A Hierarchical Approach to Classifying Toxic Conversations

Stefan Andrei Alexandru
University of Bari Aldo Moro

Department of Computer Science

August 29, 2025

Abstract

Detecting the subtle nuances of toxic relational dynamics such as manipulation, coercion, and emotional harm is a particularly challenging task in natural language processing. Traditional classification models often struggle, largely because the linguistic patterns of different forms of abuse overlap heavily. In other words, the same kinds of words and expressions tend to appear across categories, blurring the boundaries and making it difficult for models to distinguish one type of harmful behavior from another.

This paper tackles the issue of class overlap head-on. First, we provide a quantitative analysis using semantic embeddings to empirically validate and measure the extent of this linguistic similarity. Next, we establish comprehensive performance baselines by evaluating traditional and state-of-the-art models on two competing feature strategies.

Base results show that class overlap sets a performance ceiling for traditional flat classification approaches. No conventional method adequately distinguishes between semantically similar toxic behavior categories.

To overcome this limitation, we introduce a hierarchical classification framework that decomposes the problem into a two-stage process. A generalist model first identifies broad super-categories of toxic behavior, after which specialized models perform fine-grained classification.

This hierarchical approach yields a significant improvement in classification accuracy, offering a robust and scalable solution to the pervasive issue of class overlap in complex text classification tasks.

1 Introduction

The shift to digital communication has significantly increased the amount of conversational data. These interactions hold a lot of information about the social relationships of the users, which include the abusive and toxic ones. The task of automated recognition of these kinds of interactions, from the exploitation and control to the emotional abuse, is the one that content moderation, online safety, and mental health technologies put at the very top of their list of priorities.

This task is generally considered to be a multi-class text classification problem. Though, in the case of relational dynamics, the categories are not as lexically distinct as in topic classification. These are presented through delicate hints, changing conversational patterns, and language that depends on the context.

One important issue addressed in this work is the high **linguistic similarity** between different classes. This semantic overlap can result in less clearly defined decision boundaries in the feature space, which may lead to a higher rate of misclassification and limit the achievable performance. This article acts like a detailed and well-structured survey of this problem and it also extends a considerable solution:

1. A quantitative analysis is performed using semantic embeddings to empirically validate and quantify the extent of linguistic overlap between classes.
2. Thorough performance baselines are established by systematically evaluating traditional and state-of-the-art models over two competing feature strategies, with a detailed description of the trade-offs of each approach.
3. A **hierarchical classification framework** is introduced specifically to dismantle the class similarity issue. The architecture, logic, and superior performance over flat classification paradigms are demonstrated.

2 Related Works

The detection of toxic language is at the center of this work, along with the identification of overlapping classes in text categorization and the use of the hierarchical classification as our third research area.

2.1 Toxic Language and Detection

Automated harmful content detection is a unit of the Natural Language Processing (NLP) domain, where it is considered a landmark application. The first contributions in the field mainly focused on the identification of explicit toxicity, such as hate speech or offensive language, using keyword-based methods and classic machine learning models. More recently, the contextualization power of deep learning models has been leveraged to detect even more subtle cases of toxicity [1].

2.2 Class Overlap in NLP

The closeness in semantic meaning of different classes is a cognizance that lies at the base of a plethora of fine-grained text classification tasks, such as toxicity detection, semantic similarity, and topic classification. If classes are not semantically separable, the flat classifiers fail to find the efficient decision boundaries, which in turn result in a performance ceiling. The problem is very acute when the question is about the classification of behaviors rather than topics, because the linguistic signs can be very ambiguous and context-dependent.

2.3 Hierarchical Classification

Hierarchical classification has been a traditional method for solving complicated classification problems. It works by breaking down the problem into smaller, structured sub-problems [4]. Instead of one model trying to separate all classes at once, a hierarchy of classifiers is employed. Koller and Sahami's work was among the first to show that such a method could yield very good results even when only a few features were used [5]. Though the effectiveness of hierarchical methods is unquestionable, their use to resolve the problem of semantic overlap in the specific area of toxic relational dynamics. By organizing the hierarchy according to real similarity between the classes, the framework takes this model further.

3 Dataset structure and analysis

The dataset is the basis of examination, which comprises **1000 dialogues**. These talks have been classified into **10 different classes**, each of which describes the various toxic dynamics types.

The length of the texts in the conversations is quite different, with an average length of about **153 words** and a standard deviation of **67 words**. Such variation shows that they have a rich and diverse set of data for the conversations, and they could be short or long, thus providing a good foundation for a model that can capture the subtle linguistic patterns. The lengths of the conversations are shown in Figure 1.

