UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Allana Tavares Bastos

Diversity-Aware Graphormer: Insights from Hate Speech Detection on 4chan

Belo Horizonte
2024

Allana Tavares Bastos

Diversity-Aware Graphormer: Insights from Hate Speech Detection on 4chan

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Dr. Martín Gomez Ravetti

Belo Horizonte
2024

Bastos, Allana Tavares.

B327d Diversity-Aware Graphormer: [recurso eletrônico]: Insights from Hate Speech Detection on 4chan / Allana Tavares. Bastos Belo Horizonte – 2024.
1 recurso online (63 f. il., color.) : pdf.

Orientador: Martín Goméz Ravetti.

Dissertação (Mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f. 55-61

1. Computação – Teses. 2. Redes neurais (Computação) – Teses. 3. Processamento de linguagem natural (Computacao) – Teses. 4. Redes sociais on-line. 5. Sistemas complexos – Teses. I. Goméz Ravetti, Martín. I. Universidade Federal de Minas Gerais Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*82(043)

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS

DIVERSITY-AWARE GRAPHORMER: INSIGHTS FROM HATE SPEECH DETECTION ON 4CHAN

ALLANA TAVARES BASTOS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Prof. Martin Gómez Ravetti - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Antonio Alfredo Ferreira Loureiro
Departamento de Ciência da Computação - UFMG

Prof. Thiago Henrique Nogueira
Departamento de Engenharia de Produção - UFV

Prof. Tiago Alves Schieber de Jesus
Departamento de Ciências Administrativas - UFMG

Belo Horizonte, 09 de dezembro de 2024.

Documento assinado eletronicamente por **Martin Gomez Ravetti**, **Professor(a)**, em

14/01/2025, às 16:12, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Thiago Henrique Nogueira**, **Usuário Externo**, em 15/01/2025, às 09:59, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Antonio Alfredo Ferreira Loureiro**, **Professor(a)**, em 04/03/2025, às 07:53, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Tiago Alves Schieber de Jesus**, **Professor do Magistério Superior**, em 10/03/2025, às 19:03, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3881475** e o código CRC **0E9AC4E1**.

Referência: Processo nº 23072.202581/2025-31 SEI nº 3881475

Acknowledgments

Com imensa gratidão, agradeço meu orientador, Prof. Dr. Martín Ravetti, pelas conversas, oportunidades e orientação contínua.

A minha mãe, Antelene, pelo apoio contínuo e pela inspiração. Obrigada por me ensinar novas palavras e tornar outras mais belas. Seu amor e apoio incondicionais foram essenciais em cada passo de minha caminhada.

Ao Prof. Dr. Tiago Schieber, por me apresentar o maravilhoso mundo de Sistemas Complexos.

A Luiza (e, claro, Marísia e Pupu), Thalissa e Luiza, pelas risadas e presença.

Aos novos amigos, colegas e professores do PPGCC e do Future Lab, por tornarem a caminhada do mestrado mais leve.

A UFMG, por continuar tanto me ensinando e por ser uma segunda casa.

A CAPES, pela concessão da bolsa de mestrado.

À banca, por sua leitura e contribuições.

"*O mundo ficaria todo cheio de preás, gordos, enormes.*"
(Graciliano Ramos)

Resumo

A arquitetura de Transformers remodelou o cenário da IA, impulsionando avanços importantes como o BERT. O Graphormer, uma Rede Neural de Grafos que utiliza mecanismos de atenção baseados em Transformer, é bem sucedido na captura de relacionamentos complexos em dados estruturados como estruturas moleculares e sistemas de recomendação. Esse sucesso pode ser atribuído às capacidades do Graphormer em aproveitar informações topológicas do grafo. Com base nessa arquitetura mais interpretável, é proposta uma nova abordagem baseada em Graphormer que incorpora uma medida de diversidade para melhorar a representação dos relacionamentos entre nós. O modelo é avaliado em duas tarefas de classificação textual, demonstrando o desempenho do modelo em comparação com outros métodos por meio de análises quantitativas e qualitativas. Essa pesquisa destaca o potencial dos Graphormers aliados à diversidade para abordar tarefas desafiadoras baseadas em grafos em tarefas de classificação textual, com potencial de aplicação em outras áreas.

Palavras-chave: processamento de linguagem natural; redes neurais de grafos; redes sociais; sistemas complexos

Abstract

The Transformer architecture has reshaped the AI landscape, powering important breakthroughs like BERT. Graphormer, a Graph Neural Network that leverages Transformer-based attention mechanisms, has found success in capturing complex relationships in structured data like molecular structures and recommendation systems. This success can be attributed to Graphormer's ability to leverage the graph's topological information. Building upon this interpretable model, we propose a novel Graphormer-based approach incorporating a diversity measure to improve the representation of node's relationships. The model is evaluated on two text classification tasks, showcasing its performance compared to other methods through quantitative and qualitative analyses. Our research highlights the potential of diversity-aware Graphormers for addressing challenging graph-based tasks in text classification, with promising applications in other domains.

Keywords: natural language processing; graph neural networks; social networks; complex systems

List of Tables

List of Abbreviations and Acronyms

BFS	*Breadth-First Search*
BERT	*Bidirectional Encoder Representations from Transformers*
BOW	*Bag of Words*
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CBOW	*Continuous Bag of Words*
DAG	*Directed Acyclic Graph*
GNN	*Graph Neural Network*
GZIP	*GNU ZIP*
iff	*If and Only If*
KL	*Kullback-Leibler divergence*
KNN	*K-Nearest Neighbors*
LIME	*Local Interpretable Model-agnostic Explanations*
LSTM	*Long Short-Term Memory*
LZMA	*Lempel-Ziv-Markov chain Algorithm*
LZ77	*Lempel-Ziv 77*
NCD	*Normalized Compression Distance*
NLP	*Natural Language Processing*
NP-hard	*Non-deterministic Polynomial-time hard*
OOD	*Out-of-Distribution*
POS	*Part-of-Speech tagging*
RNN	*Recurrent Neural Network*
SHAP	*SHapley Additive exPlanations*
SOTA	*State Of The Art*
SVM	*Support Vector Machine*
TF-IDF	*Term Frequency-Inverse Document Frequency*
TXT	*Text File Format*
UFMG	Universidade Federal de Minas Gerais
WL-test	*Weisfeiler-Lehman test*

Contents

Chapter 1

Introduction

Transformers [62] have dominated many domains, from computer vision to natural language processing, as the state-of-the-art architecture for learning. Their encoder-decoder architecture and attention mechanisms excel at modelling long-range dependencies in text and capturing contextual information more effectively than previous deep learning models. However, as for graph representation learning, Transformers' performance does not excel that of Graph Neural Networks (GNNs).

Graph representation learning aims to incorporate graph structures' information and attributes into typically low-dimension embeddings via a machine learning model [67, 6]. There is no straightforward way to encode the high-dimensional, non-Euclidean information about graph structure into a feature vector [19]. A key challenge in learning and optimizing this mapping of nodes or even entire graphs to a low-dimensional vector space is the trade-off between expressiveness and efficiency [6].

In this context, Graph Neural Networks (GNNs) effectively learn from the complex relationships within a graph by iteratively aggregating information from neighbouring nodes and updating their own representations. This makes GNNs well-suited for various tasks, including recommendation systems, NLP classification tasks, and graph generation tasks - like machine translation and summarization [67]. Furthermore, GNNs are intrinsically more interpretable than other deep learning models [68]. However, GNNs face their own shortcomings, especially regarding their expressiveness limitations and oversmoothing.

Graphormer [71] addresses the expressiveness limitations of traditional GNNs by leveraging a Transformer architecture with self-attention to enable attention to all nodes when updating nodes. Thus, it effectively encodes the structural information of a graph into the model, better modelling the relationship for node pairs and successfully representing the graph. Graphormer was initially tested on molecule modelling tasks, but was also altered and applied in various tasks ranging from image reconstruction [31] to recommendation systems [65] and extractive multimodal summarization [23].

We propose a Graphormer adaptation for text classification, using a challenging context to further investigate the model's capabilities for graph classification. We test this approach to classify online hate speech and expressions of suicidal ideation. Due to the

sensitive nature of these textual examples, reader discretion is advised. The examples of hateful comments included in this paper are presented solely to illustrate the classification process and results of the models used. These comments do not reflect the views or beliefs of the authors.

Our contributions are as follows:

- Setting: we adapt Graphormer, a model adept in preserving graph's topological information, with diversity-awareness,
- Methods: we use our adapted Graphormer for hate speech classification and multi-target text classification with the incorporation of sarcasm detection in our models, the Rhetoric Graphormers (HRG, HSRG, and SRG),
- Analyses: we demonstrate the performance of diverse-aware and multi-target models in hate speech classification, finding Graphormer to be a promising interpretable architecture for graph-based tasks.

1.1 Objectives

This research aims to analyse the effectiveness of graph modelling with self-attention for preserving structural information, as proposed in the Graphormer architecture, for NLP tasks. Additionally, we will explore the impacts of incorporating diversity awareness to node representation. More specifically, this research aims to respond:

- How can an interpretable graph-based approach, Graphormer, be used in text classification? What are the advantages of this approach?
- How does diversity awareness influence a Graphormer-based classifier?
- For the application in hate speech detection, what are the effects of multi-target classification including sarcasm?

Chapter 2

Literature Review

Network Science is an interdisciplinary field that studies the principles governing networks, providing insight into complex behaviours that emerge within them. This field provides insightful metrics for complex networks, capturing patterns and their influence on the network as a whole, which can represent social interactions. A key component of Network Science is graph theory, which serves as a formalism for representing relationships and structures. This representation tool can accurately represent natural language, aiding in several Natural Language Processing (NLP) tasks, such as knowledge networks and sentiment analysis. In NLP, graphs can be used to model relationships between words, sentences, or documents

As such, to understand the role of graphs in hate speech detection, it is necessary to first define and explore the concepts of graphs, NLP, and hate speech. To that end, this section will explore network science concepts and metrics, the use of word embeddings and classification tasks in NLP, and the challenges of hate speech detection. Additionally, it highlights the application of graph structures in modelling language, the integration of graph theory with deep learning, and the specific use of Graphormer in NLP.

