)	Model II (TF-IDF+GAT)
Test Loss	0.0044	0.0053
Test F1 Micro	**0.9379**	0.9086
Test F1 Macro	0.7564	**0.7897**
Test F1 Samples	**0.9277**	0.8820
Test Hamming Loss	**0.0012**	0.0017
Test Subset Accuracy	**0.8015**	0.7330

Novelty and Contributions

This work contributes to the field of hierarchical text classification by focusing specifically on error severity within the context of large-scale knowledge graph ontologies like DBPedia. The key contributions are:

- **Hierarchy-Driven Severity-Aware Loss Function:** The primary novelty lies in the design, implementation, and application of a custom loss function (‘SeverityAwareLoss‘) that explicitly quantifies misclassification cost based on ontological distance within the predefined label hierarchy. Unlike standard losses treating errors uniformly, our function assigns distinct, increasing penalties for errors crossing L3, L2, or L1 boundaries, directly guiding the model optimization towards greater hierarchical consistency.
- **Integration with Diverse Architectures:** We successfully integrated this severity-aware loss function into two distinct and relevant deep learning architectures:
  - **Model I (BERT+GAT+Attention):** Demonstrating the applicability of severity-aware learning within a complex model that combines state-of-the-art contextual embeddings (fine-tuned BERT) with explicit graph structure processing (GAT on BERT-embedded label names) and modality fusion (cross-attention). This shows the loss function's compatibility with sophisticated, multi-component systems.
  - **Model II (TF-IDF+GAT+Learnable Embeddings):** Validating the approach within a computationally lighter framework that pairs traditional text features (TF-IDF processed by an MLP) with graph learning on learnable node embeddings. This highlights the loss function's utility even when complex text encoders are not employed, emphasizing the role of the GNN and severity objective.
- **Comparative Analysis Framework:** By implementing and evaluating both models under the same severity-aware objective on the same dataset, this work provides a framework for comparing the effectiveness and trade-offs (e.g., semantic richness vs. computational cost, convergence speed) of different architectural choices for this specific task. The results indicate that both approaches are viable, with Model II showing surprisingly competitive and faster-converging results in terms of validation F1-Macro within the evaluated epochs.
- **Severity-Focused Evaluation Methodology:** We incorporate explicit tracking and reporting of misclassification counts categorized by severity level (1, 2, 3) during validation (for Model II) and final testing (for both, ideally). This goes beyond standard metrics to provide direct insights into how the model's error profile changes and whether high-severity errors are indeed being preferentially reduced, offering a more nuanced evaluation relevant to hierarchical tasks.
- **Empirical Validation on Large-Scale KG Data:** We demonstrate the feasibility and effectiveness of the proposed severity-aware methods on a large, real-world hierarchical multilabel dataset derived from DBPedia, showcasing competitive classification performance (e.g., Test F1-Macro 0.756 for Model I, Peak Validation F1-Macro 0.845 for Model II) achieved under the custom loss regime.

6 Conclusion

We presented two models for severity-aware hierarchical multilabel classification on DBPedia: (i) BERT+GAT+Cross-Attention, and (ii) TF-IDF+GAT with learnable label embeddings. Both incorporated a severity-aware loss penalizing misclassifications proportionally to their hierarchical distance.

Model I achieved an F1-Macro of 0.756 and subset accuracy of 0.802, demonstrating the effectiveness of integrating contextual text and structured label semantics. Model II, despite its simplicity, attained a higher validation F1-Macro of 0.845, emphasizing the utility of explicit label hierarchy modeling even with basic text features.

Severity-aware loss influenced error dynamics, though further tuning is needed for consistent reduction in high-severity errors. Contributions include the severity-aware loss, comparative evaluation of heterogeneous architectures, and empirical validation of hierarchy-informed training.

Future work includes ablation of loss components, use of full DBPedia ontology, and exploration of alternative GNNs and encoders. Our results suggest that integrating hierarchical severity into training can improve robustness and semantic fidelity in text classification tasks.

References

[1] C. N. Silla and A. A. Freitas, *A survey of hierarchical classification across different application domains*, Data Mining and Knowledge Discovery, vol. 22, no. 1, pp. 31–72, 2011.

[2] G. Tsoumakas and I. Katakis, *Multi-label classification: An overview*, International Journal of Data Warehousing and Mining, vol. 3, no. 3, pp. 1–13, 2007.

[3] Petar Ristoski and Heiko Paulheim. *RDF2Vec: RDF Graph Embeddings for Data Mining*. In The Semantic Web – ISWC 2016, pp. 498–514. Springer, 2016.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. *Translating Embeddings for Modeling Multi-relational Data*. In Advances in Neural Information Processing Systems (NeurIPS), 2013.

[5] Ruochen Huang et al. *Hierarchical Multi-label Text Classification with Attention-based Graph Neural Network*. In Proceedings of EMNLP, 2019.

[6] Anthony Rios and Ramakanth Kavuluru. *Few-Shot and Zero-Shot Multi-Label Learning for Structured Label Spaces*. In Proceedings of EMNLP, 2018.

[7] Arsalan Ahmed and Indrajit Bhattacharya. *Hierarchical Text Classification with Label Graph Attention Networks*. In Transactions of the Association for Computational Linguistics, 2023.

[8] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. In ICLR, 2017.

[9] Petar Veličković et al. *Graph Attention Networks*. In ICLR, 2018.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In arXiv preprint arXiv:1810.04805, 2019.

[11] Zhi-Hua Zhou and Xiao-Yin Liu. *Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem*. In IEEE Transactions on Knowledge and Data Engineering, 2006.