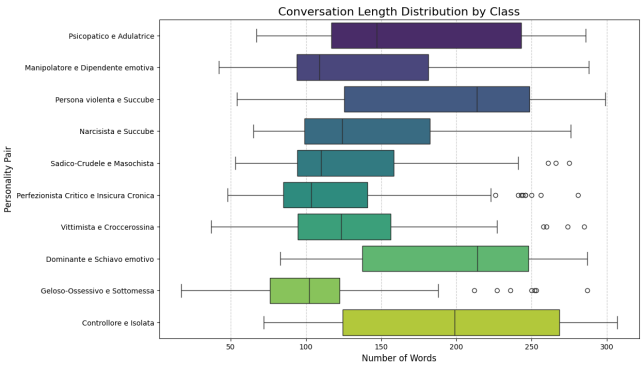

Figure 1: Distribution of Conversation Lengths

3.1 The Impact of Text Preprocessing

Differently from standard NLP practices, conventional text preprocessing steps, such as lemmatization and stop-word removal, tended to **reduce model performance**.

The reason is that the **most important information** is lost because the detection of the toxic relational dynamics is involved in the most delicate signals that are wiped out by normalization. For example, stop-words which appear to be the most insignificant ones turn out to be quite important: pronouns in such instances as *"Perché tutto deve sempre riguardare te?"* (Why must everything always be about you?) or *"Tu sei il mio unico mondo"* (You are my only world) are the ones that not only embody the whole accusation and dependency but also simultaneously become the most "lighthearted" words in these phrases. Similarly, if lemmatization is applied to a term like *"fallita"* (a female failure), the result will be the infinitive form "fallire" (to fail), and thus the gendered and definitive nature of the insult will be lost. At the same time, the removal of punctuation marks would make the angry character of the following words lighter: *"Un problema?! Sei sempre un problema!"* (A problem?! You are always a problem!).

Therefore, a minimal preprocessing strategy was adopted, preserving the raw text to retain these crucial nuances for model training.

3.2 Quantifying Semantic Overlap

The classification problem becomes evident when looking at the similarity matrix in Table 1. The very high similarity values indicate a strong semantic overlap between some classes. As a result, these classes are positioned very close to each other in the vector space, which makes it difficult for a single classifier to distinguish them by defining appropriate decision boundaries.

3.2.1 Analytical Procedure

The operations conducted to obtain such an effect are:

1. *Corpus Aggregation*: The texts of 10 classes of the training dataset were concatenated together with the class definitions. After that, representative corpora for each class were extracted from the merged data.

2. *Semantic Embedding Generation*: The tool to accomplish this was Sentence-BERT (SBERT). SBERT takes the text corpus for each class and transforms it into a machine-understandable dense vector (embedding). This vector then serves as one of the main features for the machine to perform further calculations. Actually, SBERT is a very effective model in creating text embeddings which still hold the original text semantics, thus the proximity of vectors would mean the semantic closeness.

3. *Similarity Matrix Computation*: For each pair of class embedding vectors, cosine similarity was computed. The cosine similarity is determined according to the formula:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

is a function that returns a value in the range of 0 to 1, which makes it very suitable for the comparison of vectors from the high-dimensional space as well as the display of their semantic connections.

3.3 Analysis and Implications

The main aspect of the classification problem can be observed in the similarity matrix, partially reported in Table 1. The high similarity values may be interpreted as an indication of notable semantic overlap between certain classes. As a result, these classes tend to be close in the vector space, which makes it more challenging for a single classifier to distinguish them reliably by identifying appropriate decision boundaries.

4 Basic Methodologies

Initially, a number of very stable baseline lines were established as the foundation of this study. This step consisted of the experimentation of various models for the two different data representation strategies.

Table 1: Cosine Similarity Between Select Class Embeddings

Class Pair	Similarity
Manipolatore e Dipendente emotiva & Geloso-Ossessivo e Sottomessa	0.85
Controllore e Isolata & Dominante e Schiavo emotivo	0.82
Persona violenta e Succube & Sadico-Crudele e Masochista	0.79
Narcisista e Succube & Psicopatico e Adulatrice	0.75
Vittimista e Crocerossina & Perfezionista Critico e Insicura Cronica	0.65

4.1 Data Representation

The format of the data was the single most important aspect of model design. The examination consisted of two radically different perspectives.

4.1.1 Strategy 1: Full Conversations

The method here is to treat a complete dialogue as a single document. In this way, all the turns are simply concatenated to form one input vector.