2.1 Complex Networks in Natural Language Processing (NLP)

Social interactions can be understood as complex networks, i.e., a graph with the emergence of nontrivial topological features. To model them as such is very useful to understand and control its behaviour [27, 21, 7]. In the context of network theory, a complex network is an interconnected network of interactions between nodes from which the overall behaviour can not be derived. Besides the structural properties of the system, understanding the heterogeneity of interactions and their correlations with the topology is fundamental to better comprehend the system's properties. Hence, the use of Graph

Theory to model networks as graphs makes it easier to examine, understand and control their structure and dynamics [33, 55]. Combined with Information Theory's contributions [17] to the field, this research framework provides measures of significant properties.

Before delving into the metrics, let's formally define a written communication graph. There are different ways of representing textual information in a graph, but for this work consider a graph

$$G = (V, E), \text{ where:} \tag{2.1}$$

- V is a finite set of nodes/vertices, representing comments.
- E is a set of pairs of elements from V, representing the edges connecting the nodes.

The specific meaning ascribed to these connections may vary depending on the research objectives and chosen model. Edges might represent co-occurrence of specific words or, in the case of comments and replies, the hierarchical reply structure. Directed Acyclic Graphs (DAGs) are particularly well-suited to represent such scenarios. Formally, a DAG is denoted as

$$G_d = (V, E_d), \text{ where:} \tag{2.2}$$

- V represents a finite set of nodes, each corresponding to a comment.
- E_d is a set of directed edges that represent the reply structure between comments. An edge $(v_i, v_j) \in E_d$ signifies that comment v_i is a reply to comment v_j.

A crucial property of DAGs is their acyclicity, meaning there are no cycles or loops within the graph structure. This ensures a clear hierarchical relationship between comments, mirroring the parent-child structure of interactions observed in many social media platforms.media domains, with a parent-child structure to interactions between users.

The effectiveness of graphs' representations ultimately relies on the selection of nodes, edge definition (including weighting if applicable), and the overall graph construction [19]. And this graph construction is critical, as not only its structure must properly model the data, but also it must be able to resolve its target NLP task in a computationally acceptable manner. Many graph-based algorithms are NP-hard, meaning they struggle to scale with large datasets [42]. While theoretical proofs may guarantee that graph problems eventually converge, or have a solution, finding them computationally is another issue.

Furthermore, real-world communication scenarios present additional challenges. These graphs are often highly dynamic, with nodes and edges being added or removed over time. Traditional static algorithms may not efficiently handle these changes, requiring

specialised update mechanisms. Additionally, large graphs with a high number of edges lead to a high-dimensional representation. This complexity poses challenges for traditional machine learning algorithms, requiring specialised techniques for efficient processing and feature extraction. As such, the goal remains to identify suitable graph structures and computationally efficient algorithms that can effectively solve NLP tasks using the least computation time and memory possible.

Some challenges in feature extraction and representation learning stem from graphs' inherent properties. Thus, we explore some key properties, namely centralities, clustering coefficient, homophily, and diversity.

Centrality, the property of a node's or cluster's influence within a network, can also significantly impact network dynamics. However, there is no prevailing definition of centrality. This leads to several centrality measures that have been developed, each prioritising local or global influence [3]. The presence of various centrality measures raises the question of which centrality best reflects the importance of a node for a target NLP task. Choosing the optimal centrality measure directly impacts the extracted features and their effectiveness in capturing relevant information.

For instance, degree centrality, a local measure, simply counts the number of direct connections attached to each node. Represented by Equation 2.3, it iterates over all u neighbours of the node v in the set V. The adjacency matrix element a_{uv} indicates the presence of an edge between v and u.

$$C_D(v) = \sum_{u \in V} a_{uv}, \tag{2.3}$$

This measure has low computational costs for large networks if compared to the other well-known centrality measures, based on betweeness and closeness, making it well suited for large-scale online social webs [8]. However, while computationally efficient, degree centrality has limitations due to its local nature. A densely connected node (hub) might be peripheral to the network's flow of information, despite its high degree.

On the other hand, the closeness centrality measures the average length of the shortest path between a node v and the other nodes u in the set V, excluding v itself:

$$C_C(v) = \frac{1}{\sum_{u \in V, u \neq v} d(v,u)}, \tag{2.4}$$

$d(u,v)$ being the shortest distance between the nodes v and u. As such, the influence of a node is intuitively evaluated based on its accessibility to other nodes. Conversely, this measure presents a narrow range of variation,potentially assigning the same level of centrality to nodes with varying levels of influence. This makes it more suitable for scenarios where distances between nodes are higher than in random networks with the similar characteristics, informing information gateways and helping identify highly accessible subsets of nodes in DAGs [38].

Another measure that considers a node's relative position within the network is the betweenness centrality [16]. It measures how often a node lies on the shortest path between other node pairs. Formally, the betweenness centrality for a node v is defined as the sum of fraction of shortest paths between all node pairs s and t, excluding the node v itself, that pass through node v $\sigma_{st}(v)$ divided by the total number of shortest paths between s and t ($sigma_st$) in the network:

$$C_B(v) = \sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{2.5}$$

This centrality highlights nodes that bridge different parts of the communication network, potentially acting as information brokers. However, the betweenness centrality is not general, since information can travel long distances in a network, challenging this measure's effectiveness. While it indicates a node's potential control over interactions, its calculation can be computationally expensive for large networks. As a result, approximation methods based on random walks are often employed [43, 3].

Eigenvector centrality is a global measure represented by:

$$Av = \lambda v, \tag{2.6}$$

finding the eigenvalue v corresponding to the largest eigenvalue λ of the adjacency matrix A. Unlike degree centrality, it considers not only the number of a node's connections but also the centrality of its neighbours. Essentially, nodes connected to highly central nodes are deemed more important. This highlights a key difference from degree centrality - a node's importance is not solely based on its direct connections, but also on the influence of its surrounding network. Therefore, eigenvector centrality provides a network-wide perspective of a node's importance, considering the entire network structure and relationships between all nodes [55].

On the matter of the nodes' relationships, an important property is homophily, that refers to the tendency of nodes to form connections with others who share similar attributes [2, 25]. In real-world networks, this principle can lead to densely connected clusters within the graph. This local structure makes it difficult to learn features that capture global relationships between distant nodes, as denoted by $f : V \times V \rightarrow \mathbb{R}$ for nodes v_i and v_j. Homophily can be quantified by analysing node attributes within the network and can be used to generate features that capture the similarity between nodes.

On the other hand, diversity is defined by the distance between its elements, following non-euclidean principle [7]. The greater the distances, the greater the differences between the nodes and, consequently, greater the network diversity. Large graphs often exhibit high diversity in node features and connections. This high dimensionality presents a challenge for traditional machine learning algorithms. Feature compression techniques, denoted as $F : \mathbb{R}^d \rightarrow \mathbb{R}^k$. (where d is the original feature space and k is the compressed

space), are crucial for efficient processing and effective representation learning. This representation learning, as will be further detailed in a next section 2.1.2, is an essential component of nodes' and graphs' classification.

One useful diversity measure is introduced by Schieber and collaborators [57]. The D measure is a discriminative and computationally efficient metric for quantifying dissimilarity between graphs by capturing both global and local topological differences. It does so through three main terms, each involving probability distribution functions (PDFs) based on graph distance characteristics. The first term evaluates global network differences by comparing overall distance distributions, accounting for metrics like average path length. The second term focuses on local heterogeneity, comparing the connectivity profiles of individual nodes across the graphs. The third term, involving α-centrality, examines centrality across nodes, reflecting connectivity patterns and distinguishing structures such as complete and distance-regular graphs. By computing the Jensen–Shannon divergence between these PDFs, the D measure quantifies structural differences with high precision, returning zero for isomorphic graphs and non-zero values to quantify non-isomorphic differences. Although computationally intensive for sparse graphs, D can be effectively used without α-centrality in certain cases, maintaining accuracy. This measure has broad applications, including network model assessment, phase transition detection, and real-world systems analysis, as well as uses in image recognition and multilayer network studies.

Another important measure is the clustering coefficient, that measures how well connected are the neighbours of a node [60]. This can help understand the network's topology, as the clustering coefficient has been taken as a indication of a hierarchical structure when it decreases with the node's degree in real networks [53]. However, the clearly identifiable modular organisation of a scale-free network causes degree-correlation biases in the clustering coefficient definition. Filtering the degree correlations, the average clustering coefficient quantifies the degree to which nodes in a graph tend to cluster [29]. As such, the choice of clustering algorithm and the number of clusters significantly impact the resulting node representations and their suitability for the target NLP task.

Another important property is isomorphism, which compares graphs' underlying structure. Two graphs are considered isomorphic if there exists a one-to-one correspondence between their nodes that preserves edge connections. In essence, isomorphic graphs are structurally equivalent, with identical patterns of connectivity. Formally, the graphs $G = (V, E)$ and $G' = (V', E')$ are isomorphic if there is a bijective function $f : V \to V'$ such that for any $u, v \in V$ $f(u)$ and $f(v)$ are adjacent in G iff they are adjacent in G'. This concept is crucial in graphs' comparisons and in machine learning for detecting similarities among data points. However, determining graph isomorphism is computationally challenging, and no known polynomial-time algorithm exists for general graphs [73]. A common approach to match graphs is the use of topological descriptors mapping each graph to a feature vector, capturing its structural properties. However, this transforma-

tion can lead to a loss of topological information, potentially limiting the effectiveness of graph comparison techniques.

The aforementioned graph properties and measures provide a foundational understanding of the structural characteristics inherent in graph-structured data. These concepts play a crucial role in shaping and understanding the behaviour of algorithms and models designed to operate on graphs. By leveraging these properties, we can develop more effective feature extraction techniques and representation learning approaches tailored to specific graph-based tasks.

2.1.1 Graphs in NLP

With a solid grasp of graph properties, we now turn our attention to the application of graph-based methods in Natural Language Processing (NLP). In this section, we will explore how graph representations can be used to capture the complex relationships between words, sentences, and documents, enabling a deeper understanding of textual data and facilitating various NLP tasks.

The application of graph theory in NLP is a well-established approach for uncovering inherent patterns within data. Furthermore, graph formalisms have demonstrated utility in unsupervised learning tasks such as part-of-speech (POS) tagging and word sense induction [42]. Additionally, their application extends to semi-supervised settings, where a limited set of labelled examples can be used alongside the graph structure to propagate labels throughout the network.

The first issue lies on the construction of a graph from the input sequences of texts. This can be achieved statically or dynamically [67]. Static approaches construct the graph structures during preprocessing. Usually, this construction is rule-based or leverages parsing tools, for example, dependency parsing. As different domain/external knowledge hidden in the original text sequences are incorporated, the raw text is augmented with rich structured information. Static graph construction includes several graph techniques, for instance dependency graph, knowledge graph, coreference graph, co-occurrence graph, and similarity graph construction. Despite most static graph construction methods only considering one specific relation between nodes, a hybrid graph can be built by combining several graphs together. A common strategy concerns representing different aspects of the data in a heterogeneous graph, with multiple types of nodes and edges. This dynamical graph construction includes graph similarity metric learning, graph sparsification techniques, and a combination of intrinsic and implicit graph structures.

To preserve as much graph property information while representing in a low di-

mensional space, graph embedding algorithms can quantify these properties differently, depending on how they deal with similarities. Matrix factorisation based graph embedding represent its properties (e.g. node pairwise similarity) in the form of a matrix and factorise this matrix to obtain node embedding. The problem of graph embedding can be treated as a structure-preserving dimensionality reduction problem that assumes the input data lie in a low dimensional manifold. There are two types of matrix factorisation based graph embedding: graph Laplacian eigenmaps, and node proximity matrix [6]. The node proximity matrix factorisation is used mostly to embed homogeneous graphs and graphs constructed from non-relational data.

Graph Laplacian Eigenmaps interpret the graph property as pairwise node similarities in order to preserve it, penalising similar nodes that were embedded far apart. Formally, the optimal embedding y can be derived from

$$y^* = \arg\min \sum_{i \neq j} (y_i - y_i)^2 W_{ij} = \arg\min\, y^T L y, \text{ where:} \tag{2.7}$$

- W_{ij} is the similarity between nodes v_i and v_j,
- $L = D - W$ is the graph Laplacian,
- D is the diagonal matrix where $D_{ij} = sum_{j \neq i} W_{ij}$

The most important y_i has the biggest D_{ij}, and the optimal y's are the eigenvectors corresponding to the maximum eigenvalue of the eigenproblem $Wy = \lambda Dy$ [6]. As this formulation only embeds nodes existing in the training set, this graph embedding is transductive. The inductive version, that is, the graph embedding that embeds new unseen nodes includes a linear function $y = X^T \alpha$ to derive embeddings as long as the node feature is provided. In this case, the objective function is:

$$\alpha^* = \arg\min \sum_{i \neq j} ||\alpha^T X_i - \alpha^T X_j||^2 W_{ij} = \arg\min\, \alpha^T X L X^T \alpha \tag{2.8}$$

Adding the constraint $\alpha^t X L X^T \alpha = 1$ to remove an arbitrary scaling factor in the embedding, the problem becomes:

$$\boldsymbol{a}^* = \arg\min \frac{\alpha^T A X L X^T \alpha}{\alpha^T A X D X^T \alpha} = \arg\min \frac{\alpha^T A X W^T \alpha}{\alpha^T A X D^T \alpha} \tag{2.9}$$

The optimal α's are eigenvectors with the maximum eigenvalues for $\alpha^T X W X^T \alpha = \lambda X D X^T \alpha$. The difference in studies results from the calculation of the pairwise node similarity W_{ij} and the use of the linear function $y = X^T \alpha$ [6].

Two classic graph-based methods that effectively leverage structural relationships in natural language data are random walks and label propagation. Random walks capture semantic and syntactic dependencies by traversing a graph, while label propagation

spreads known labels to unlabelled nodes, supporting tasks like classification and clustering.

In NLP, these methods have been applied to various tasks. For instance, random walks on co-occurrence graphs are used to capture semantic similarities between words, improving the quality of word embeddings. The random walk algorithm is not only efficient but also popular, successfully providing information on distance measures between nodes before clustering [40]. In document classification, label propagation allows labels to be transferred from labelled to unlabelled documents by exploiting their structural relationships within a document graph. Similarly, in sentiment analysis, label propagation helps determine the sentiment polarity of text by propagating sentiment labels through graphs representing sentence structures. Additionally, random walks have been used in knowledge graph completion, predicting missing links by navigating the existing graph structure.

These traditional graph-based algorithms succeed in capturing the structural information of graphs. However, they have limited expressive power given that they disregard the node and edge features [67]. Additionally, the lack of a unified learning framework for traditional graph-based algorithms is challenging. Each algorithm possesses distinct properties and requires specific settings, limiting their applicability to diverse NLP tasks[40]. Consequently, despite traditional graph-based algorithms' success in many NLP tasks, they face significant limitations.

2.1.2 Graphs and Deep Learning

Many traditional graph-based methods - like random walk and label propagation - successfully address challenging NLP problems including word-sense disambiguation, name disambiguation, co-reference resolution, sentiment analysis, and text clustering. However, their limitations lead to the exploration of other algorithms to solve NLP tasks, such as traditional machine learning (ML) algorithms.

The major problem of traditional ML algorithms in graphs pertains to the incorporation of structural information. Encoding and modelling the inherent structure and information propagation in graphs is not trivial. This difficulty is related to graph isomorphism, as naive vector representations of graphs often lead to substantial information loss due to the inability to capture the full structural context. Additionally, a direct use of graph isomorphism for graphs with more than a few hundred vertices is infeasible [5]. This intractability severely hinders learning on real-world graphs, which can contain significantly more nodes. For instance, the Oxford English Dictionary alone encompasses

over 170,000 words, highlighting the vast potential node count in NLP tasks.

Graph Neural Networks (GNNs) offer a powerful solution to this challenge. GNNs are a specialised class of neural networks designed to operate on arbitrary graph-structured data. By iteratively aggregating information from neighbouring nodes and updating their own representations, GNNs can effectively learn from the complex relationships within a graph. This capability makes them well-suited for various NLP tasks, including classification tasks - like sentence classification and relation extraction, link prediction -, and graph generation tasks - like machine translation and summarization [67].

Usually, GNNs learn node embedding using the graph structure and input node embeddings in the graph filtering process, which can be denoted as:

$$h_i^{(l)} = f_{\text{filter}}(A, \mathbf{H}^{(l-1)}), \text{ where:} \tag{2.10}$$

- $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix of the graph
- $\mathbf{H}^{(l-1)} = \{\mathbf{h}_1^{(l-1)}, \mathbf{h}_2^{(l-1)}, ..., \mathbf{h}_n^{(l-1)}\} \in \mathbb{R}^{n \times d}$ is the input node embeddings at the $l-1$-th GNN layer
- $\mathbf{H}^{(l)}$ is the updated node embeddings
- d is the dimensionality of the node embedding

In this context, the function $f_{\text{filter}}(\cdot, \cdot)$ acts as a graph filter. The specific models then differ only in how this filter is chosen and parametrised [67]. Importantly, graph filtering refines node representations without altering the underlying graph structure. By stacking L such filtering layers, the model progressively generates the final node embeddings. The graph filtering can be categorised into four types: spectral-based graph filters, spatial-based graph filters, attention-based graph filters and recurrent-based graph filters [67].

Following the graph filtering, the node embeddings are aggregated to generate graph-level embeddings. The graph pooling generates smaller graphs with fewer nodes and their embeddings, based on an input graph and its node embeddings. This operation can be summarised as:

$$\mathbf{A}', \mathbf{H}' = f_{\text{pool}}(\mathbf{A}, \mathbf{H}), \text{ where} \tag{2.11}$$

- $f_{\text{pool}}(.,.)\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{A}' \in \mathbb{R}^{n' \times n'}$ are the adjacency matrices before and after graph pooling, respectively
- $\mathbf{H}' \in \mathbb{R}^{n' \times d'}$ represent the node embeddings before and after graph pooling
- n' is generally set to 1 to obtain the embedding for the entire graph

This operation is crucial for graph-focused downstream tasks like graph classification, where a single graph-level representation is required. The learned node embeddings alone are insufficient for this purpose. Graph pooling techniques aggregate these embeddings to generate a more compact and holistic representation of the entire graph. The graph pooling layers can be classified as flat or hierarchical.

The flat graph pooling generates the graph-level representation directly from the node embeddings in one step. Common implementations include max-pooling and average-pooling, which aggregate node embeddings across the entire graph. A more powerful, albeit less common, option is the BiLSTM aggregation function. It has better expressive power and is not permutation invariant on the set of node embeddings [19] In contrast, the hierarchical graph pooling employs several graph pooling layers, each one following a stack of graph filters. This approach progressively coarsens the graph structure, iteratively collapsing nodes into higher-level representations until a single graph-level embedding is obtained [67].