- **Advantages**: Provides the most context continuity.
- **Disadvantages**: the key linguistic cues can become diluted with the presence of large amounts of irrelevant conversation. Also, long inputs can be difficult for models with fixed context window sizes.

4.1.2 Strategy 2: Dialogue Turns

The method envisages the turn of each speaker as a separate unit of the training set. The model figures out the class of each turn, and the decision on the whole conversation is made by combining the per-turn outputs.

Aggregation Logic. Averaging or applying a simple majority vote across all turns can be misleading: one highly toxic utterance might outweigh ten neutral ones. To handle this, we introduce a multi-stage aggregation heuristic parameterized by three hyperparameters: the *high-confidence threshold* (θ_h), the *minimum confidence threshold* (θ_m), and a *margin parameter* (δ).

Formally, let a conversation consist of n turns, each producing a predicted class c_i with confidence $p_i \in [0, 1]$. The heuristic proceeds as follows:

1. **High-Confidence Knock-Out.** If $\exists i \in \{1, \ldots, n\}$ such that $p_i \geq \theta_h$, then the final class is immediately set to c_i.

$$\hat{C} = c_i \quad \text{if } p_i \geq \theta_h$$

2. **Confident Majority Vote.** Otherwise, consider the subset $S = \{c_i \mid p_i \geq \theta_m\}$. If $S \neq \emptyset$, assign as provisional label $\hat{C}_{maj}$ the majority class in S.

3. **Margin-Based Conflict Resolution.** Let $c^* = \arg\max_i p_i$ be the class of the most confident turn, with score $p^* = \max_i p_i$. If $c^* \neq \hat{C}_{maj}$, then:

$$\hat{C} = \begin{cases} c^* & \text{if } p^* \geq \max\{p_j \mid c_j = \hat{C}_{maj}\} + \delta \\ \hat{C}_{maj} & \text{otherwise} \end{cases}$$

4. **Fallback.** If $S = \emptyset$, assign the class of the most confident prediction:

$$\hat{C} = c^*$$

This formulation puts highest emphasis on signals that are clear and of high certainty, however it also allows for use of collective evidence when the confidence is more evenly distributed. The values of hyperparameters $(\theta_h, \theta_m, \delta)$ were determined by an exhaustive search over a grid on a validation set.

4.2 Baseline Performance

Three models were trained: Logistic Regression (LR) and Support Vector Machine (SVM) with TF-IDF vectorization as classical ML baselines, and a fine-tuned Large Language Model (LLM) as a deep learning baseline.

According to Table 2, the 'full conversations' method was the winner on each occasion over the 'dialogue turns' way of thinking for SVM and LR models, thereby giving a strong indication that the complete context plays a crucial role. The less than ideal performances from these models highlight the main idea of the paper: the problem of overlapping characteristics between the classes being the major cause of performance limitations that cannot be alleviated by just picking a stronger flat classifier.

5 The Hierarchical Framework

In order to solve the problem of overlap between classes directly, a system of hierarchical classifying was developed and put into practice. This model moves from one complex decision to a series of simpler, more precise decisions.

Let $\mathcal{X}$ be the input space and $\mathcal{Y} = \{y_1, y_2, ..., y_K\}$ be the set of original fine-grained classes, respectively. We start by defining a M-partition of $\mathcal{Y}$, which represents the union of disjoint sets of super-categories:

$$\mathcal{Y} = \bigcup_{m=1}^{M} \mathcal{Y}_m, \quad \mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset \text{ for } i \neq j,$$

Each $\mathcal{Y}_m$ is then a set of classes that are semantically close to each other.

5.1 Architecture and Logic

The framework operates as a two-stage cascade:

1. **Stage 1: The Generalist Model**. This is a classifier trained on broad "super-categories". These super-categories are formed by merging the original classes that the analysis identified as semantically similar. The purpose of this model is to perform a coarse-grained classification, reliably sorting a conversation into a less ambiguous parent group (e.g., 'Controlling Dynamics').

   The Generalist Model $G : \mathcal{X} \to \{1, \ldots, M\}$ outputs a probability distribution over super-categories:

$$P_G(m \mid x) = \frac{\exp\left(f_m(x)\right)}{\sum_{j=1}^{M} \exp\left(f_j(x)\right)},$$

   where $f_m(x)$ is the logit score for super-category m. The predicted super-category is:

$$\hat{m} = \arg\max_m P_G(m \mid x).$$

2. **Stage 2: The Specialist Sub-Models**. For each super-category containing more than one original class, a dedicated specialist model is trained. Crucially, each specialist is trained **only** on the subset of data belonging to its assigned super-category. This enables the model to dedicate its full learning capacity to the difficult task of distinguishing between a few, highly similar subclasses (e.g., distinguishing 'Controllore e Isolata' from 'Dominante e Schiavo emotivo').