Graph representation learning aims to incorporate graph structures' information and attributes into a low-dimension embeddings via a machine learning model [19, 67]. This process, unlike graph embedding, which converts a graph into a low dimensional space where the graph information is preserved, leverages both graph analytics and representation learning to extract meaningful information for tasks like classification and prediction. Representation learning obtains data representations that facilitate the extraction of useful information when building classifiers or other predictors. Graph representation learning is distinct from graph embedding as the first does not require the learned representations to be low dimensional [6].

There is no straightforward way to encode the high-dimensional, non-Euclidean information about graph structure into a feature vector [19, 6]. This challenge is further compounded by varying input and output requirements depending on the goal application. The input of graph embedding varies in different scenarios according to its type and the goals with this operation, preserving different information in the embedded space. On the other hand, the graph embedding output is a task-driven low-dimensional vector representing a part or the entirety of the graph. Selecting the most suitable embeddings output type - node embedding, edge embedding, hybrid embedding or whole-graph embedding - for the application of interest is challenging [6]. Different granularities have different criteria for good embeddings and different challenges to be overcome. For example, the most common type of embedding output is node embedding. A good node embedding preserves the similarity to its neighbouring nodes in the embedded space. In contrast, a good whole-graph embedding represents a whole graph as a vector to preserve the graph-level similarity.

Overall, this representation learning can be formalised for the arbitrary graph $G(V, E, T, R)$, where:

- V is the set of nodes in the graph.
- E is the set of edges in the graph.
- $T = \{T_1, T_2, \ldots, T_p\}$ is the collection of all possible node types.
- $R = \{R_1, \ldots, R_q\}$ is the collection of all possible edge types.

The goal is to learn a continuous vector representation $z_S \in \mathbb{R}^d$ of an induced subgraph $G[S]$ of the full graph G, where $S \subseteq V$. Here, V represents the set of all nodes in the graph, as it embeds both subgraphs (where S is a proper subset of V) and entire graphs (where S is equal to V). This z_S embedding can then be used to make predictions about the entire subgraph $G[S]$. Subgraph representation learning is related to the design of graph kernels (which define a distance measure between subgraphs) but differs in its ability to learn feature representations through the embedding process, unlike the pre-specified features used in kernel functions [19]. Furthermore, unlike node embedding, subgraph approaches are fully-supervised, as its goal is to predict a label associated with a specific subgraph [6]. Even so, the methods for subgraphs usually build upon the techniques used to embed individual nodes [67].

Here, $|\cdot|$ denotes the cardinality of a set. We also define two indicator functions:

- $\tau(\cdot) \in T$: This function assigns a node type to a node. For example, $\tau(v_i) \in T$ represents the type of node v_i.
- $\phi(\cdot) \in R$: This function assigns an edge type to an edge. For example, $\phi(e_{i,j}) \in R$ represents the type of edge $e_{i,j}$ connecting nodes v_i and v_j.

This way, we can learn and optimise a mapping that embeds nodes, or entire (sub)graphs, as points in a low-dimensional vector space $\mathbb{R}^d$ that reflects the structure of the original graph. In the case of whole graph embedding, the key challenge is the trade-off between expressiveness and efficiency [6].

GNNs typically construct graphs in two stages: converting edge information to an adjacency matrix A, and node representation learning, given initial node embedding X and adjacency matrix A. While static graph construction can encode prior knowledge of the data in the graph structure, for dynamic graphs the graph structure is usually learned together with the downstream task jointly. This category also includes attention-based and metric learning-based mechanisms to learn the implicit graph structure from unstructured texts with the downstream tasks [67]. The challenge in this approach is capturing the diversity of connectivity patterns observed in graphs, since only structural information is available in homogeneous graphs [6].

If the information is allowed to propagate equally in both directions or discarded in outgoing edges, some important structure information for the final representation learning

will be lost. The common solution for this is to implement bidirectional GNNs, capable of learning the node representation from both incoming and outgoing edges in a interleaved fashion [67]. Not unlike the Bi-LSTM architecture, a general bidirectional GNN$(\cdot)$ that can be easily applied to most existing GNNs first encodes the graph in two directions, separately. Then, the forward and backward directions are concatenated together to be later stacked over several layers to achieve better performance.

For more complicated relations between nodes of the same type, we construct a multi-relational graph with the same node type but different edge types [67]. Here, edges are initially assigned a "default" type. To capture relational information, a reverse edge $e_{j,i}$ with type "reverse" is added for each original edge $e_{i,j}$. Additionally, self-loops with type "self" are added for each node. This results in a multi-relational graph with a single node $|V| = 1$ and a constant number of edges $|E| = 3$. In these graphs, it is challenging to explore the global consistency between different types of objects and to deal with imbalance of objects belonging to different types, as the embedding space is the same for all.

These multi-relational graphs can be represented with multi-relational GNNs or transformers [67]. The multi-relational GNN expands the classic GNN by employing type-specific parameters to individually model the relationships represented by different edge types. Notably, a common challenge in classic GNNs is over-smoothing, which occurs when stacking multiple layers to capture implicit correlations between distant nodes, not directly connected. To avoid the loss of important details due to repeated local aggregation, multi-relational GNNs incorporate a gating mechanism. During message passing, gates combine the node's input features with aggregated features from neighbours. This approach regulates how much information from the update message is propagated to the next layer, preventing the model from completely overwriting past information and preserving crucial details.

A basic example of the gating mechanism can be taken from [58], an extension of Graph Convolutional Networks (GCNs). The update process involves two steps: aggregation and combination. In the aggregation step, the previous node embedding, $h_i^{(k-1)}$, is transformed by the aggregation function $f^{(k)}$ to obtain an update message, $u_i^{(k)}$:

$$u_i^{(k)} = f^{(k)}(h_i^{(k-1)}), \tag{2.12}$$

Next, the combination step determines how much of the update message is integrated with the previous embedding in simple differentiable message-passing framework [58]. This is achieved through a gated combination, where gating vectors $g_i^{(k)}$ control the influence of the update and the past information. These gating vectors are calculated using a linear transformation $f^{(k)}(\cdot) : \mathbb{R}^{2d} \to \mathbb{R}$ of the concatenated update message $u_i^{(k)}$ and previous embedding $h_i^{(k)}$. The sigmoid activation function σ ensures the gating vectors fall within the range [0, 1]. With highly multi-relational data, the number of parameters

grows rapidly with the number of relations in the graph. As such, the weights of the layers must be regularised to avoid overfitting. This entire process can be denoted as:

$$g_i^{(k)} = \sigma(f^{(k)}([u_i^{(k)}, h_i^{(k-1)}])), \tag{2.13}$$

The gating vectors $g_i^{(k)}$ are calculated d times to get the gating vector $\mathrm{g}_i^{(k)}$. The final representation of node i combines the previous embeddings $h_i^{(k)}$ and the update message, scaled by the gating vector, $\sigma(u_i^k)$ as:

$$h_i^{(k)} = \sigma(u_i^{(k)}) \odot g_i^{(k)} + h_i^{(k-1)} \odot (1 - g_i^{(k-1)}), \tag{2.14}$$

where $\sigma(\cdot)$ is an element-wise activation function, often the $tanh(\cdot)$ function. This calculation ensures the gating mechanism can selectively learn to retain or forget information as the network processes the graph. This type of transformation is very effective at accumulating and encoding features from local, structured neighbourhoods, improving the performance in graph classification and graph-based semi-supervised learning.

Even though the gating mechanism improves the GNN's learning, it can be further improved by a structure-aware self-attention mechanism inspired by transformers. Traditional transformers fail to leverage the graph's structural information, but this self-attention mechanism can be seen as a special procedure of fully connected implicit graph learning, making transformers as a special case of message-passing GNN [67]. Furthermore, GNNs can be theoretically understood as a generalisation of graph convolutions or as a continuous and differentiable version of the WL-test, a classic graph isomorphism algorithm [19].

The input of self-attention can be formalised as query matrix $Q = \{q_1, q_2, \ldots, q_m\} \in \mathbb{R}^{m \times d_q}$, key matrix $K = \{k_1, k_2, \ldots, k_n\} \in \mathbb{R}^{n \times d_k}$ and value matrix $V = \{v_1, v_2, \ldots, v_n\} \in \mathbb{R}^{n \times d_v}$. For a graph transformer, the query, key, and value all refer to the nodes' embeddings: namely, $q_i = k_i = v_i = h_i$. And the output representation z_i is calculated as:

$$z_i = \text{Attention}(q_i, K, V) = \sum_{j=1}^{n} \alpha_{i,j} W_v v_j, \tag{2.15}$$

where $\alpha_{i,j}$ is the attention score calculated using softmax:

$$\alpha_{i,j} = \text{softmax}_j(u_{i,j}) \tag{2.16}$$

and $u_{i,j}$ is the attention weight:

$$u_{i,j} = (W_q q_i)^T (W_k k_j) \sqrt{d_k} \tag{2.17}$$

Here, $W_q \in \mathbb{R}^{d \times d_q}$, $W_k \in \mathbb{R}^{d \times d_k}$, and $W_v \in \mathbb{R}^{d \times d_v}$ are trainable parameters, and d is the model dimension.

Given the learnt attention for each relation, edge embedding representation is the next critical step for incorporating the structure-information. An alternative to the explicit encoding involves learning a relative position encoding based on the absolute position of nodes [67]. This is achieved with a set ($[-K, K]$) of $2K + 1$ latent labels, projecting the difference between node positions $j - i$ to one specific label embedding for the node pair (vi, vj) to retrieve the edge embedding $e_{i,j}$.

For heterogeneous networks, besides the relation graph transformation, there are two typical graph representation learning methods, one based on meta paths and the other on type-specific aggregation [67]. The challenge here is incorporating rich and unstructured information so that the learnt embeddings both represent the topological structure and discriminate nodes with its auxiliary information [6]. This auxiliary information can be labels, attributes, mode features and information propagation - valuable information in the context of NLP problems.

Given the great power of GNNs for modelling graph-structured data, many efforts have been made to develop GNN-based encoder-decoder frameworks [67]. Formally, the encoder is a function ENC : $V \rightarrow \mathbb{R}^d$ where $z_i \in \mathbb{R}^d$ represents the embedding for node $v_i \in V$. This function maps nodes to real-valued vector embeddings of dimension d. The decoder, denoted as DEC, is a function that decodes user-specified graph statistics from a set of node embeddings. While various decoders are possible, the majority of works employ a basic pairwise decoder DEC : $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$.

With the pairwise decoder, we get a reconstruction of v_i and v_j's similarity in the original graph in a a real-valued node similarity measure. For the reconstruction loss that optimizes the decoder mappings, $s_G(v_i, v_j)$ denotes a user-defined, graph-based similarity measure between nodes v_i and v_j within the graph G. There are three proximity measures, $s_G : V \times V \rightarrow \mathbb{R}^+$, usually adopted to quantify the graph property to be preserved in the embedding space [6]. The first-order proximity is the local pairwise similarity between nodes directly connected by edges, namely the weight of the edge $e_{i,j}$ between nodes v_i and v_j, that is, $A_{i,j}$. The second-order proximity compares the similarity of the nodes' neighbourhood structures as $s_{ij}^{(2)}$ between node v_i's neighbourhood $s_i^{(1)}$ and v_j's $s_j^{(1)}$. Finally, the higher-order proximity can be defined as the k-th-order proximity between nodes v_i and v_j: the similarity between $s_i^{(k-1)}$ and $s_j^{(k-1)}$. These higher-order proximities can also be defined with other metrics, such as Rooted PageRank and Katz Index [6].

Additionally, in works of graph embedding with edge reconstruction based optimisation, the first and second-order proximities are empirically calculated based on the joint probability and conditional probability of two nodes. The empirical first-order proximity between nodes v_i and v_j is $\hat{p}^{(1)}(v_i, v_j) = \frac{A_{ij}}{\sum_{e_{ij} \in E} A_{ij}}$, where A_{ij} represents the weight of edge e_{ij}. Here, a smaller distance between the true probability $\hat{p}^{(1)}(v_i, v_j)$ and the empirical probability $\hat{p}^{(1)}(v_i, v_j)$ indicates better preservation of first-order proximity [6]. Using KL-divergence as the distance function between $p^{(1)}$ and $\hat{p}^1$ and omitting constant terms, the

objective function to minimise for preserving first-order proximity in graph embedding becomes:

$$O_m^{(1)}in = \min - \sum_{e_{ij} \in E} A_{ij} \log p^{(1)}(v_i, v_j) \tag{2.18}$$

Similarly, the second-order proximity between v_i and v_j refers to the conditional probability of generating node v_j given node v_i. The empirical second-order proximity is calculated as $\hat{p}^{(2)}(v_j|v_i) = \frac{A_{ij}}{d_i}$, where $d_i = \sum_{e_{ik} \in E} A_{ik}$ represents the degree of node v_i. As this is expensive to calculate, negative sampling is adopted for approximate computation [6]. Minimising the KL divergence between $p^{(2)}(v_i|v_j)$ and $\hat{p}^{(2)}(v_i|v_j)$, the objective function becomes:

$$O_m^{(2)}in = \min - \sum_{e_{ij} \in E} A_{ij} \log p^{(2)}(v_i|v_j) \tag{2.19}$$

Alternatively, similarity can be defined based on co-occurrence probability during random walks, defining s_G according to the probability of v_i and v_j co-occurring on a fixed-length random walk over graph G [19].

In practice, the reconstruction objective is achieved by minimising an empirical loss function L over a set of training node pairs D:

$$L = \sum_{(v_i, v_j) \in D} \mathcal{L}(\text{DEC}(z_i, z_j), s_G(v_i, v_j)), \tag{2.20}$$

where $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a user-specified loss function for the discrepancy between the decoded similarity values $\text{DEC}(z_i, z_j)$ and the true values $s_G(v_i, v_j)$. With an optimised encoder-decoder system, we can generate embeddings for nodes to use as feature inputs for downstream machine learning tasks.

Overall, GNNs learn node-level representations through message passing and aggregation from neighbours, adaptively learning correlations between a node and its neighbour via the attention mechanism. Some shortcomings of GNNs include the degradation of the model due to non-useful information in features of a neighbourhood and GNN's reliance on message passing, making it prone to noise and adversarial attacks [69]. The scalability of GNNs is gained at the price of corrupting graph completeness. Graph information is typically lost, regardless of the use of sampling or clustering. Sampling may lose node's influential neighbours and clustering may deprive the graph of distinct structural patterns. As such, the trade-off between algorithm scalability and graph integrity remains an open issue. Another shortcoming comes from the assumption of the majority of current GNNs of homogeneous graphs.

Despite these expressiveness issues, GNNs have demonstrated impressive performance, leveraging information from both the node features and the graph structure itself. The intersection of graph theory and deep learning has produced new architectures that

leverage the power of graph representations for tasks such as classification and entity recognition. GNNs extend traditional deep learning approaches by incorporating relational data directly into the learning process, which lays a foundation for much needed interpretable modelling.

2.1.3 Graphormer

Graphormer is a novel approach to graph representation learning, introduced by [71], which combines graph structural encoding with transformer-based attention mechanisms. It addresses the expressiveness limitations of traditional GNNs by incorporating spatial encoding based on shortest path distances between nodes, thereby preserving crucial topological information. Notably, Graphormer demonstrates greater discriminative power than the Weisfeiler-Lehman (WL) test for graph isomorphism [71], making it both effective and efficient for graph isomorphism classification. This also makes it more expressive than GNNS, which are at most as powerful as the WL test in distinguishing different graphs when their aggregation functions and the readout functions are injective [67]. By integrating spatial encoding with the attention mechanism, Graphormer can distinguish between various graph structures, effectively simulating the neighbourhood aggregation step of the WL test. This allows the model to better capture both local and global graph structures, facilitating the learning of complex patterns and the handling of diverse graph sizes and structures.

Graphormer extends traditional transformer models by incorporating information from external knowledge graphs or contextual graphs, allowing for the modelling of complex, long-range relationships between words that may not be captured by linear sequence analysis alone. The core innovation of Graphormer lies in its self-attention mechanism, adapted from transformer architectures, which can be formally expressed as:

$$A_{ij} = \text{softmax}\left(\frac{Q_i K_j^T}{\sqrt{d}} + b(\phi(i,j)) + c(e_{ij})\right)$$

Here, A_{ij} represents the attention weight from node i to node j, Q_i and K_j are the query and key vectors respectively, d is the dimension of the attention heads, $b(\phi(i,j))$ is the spatial encoding bias term based on the shortest path distance between nodes i and j, and $c(e_{ij})$ is the edge feature encoding term. Therefore, this formulation incorporates both the spatial relationships between nodes and the features of the edges connecting them, providing a more comprehensive representation of the graph structure within the attention mechanism. Additionally, the model leverages degree centrality, enhanced with

node feature information, as expressed by: $H_i^{(0)} = x_i + z^-_{\deg^-(v_i)} + z^+_{\deg^+(v_i)}$.

Popular GNN models were shown to be Graphormer's special cases [71]. Graphormer can simulate the AGGREGATE and COMBINE operations of widely-used GNNs, including GIN, GCN, and GraphSAGE, by selecting appropriate weights and distance functions. Graphormer's self-attention mechanism inherently fulfills the role of virtual nodes, often used to enhance GNN performance by aggregating and redistributing global graph information. While virtual nodes risk over-smoothing, Graphormer's self-attention achieves scalable graph-level readouts without this drawback, making the architecture robust and efficient for large-scale applications.

Besides that, with spatial encoding, Graphormer's self-attention can distinguish neighbours, enabling softmax to compute mean statistics over these sets. Combining multiple attention heads and feed-forward networks (FFNs) allows for the separate processing of node and neighbour representations before integration. Consequently, Graphormer surpasses traditional message-passing GNNs, which are limited by the 1-Weisfeiler-Lehman (WL) test, in distinguishing graph structures [67].

As such, the significance of Graphormer lies in its ability to exploit the rich structural information present in language, capturing not only the sequential order of words but also their semantic relationships. This improved performance was proved in graph tasks in multiple areas [31, 65, 23], ranging from image reconstruction to recommendation systems and extractive multimodal summarization. Such capability is crucial for NLP tasks such as sentiment analysis, question answering, and text summarization. Its use in hate speech detection was first proposed for the anticipation of hateful content in online boards [20] as a node-prediction task, differently from our proposal.

While Graph Transformers are a recent development in the field, with various architectures proposed to incorporate graph information into the Transformer framework, Graphormer stands out. These Graph Transformers typically incorporate the graph information into the Transformer in three manners: GNNs as Auxiliary Modules, Improved Positional Embedding from Graphs, and Improved Attention Matrix from Graphs [41]. Graphormer benefits from the last two, as it enriches node representations through degree-aware positional encoding, and it refines the attention mechanism by incorporating both spatial information (via shortest path distances) and edge characteristics (through learnable edge feature embeddings).

2.2 Natural Language Processing

This section introduces key concepts in NLP relevant to text classification, focusing on how textual data is represented, classified, and interpreted. The challenging tasks chosen here are hate speech classification and suicidal posts classification.

The detection of hate speech involves complex sociocultural contexts and evolving linguistic patterns that make rigid classification challenging. What constitutes hate speech can vary across communities and time periods, requiring regular revalidation of both datasets and detection approaches. Additionally, automated systems risk perpetuating existing societal biases or disproportionately affecting marginalized communities through false positives.