   For each super-category $\mathcal{Y}_m$, a Specialist Model $S_m : \mathcal{X} \to \mathcal{Y}_m$ is trained only on data within that group. It outputs:

$$P_{S_m}(y \mid x, \hat{m}) = \frac{\exp\left(g_y(x)\right)}{\sum_{z \in \mathcal{Y}_{\hat{m}}} \exp\left(g_z(x)\right)}, \quad y \in \mathcal{Y}_{\hat{m}}.$$

5.1.1 Inference Pipeline

During inference time, the classification method is a hierarchical decision flow that is aimed at integrating the scope of a generalist with the accuracy of specialists. The operation goes through these four steps:

1. **Generalist Prediction:** The first step is handing over the input sample x to the Generalist Model $G(\cdot)$ which then emits a probability distribution $p_G(y)$ over a set of coarse *super-categories* $\mathcal{Y}_G$. After that, we have the super-category with the highest predicted probability as the one predicted by the Generalist:

$$\hat{y}_G = \arg\max_{y \in \mathcal{Y}_G} p_G(y).$$

Table 2: Performance of Baseline Models across Data Representation Strategies (Metrics in %)

	Full Conversations			Dialogue Turns		
Model	Accuracy	Recall	F1-Score	Accuracy	Recall	F1-Score
Logistic Regression	73.33	73.33	72.85	63.00	63.00	61.36
Support Vector Machine	70.67	70.67	70.59	68.67	68.67	66.67
Large Language Model	66.00	67.48	66.47	67.33	67.33	67.52

2. **Routing:** After obtaining $\hat{y}_G$, a routing function $\mathcal{R}(\cdot)$ based on it directs the new instance to the Specialist Model $S_{\hat{y}_G}$. There is a separate specialist S_c for each super-category that is only trained on data from that category. The process ends here, and $\hat{y}_G$ is output as the final prediction if $\hat{y}_G$ is a terminal class (i.e., it does not have any more detailed sub-classes).

3. **Specialist Prediction:** When the first branch is not a terminal, the Specialist Model $S_{\hat{y}_G}(\cdot)$ is responsible for the task of giving the probability distribution $p_S(y \mid \hat{y}_G)$ over the finer-grained sub-classes $\mathcal{Y}_S^{(\hat{y}_G)}$ of the predicted super-category. The final class prediction within this branch is

$$\hat{y}_S = \arg \max_{y \in \mathcal{Y}_S^{(\hat{y}_G)}} p_S(y \mid \hat{y}_G).$$

4. **Final Output:** Either the Generalist's prediction $\hat{y}_G$ (if terminal) or the Specialist's refined prediction $\hat{y}_S$ (if non-terminal) is returned by the system. In this way, the decisions are not only globally consistent but also locally adjusted.

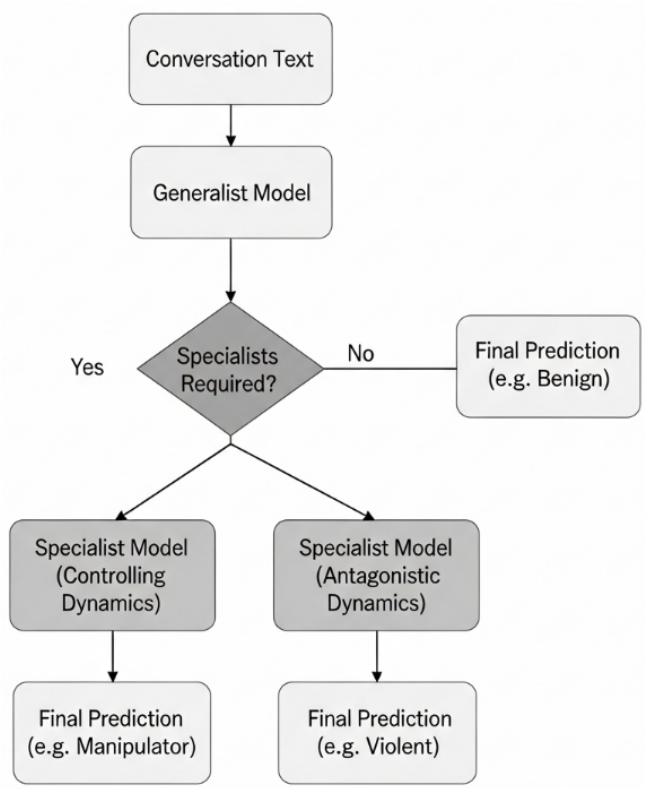

Figure 2: Flowchart of the hierarchical classification logic. The input is first routed by a Generalist Model and then, if necessary, to a specialized classifier.