Similarly, the identification of suicidal ideation in text requires extreme care, as false classifications in either direction can have severe consequences. Over-identification risks unnecessary interventions and potential privacy violations, while under-identification could miss crucial opportunities for support. We emphasize that any automated system should be viewed as a supplementary tool to support, rather than replace, human judgment in crisis intervention contexts.

2.2.1 Text Representation

Textual data cannot be directly read by machine for model training. As such, the text representation converts text into a numerical representation. This text input can also be preprocessed. This preprocessing usually involves normalisation techniques like lowercasing and punctuation removal, tokenization to separate the text into individual units, stopword removal, and text cleaning. The text representation itself can be done with techniques like word embeddings, which capture the semantic meaning of words, and contextual embeddings, such as those produced by BERT. These embeddings allow models to understand language nuances in ways that traditional representations cannot.

Term Frequency and Inverse Document Frequency (TF-IDF) is a statistical measure used to evaluate the importance of a word in a document relative to a corpus with simplicity and interpretability. TF-IDF is calculated by multiplying the term frequency of a word in a document by the inverse document frequency of the word across the corpus, offsetting the frequency of a term with its frequency in the corpus [15]. As such, unlike a bag of words or n-grams, this method compensates for the fact that some words appear more frequently in general. However, TF-IDF lacks the capability of learning the

sequential relationship patterns between words and textual documents, failing to preserve the rich semantic information between compound words and phrases. This limitation also extends to potential biases in classification models by assigning higher weights to rare words that are not necessarily informative.

Word2vec [39] is an iteration-based method, that is, it captures co-occurrences of words one at a time and, eventually, encodes the probability of a word given its context. This method typically uses two main architectures: continuous bag-of-words (CBOW) and skip-gram. CBOW predicts a centre word from the surrounding context in terms of word vectors. Skip-gram does the opposite, predicting the distribution of context words from a centre word. These word embedding vectors carry rich local-based contextual information as distinctive features. These features can effectively handle some NLP tasks, including the classification task. Word2Vec has limitations, such as its inability to capture the contextual meaning of words and its tendency to produce gender and racial biases. Moreover, Word2Vec may not be suitable for tasks that require understanding the overall structure or meaning of a text, as it primarily focuses on word-level representations [49].

Advanced neural architectures, such as sequence-to-sequence and Transformers, have significantly improved text analysis and representation learning. Sequence-to-Sequence (seq2seq) learning is an encoder-decoder framework that maps a variable-length input sequence to a variable-length output sequence. It typically uses a RNN-based encoder to read the input sequence to build up a fixed-dimensional vector representation. Subsequently, a RNN-based decoder generates the output sequence conditioned on the encoder output vector, one token at a time. One common Seq2Seq variant employs a bidirectional LSTM encoder to encode the input sequence and a LSTM decoder or Transformer model to decode the output sequence [49].

Transformers [62] and their derivative models like BERT (Bidirectional Encoder Representations from Transformers) [13] have become the state-of-the-art for hate speech classification and other NLP tasks. Their encoder-decoder architecture and attention mechanisms excel at modelling long-range dependencies in text, handling varying sequence lengths, and capturing contextual information more effectively than previous deep learning models. The self-attention mechanism enables these models to dynamically focus on different parts of the input sequence, capturing dependencies regardless of their distance within the text. Transformer-based models have significantly improved text classification performance by enhancing textual representations. However, their focus on local sequential relationships can limit their ability to capture long-range dependencies and global semantics, particularly in complex text corpora [49].

BERT further enhances transformer-based models' capability by employing bidirectional training, allowing it to understand context from both left and right sides of a word. This enables these models to attend to information from multiple representation subspaces, enhancing their ability to understand context and nuanced linguistic patterns,

such as hate speech. Transformers and BERT models are typically pre-trained on large-scale corpora, creating general-purpose language representations that can be fine-tuned for specific tasks, generalising well across tasks and domains. This approach has proven particularly effective for complex language understanding tasks, including hate speech detection. Not only that, but BERT is considered as the most powerful pre-trained language model [49].

BERT embeddings can be further improved regarding efficiency with quantization [13]. It involves reducing the precision of numerical values, typically from 32-bit floating-point to 8-bit integer or lower. This can significantly reduce the memory footprint and computational cost of storing and processing them, making them more efficient for deployment in resource-constrained environments. Experiments have demonstrated that quantization can maintain high performance while accelerating computation and saving on memory, storage, and costs. For example, the quantisation of float32 embeddings to binary is simple, thresholding normalised embeddings at 0:

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

Deriving a sentence representation from individual word embeddings can be simple, employing various techniques like averaging, pooling, or sequence modelling. The resulting sentence embedding encapsulates the meaning and context of the entire input text, preserving important information that can be used for downstream tasks. With this sentence embedding in hand, a classification model can be built to predict the probability of the input belonging to a particular class. A common architecture for this task includes, after the embedding layer and the sentence encoder, a classification layer. This layer is fully connected, mapping the sentence embedding to a fixed-dimensional vector space, typically with a number of units equal to the number of target classes. Then, a sigmoid activation function is often applied to the output of the fully connected layer to produce a probability value between 0 and 1, representing the likelihood of the input belonging to the positive class.

During its training, the model learns to adjust the parameters of the fully connected layer and the sentence encoder to minimize the classification loss between the predicted probabilities and the true labels in the training data. The loss function can be the cross-entropy loss (CE), a theoretically grounded and noise-robust loss function [74]. This architecture allows the model to learn effective sentence representations and use them to make accurate classification predictions, which is a common approach in text classification tasks.

2.2.2 Text Classification and Explainability

The State of the Art (SOTA) for text classification usually is a fine-tuned Transformer model, commonly a BERT variant [49]. However, as with most Deep Learning models, the results are not, on its own, explainable. In order to better understand these models, there are both ante and post-hoc approaches to explainability, that is, the ability to understand the reasoning behind a model's predictions. This explainability is essential to understand a model's decision-making process, especially in sensitive areas like hate speech detection.

Ante-hoc explanations are generated during the model training process. These methods involve incorporating interpretability into the model architecture or training objective. For instance, attention mechanisms in neural networks can highlight the specific parts of the input sequence that contribute most to the prediction. Similarly, rule-based systems explicitly represent the decision-making process as a set of rules, making the reasoning process transparent. While ante hoc methods offer inherent interpretability, they may limit the model's performance compared to more complex models.

Post-hoc explainability, on the other hand, focuses on providing explanations for the predictions of black-box models after they have been trained. This approach involves various techniques such as LIME, SHAP, and saliency maps. These methods can identify the most influential features that contributed to a particular prediction, offering insights into the model's reasoning. While post hoc explanations can provide valuable information, they may not always be accurate or complete, especially for complex models. This is the case for GNNs, where post-hoc explanations are shown to be suboptimal in interpretation and sensitive to pre-trained GNNs performance [10].

Another approach to explainability consists in the use of models that are, themselves, more explicable. This can be exemplified with two methods used in text classification: compression and graphs. For compression models, please refer to Appendix A.

As explored in section 2.1.1, graphs effectively formalise text relations. Additionally, GNNs strengthen the representations achieved, leading to improved performance on NLP tasks. GNNs often demonstrate strong generalisation capabilities to unseen or Out-of-Distribution (OOD) graphs, accompanied by interpretations of the predictions. GNN architectures are often considered more interpretable due to their ability to learn about entities, their relationships, and the rules governing their interactions [68].

GNN inference, which propagates information through graph connections, offers a degree of interpretability by revealing explicit reasoning paths or subgraphs that contribute to predictions. This explicit use of graph structure lends itself to interpretability and facilitates a deeper understanding of the model's decision-making process. However,

GNNs can still exhibit black-box characteristics due to the underlying convolution operations, making it challenging to fully understand the information processing in intermediate layers [68]. GNNs have different information propagation subgraphs contributing to the final prediction. As GNNs do not directly provide the most important reasoning paths for its prediction, post-hoc interpretation methods are still needed.

An intrinsically explainable GNN $f := f_c \circ g$ aims to identify a (causal) subgraph $G_c \subseteq G$ via a subgraph extractor GNN $g : G \to G_c$, given samples from training data $D_{tr} = (G_i, Y_i)$, where Y is the space of the labels. Explainable GNNs (XGNNs) often employ a two-step approach: subgraph extraction and prediction [10]. The attention mechanism is used to learn sampling probabilities, ensuring that extracted subgraphs are representative. Predictions are made through weighted message passing based on attention scores. While this paradigm enhances interpretability and OOD generalisation, there is a lack of theoretical understanding regarding its representational properties and limitations, particularly in terms of providing faithful interpretations.

The relationships between nodes and edges in the graph can be visualised and analysed to gain insights into how the model is processing information. Graphormer's sensitivity to structural properties enables predictions based on structural similarities or differences between the input graph and known examples.

Graphormer uses both interpretable modelling approaches for GNNs, namely the attention mechanism and the disentangled representation learning on graphs [68]. The self-attention mechanism enables the model to dynamically focus on relevant nodes and edges, capturing long-range dependencies and contextual information within the graph. Graphormer also implicitly learns disentangled representations by using separate positional embeddings for nodes and edges. These positional embeddings help the model distinguish between the structural properties of nodes and edges, enabling it to learn separate representations for these aspects of the graph.

Furthermore, Graphormer's connection to the WL-test can provide additional insights into its interpretability. The WL-test can help identify important features and generate counterfactual explanations, which can aid in understanding the model's decision-making process. By systematically modifying the input graph and observing how the WL-test results change, we can understand how changes in the graph's structure affect the model's predictions. This can provide valuable insights into the causal relationships between the graph's features and the model's output. Additionally, visualising attention weights can reveal the importance of specific nodes and edges in shaping predictions.

Pursuing more interpretable and explainable models is crucial, and architectures like Graphormer represent an essential step forward in this effort. Additionally, these interpretable models can be used to help explain another model's results to its end users, allowing for both the use of more powerful models and the necessary pursuit of explainability.

2.2.3 Hate Speech and its Detection

There is no consensus on the parameters of what constitutes hate speech [72], be it legally, for companies, or for ethics. Hate speech here is defined as abusive communication targeting specific marginalized groups, such as those of ethnicity, nationality religion, gender, or sexual orientation [14]. This impacts the well-being of targeted and non-targeted people reached by the message and can often be a pre-cursor to, or an extension of, offline hate crime, intensifying its repercussions [47]. The identification of this type of speech is complicated by a myriad of factors like its textual and discourse context, timing of posting and the identity of the author [59]. Most works in hate classification involve supervised learning techniques [59], whether they focus on user-based or content-based classification. However, the addition of more context to hate speech classification is not a given. [48] found no evidence that context improves the performance of toxicity classifiers, but also indicates that this seems to be related to the fact that context-sensitive comments are infrequent in their collected data.

A token-based approach to the classification of hateful texts is unable to handle the alternative spelling of slurs [37] and the use of hate codes that are known in-group to symbolise a determined group or even covert communication [35, 47]. To further complicate the classification of hateful content, some online groups encourage the use of slurs as community lingo[12]. For example, well-integrated users of 4chan, a fringe social media forum, will identify new users unfamiliar with the forum's lingo and call them out as newf*gs[1].

Even though the ironic use of slurs or hateful memes templates is still reprehensible for normalising hate speech [12], its use in 4chan's context is generalised enough that separating its ironic use - especially for hateful meme variants - might provide a better classification of unironic hate speech in the platform. The differentiation of slurs is not novel to the literature [50], as the target and the offensiveness of slurs is not monolithic. Slurs that not only seek to describe oppression, but also seek to actively create it and maintain it, are more offensive, such as the n-word. In this way, not unlike the reclamation of the n-word by the black community, the use of created words composed with the ending -fag by the 4chan community acquires a new convention of use that serves a different, not hateful purpose for the in-group community. These uses work in the in-group as a marker of belonging, but the fact that a slur is used by in-group members does not automatically make it inoffensive.

This irony or sarcasm can be detected [64, 51] by identifying the juxtaposition of hateful elements and their supposed opposites, that is, elements that in a intentionally

[1]In no way do these terms represent the author's perspective. However, the exemplification and explanation of hateful monikers is necessary to this dissertation's subject.

hateful message would not be combined. There are also online markers used to explicitly denote sarcasm, like "\s". This complex linguistic phenomena, so often found in hate speech, is pointed as a possible improvement to hate speech classification [66, 22, 18, 70].

Despite the fuzziness of defining sarcasm [9], here it is understood as the reversal of the apparent original meaning of a phrase, marked by the incongruity of self-contradiction by combining contrasting words or by the use of expressions such as "imagine" prefacing common sense utterances and "\s" prefacing absurd statements. This does not encompass all possible scenarios, but it does capture more evident instances in internet comments. This addition refines hate speech detection as the reversal of meaning can transform a seemingly normal phrase into a hateful one. This is well exemplified by a phrase found in the analysed dataset, altered to exclude which group it is directed to: "imagine thinking that killing a certain group is bad". The reversal of "imagine thinking" to a common sense phrase changes the comment's intended meaning and such a phrase should be detected as hateful.

2.2.4 Suicidal Posts Detection

According to the World Health Organization (WHO), every year, about 700,000 million people take their own lives worldwide [45]. Suicide is the leading cause of death among those aged between 15 and 39, with an estimated 10–20 million non-fatal attempted suicides occurring every year. Despite mental health issues being such a pressing problem, their treatment is troubled by insufficient healthcare systems and the stigma related to these diseases. The study of suicide is further complicated by the unavailability of quality data on the issue [45].

Despite this difficulty in accessing data, social media contain a wealth of mental health data. Online mental health support communities offer support for issues ranging from anxiety to suicidal attempts [34]. Studies on these communities report a range of benefits to the individual. These include the supportive atmosphere created by site moderators and other forum members, the accessibility and utility of these communities, and the opportunity that online environments offer individuals to talk about their feelings and experiences with similar others without being judged [56].

In this context, the use of NLP to analyse social media posts for the study of mental health has gained popularity. A popular approaches is the Linguistic Inquiry and Word Count (LIWC), focused on analysing text using predefined dictionaries of words associated with various psychological categories (such as anxiety, depression, or positive emotions), validated by psychology research [30, 34]. More traditional NLP techniques

employ deep learning models and transformer architectures to detect subtle linguistic patterns, emotional content.

Chapter 3

Methods

3.1 Our Proposal: Rhetoric Graphormers

Our adaptation of the Graphormer architecture, as seen in Figure 3.1 is specifically designed for the task of textual graph classification, where each word or sub-word is represented as a node, and the entire comment is represented as a graph with a corresponding label. In this context, the quality of node representations depends on the chosen embedding technique. By employing BERT as our embedding method, we leverage its word-piece tokenizer, which is particularly effective in understanding in-group hate codes that often rely on neologisms or made-up words.

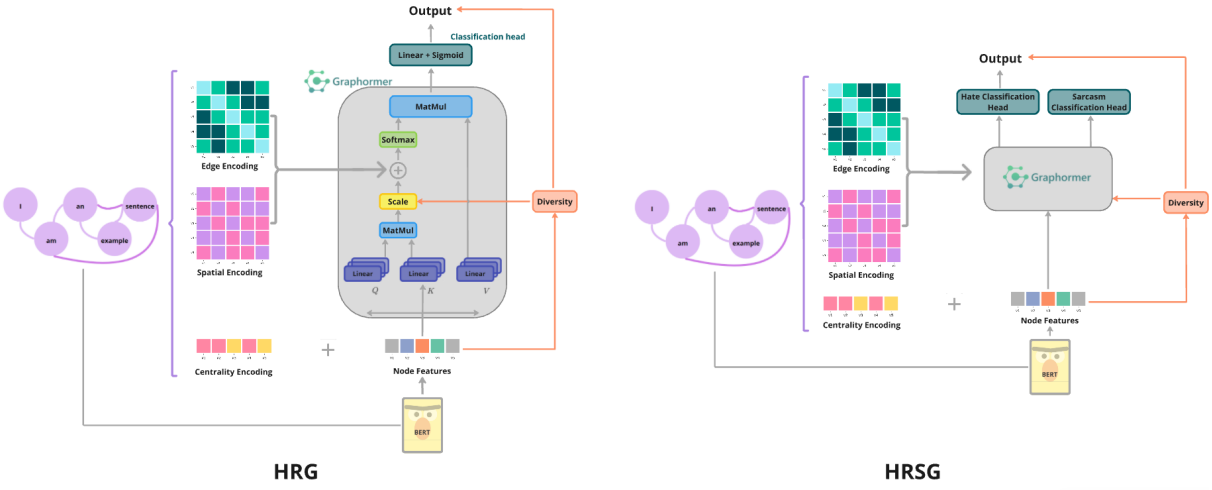

Figure 3.1: Depicted on the left, the Hate Rhetoric Graphormer (HRG) and Suicidal Rhetoric Graphormer (SRG) architectures use the text's graph and BERT encodings to train the Graphormer to classify hate speech, with the inclusion of diversity. The Hate and Sarcasm Rhetoric Graphormer (HSRG) has 2 classification heads, one for the main task of hate speech detection and another for sarcasm detection.

BERT's pre-training on a large corpus of text provides a solid foundation, ensuring that our model benefits from its rich understanding of language patterns, syntactic structures, and semantic relationships. This foundational knowledge is essential for com-

prehending the nuances of hate speech, which often relies on subtle linguistic cues in a specific context. Furthermore, the use of BERT-like encodings opens up the possibility of pre-training our model on unlabelled data from the specific domain of interest. This domain-specific adaptation can help the model learn the unique vocabulary and language patterns of hate speech prevalent in the target domain.

With these node representations, the graph is then constructed with sequential edges and dependency parsing to represent the connections. This allows for a robust representation that is less computationally expensive than complete graphs. The graph's ensuing topological properties are preserved by three elements incorporated to the transformer's architecture: the centrality encoding (calculated similarly to the degree centrality), the edge encoding and the spatial encoding (defined by the distance of the shortest path between two connected nodes). Similar to how molecules can be comprised of the same nodes but have distinct behaviours and, as such, labels, this modelling aims to feasibly capture the complex relations between words in different contexts. The learning focuses on improving the contextual word's representation to improve hate speech detection.

After the Graphormer training phase is completed, the output is passed through a classification head to identify hate speech or suicidal posts. A combined loss function is used, merging cross-entropy with a diversity-based loss to balance training objectives. In this process, diversity scores modify attention weights by projection and scaling, allowing the Rhetoric Graphormers to account for variability in node characteristics.

The diversity measure used was first proposed by Schieber and collaborators [57]. The D-measure addresses a critical limitation in other network comparison methodologies: the failure to capture nuanced topological differences, particularly in complex network structures. This measure is a three-term function that compares graphs using probability distribution functions (PDFs) derived from network properties. As such, the metric combines three key analytical components: global network distance distributions, node connectivity heterogeneity, and node centrality patterns. First, the method compares averaged connectivity patterns, capturing global topological features such as network diameter and path length. Second, it analyses the Network Node Dispersion (NND), which quantifies the heterogeneity of node connectivity across the network using the Jensen–Shannon divergence among N PDFs [57, 7]. A network is considered heterogeneous if it exhibits a high diversity of node-distance patterns, resulting in a high NND value. The third component examines node centrality, considering both direct and indirect connectivity spans, including the graph's complement to capture disconnection effects.