Hierarchical inference pipeline such as this one strikes a balance between coverage (through the Generalist) and granularity (through the Specialists). Essentially, the Generalist performs the function of a broad filter, whereas the Specialists supply the detail necessary for subtle, fine-grained differentiations. Besides, the modular architecture facilitates efficient enlarging: the addition of new specialists for training and use in the hierarchy can be done without the need to re-train the entire hierarchy.

5.2 The Hierarchical Approach

Advantages. The hierarchical strategy offers several benefits:

- ***Problem Decomposition***. Instead of asking a single model to directly solve a very complex classification task, the problem is decomposed into simpler subtasks. The Generalist focuses only on distinguishing broad categories, while each Specialist deals exclusively with the finer distinctions within its own domain. This separation reduces noise: a Specialist does not need to worry about classes belonging to other groups, and can therefore reach higher accuracy within its restricted scope.

- ***Modularity and Scalability***. The system is modular: new Specialists can be added when new categories appear, or existing ones retrained or replaced without retraining the entire pipeline.

Disadvantages. The hierarchical approach, however, has one notable limitation:

- ***Error Propagation***. it depends on the chain of decisions. If the Generalist incorrectly predicts the next step, the data point may be routed to a suboptimal Specialist. In this case, the error may propagate through the hierarchy.

6 Results and Discussion

The hierarchical framework was evaluated at both the component level and as an end-to-end system.

6.1 Component and Performance

The performance of the individual models within the hierarchy is detailed in Table 3.

Table 3: Performance of Hierarchical Model

Model	Accuracy (%)	F1-Score (%)
Generalist Model	77.00	75.85
Specialist Sub-Models		
Controlling Dynamics	77.50	77.68
Antagonistic Dynamics	83.33	84.41
Hierarchical Model	89.33	89.27

The Generalist model was successful in its basic task. Additionally, the Specialist models showed great results in their detailed tasks, which means that they were able to recognize the small differences between the classes that the flat models could not.

On a simulated production environment, the experimental evaluation of the integrated system, from start to finish, reached an accuracy of **89 %** as its final result.

6.2 Discussion

The hierarchical framework's final accuracy of 89% is a 16 percentage point absolute improvement over the best baseline model (a flat Logistic Regression with 73% accuracy) .

Among other things, a very impressive aspect is that this increase in performance was attained by using the same underlying algorithm (Logistic Regression) for each component of the hierarchy. Hence, the result is a clear indication that the main claim is true: the architectural change, which is explicitly designed to fit the problem's known structure (i.e., class similarity), is a more effective way than just using a generic, "flat" model. The hierarchical approach went a long way in addressing the issue of semantic overlap that was at the heart of the problem, thereby enabling classification to be done more accurately and reliably.

7 Conclusion and Future Work

Class similarities were identified as a primary factor when categorizing negative interactions in toxic relationships. One possible reason for the limited performance of traditional classification models is the overlap of linguistic features. The two-stage hierarchical approach proved effective in decomposing the problem and achieved noticeable performance improvements over the tested baselines.

This research highlights the need for model architecture that takes these issues explicitly into account. In NLP, tasks where semantic ambiguity and class overlap are present, a hierarchically structured approach is a more scalable and effective solution to be closer to the accurate results. Techniques like soft-decision routing, wherein a particular data point can be allocated to different specialists with separate confidence scores, are an optimistic avenue for further fortifying the system's robustness.

References

[1] J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos. "Toxicity Detection: Does Context Really Matter?". In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[2] Z. Waseem and D. Hovy. "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter". In *Proceedings of the NAACL Student Research Workshop*, 2016.

[3] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

[4] C. N. Silla and A. A. Freitas. "A survey of hierarchical classification across different application domains". *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

[5] D. Koller and M. Sahami. "Hierarchically classifying documents using very few words". In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[6] E. Cappuccio, B. Muscato, L. Pollacci, M. Marchiori Manerba, et al. "Beyond Headlines: A Corpus of Femicides News Coverage in Italian Newspapers". In *Proceedings of the Tenth Italian Conference on Computational Linguistics (CLiC-it 2024)*, 2024.