Computationally, the D-measure offers significant advantages. Unlike graph edit distance, which is computationally expensive and NP-hard, this approach achieves polynomial time complexity. The metric returns a value between 0 and 1, with 0 indicating isomorphic graphs and increasing values representing growing topological differences. This

measure consistently and accurately distinguishes non-isomorphic structures [57]. This approach is particularly powerful in detecting subtle structural variations that previous methods overlooked, such as network disconnections and connectivity patterns.

This diversity-based component in the Graphormer enhances feature discrimination by refining the representations for different content types, thereby helping the model capture subtle linguistic variations between hate and non-hate speech. This approach reduces the risk of overfitting to specific patterns and supports the learning of more robust and transferable features. Moreover, diversity scores allow for greater interpretability, offering insights into model decisions and enabling closer monitoring of how the model differentiates between content types. Overall, this methodology establishes a comprehensive classification system that not only detects hate speech but also provides valuable insights into the structural and semantic patterns underlying different types of content. The integration of diversity-aware components enhances the model's sensitivity to language nuances, while the training process ensures robust and generalizable learning.

Additionally, for our multi-target model (HSRG), the classification of sarcasm aims to improve the hate speech classification. Sarcasm consists of the reversal of the apparent original meaning of a phrase [9], marked by the incongruity of self-contradiction by combining contrasting words or by the use of expressions such as "imagine" prefacing common sense utterances and "\s" as a commentator's self-annotation of sarcasm. This addition refines hate speech detection as the reversal of meaning can transform a seemingly normal phrase into a hateful one. This is well exemplified by a phrase found in the dataset: "imagine thinking that killing Muslims is bad". The reversal created by "imagine thinking" changes the comment's intended meaning from benign to hateful, and such a phrase should be detected as a hate rhetoric example. To incorporate sarcasm, the HSRG adds a classification head for sarcasm detection next to the hate speech classification head. Even though hate and sarcasm detection are still two different tasks, the alterations in the text representation for sarcasm detection will also influence the hate detection task.

The use of Graphormer for hate speech detection is not novel. Hebert and collaborators [20] have first proposed its use representing comments as nodes and edges as reply relationships. However, this work defines hate speech more broadly, including offensive comments directed at non-marginalized groups - for example, comments directed at democrats or republicans. As such, this work lacks the focus on targeted and intentional hate rhetoric directed towards minorities.

Moreover, the task in this work is node classification, considering entire comments as nodes, and the chain of comments as the graph. This allows for a method that aims to proactively predict hate speech based on the prevalence and encouragement of subsequent hate speech. However, this method focuses on predicting if future comments might be hateful, not on the texts' content themselves.

The addition of sarcasm detection to hate speech detection tasks was also previ-

ously explored to improve the detection of more implicit hate speech [70]. This shows promising improvements to hate detection. In a context such as a fringe network, where sarcasm is prevalent, adding it to hate detection may improve the proposed framework.

Overall, our approach is a novel implementation of a diversity-aware Graphormer for text classification. Using an intrinsically interpretable architecture, we investigate the influence of the diversity awareness in the Graphormer.

3.2 The datasets

In this section, we present the publicly available datasets used, divided by task. Both tasks - hate speech detection and suicidal ideation identification - address critical social challenges that require careful consideration of ethical implications throughout the research process. While computational approaches offer promising avenues for early intervention and content moderation, we acknowledge that automated classification of such sensitive content carries inherent risks and limitations that must be carefully weighed against potential benefits.

Regular evaluation and updating of both data and classification approaches is essential to maintain effectiveness and ethical alignment with evolving societal standards and needs. The following subsections detail the specific characteristics of each dataset while highlighting their contributions and limitations.

3.2.1 Hate speech detection

We use 3 corpus to train and evaluate our models. To understand the nuances of intended hate speech, we train our models using the Slur corpus. For sarcasm detection, the iSarcasm corpus is employed. Model evaluation is performed using the "Raiders of the Last Kek" dataset, which contains comments from 4chan, a fringe platform. This dataset also facilitates comparison of domain-specific pre-training, addressing the lack of reliable annotated corpora for fringe social networks with distinct vocabularies.

Kurrek and collaborators [26] introduced a 39.8k human annotated corpus with posts from Reddit, designed to aid in the training and evaluation of hate speech detection models. This corpus is particularly valuable for addressing the challenge of overfitting to pejorative language. The annotation schema employs a comprehensive taxonomy with

four primary categories: derogatory, non-derogatory non-appropriative, homonym, and appropriative. The diverse group of annotators responsible for classifying the corpus introduced variability in labelling, which aligns with the definition of hate speech as abusive communication targeting marginalized groups.

A reliable and comprehensive annotated corpus is crucial for evaluating the performance of hate speech classification models. It allows for a more nuanced understanding of slur usage, preventing oversimplification and enabling a more sophisticated representation. For instance, a sentence that explicitly denounces a slur should not be classified as hate speech, even if it includes the slur itself. A simple word filter would incorrectly flag such a sentence.

Unlike real-world hate speech, which is often imbalanced [54], this dataset is balanced, ensuring that both the training and test sets maintain this characteristic. Approximately 80% of the corpus exhibits inter-annotator agreement, from which the training and test sets are extracted. Training on balanced datasets offers several advantages. Firstly, it mitigates bias towards the majority class, leading to more equitable model performance. Secondly, it enhances the model's ability to recognize patterns and features associated with minority classes, improving generalization to unseen data and mitigating overfitting. Finally, it contributes to more robust decision boundaries, reducing the likelihood of misclassification.

The iSarcasm corpus was proposed for SemEval-2022 Task 6 [1]: detecting intended sarcasm. The sarcasm label of texts, as well as their non-sarcastic rephrases, are provided by the authors of those texts. This collection aims to minimise the disadvantages of other collection methods, which rely on self-annotation (with the use of "\s" or third-party annotations. The training set contains 867 pairs of sarcastic texts and their rephrasing, and other 2,601 sarcastic and non-sarcastic texts. There is also a separate test set with 200 sarcastic texts, and a total of 1,200 non-sarcastic texts.

Another dataset represents fringe social platforms, ecosystems with little to no moderation[46] that amplify unsavoury online behaviour, like the dissemination of fake news and the emergence and evolution of hateful memes. One popular example of these platforms is 4chan, an imageboard organised by threads on which users can anonymously post text and images and interact by replying to these posts. The existing moderation is hard to enforce, as the absence of user registration makes bans unfeasible, as they can be evaded with the simple use of a new IP or proxy. This scenario is further complicated by the commonplace ironic use of hateful monikers [12] and the platforms' offensive social group language. Content from this platform, in the form of radicalised ideas, conspiracy theories or hateful memes, tends to branch out to other more mainstream social platforms, such as Twitter and Facebook [63]. Even though mainstream platforms have more rigorous moderation standards, hate speech can still proliferate in them to a greater extent, as well as on messaging apps - which don't have moderation teams.

To further complicate the classification of hateful content, on 4chan the use of slurs as community lingo is encouraged [12]. For example, well-integrated users will identify new users unfamiliar with the forum's lingo and call them out as newf*gs[1]. Even though the ironic use of slurs or hateful memes templates is still reprehensible for normalising hate speech [12], its use in 4chan's context is generalised enough that separating its ironic use - especially for hateful meme variants - might provide a better classification of unironic hate speech in the platform. This irony or sarcasm can be detected [64, 51] by identifying the juxtaposition of hateful elements and their supposed opposites, that is, elements that in a intentionally hateful message would not be combined.

To study such an environment, the dataset of [46] was used, alongside referenced images extracted from 4plebs, a website dedicated to archiving 4chan threads. This dataset was collected from June, 2016 to July, 2017, with 4.3 million images shared on /pol/, amounting to 12.5 million image-text pairs, as presented in [52].

3.2.2 Suicidal posts detection

We use the Reddit Mental Health Dataset [34], a publicly available dataset including posts from 826,961 unique users from 2018 to 2020 in online mental health support groups. We compare posts from dedicated mental health support communities like r/SuicideWatch with posts from more general groups, namely r/Teenagers. This methodology allows for more robust and contextually informed classification models. The training, test and validation sets are all balanced.

3.3 Experiments

We evaluate our proposed models using publicly available datasets and compare against a range of established baselines and state-of-the-art approaches. Namely, we compare HRG and HSRG against traditional approaches including TF-IDF and Word2Vec, as well as more recent transformer-based models, namely BERT and BERT-HateXplain. These embeddings are paired with a established classifier - K-Nearest Neighbors (KNN). For BERT, we utilize the BERTforSequenceClassification model, a baseline for all the

[1]In no way does this represents the author's perspective. However, the exemplification and explanation of hateful monikers is necessary to this dissertation's subject.

Rhetoric Graphormers proposed (HRG, HSRG and SRG). To benchmark against current state-of-the-art, we also compare our HRG and HSRG results with BERT-HateXplain [36], a model specifically designed for explainable hate speech detection.

In our Rhetoric Graphormer experiments, we utilize BERT embeddings. The pre-processing of the text was minimal, with no removal of symbols since some may indicate hate speech. One such case is the use of triple parenthesis with a word or on its own, ((())), in order to represent the antisemitic view of Jewish people. HRG and HSRG, are trained using the Slur Corpus and the HateXplain Corpus, and evaluated on the 4chan dataset. HSRG incorporates additional training and evaluation using the iSarcasm corpus for sarcasm classification. These models are evaluated on both the Slur Corpus and the 4chan corpus. The ground-truth of the Slur Corpus is trustworthy, as it was annotated by humans and gives us if there was disagreements in the annotation. The 4chan corpus, on the other hand, is more challenging and has an untrustworthy annotation, done by the Rewire API. As this API is no longer publicly available, it could not be further tested. We also showcase examples from the 4chan dataset and common misclassifications in order to qualitatively assess and better evaluate the model's performance.

We train and test BERTforSequenceClassification as a baseline for all tasks - hate speech detection, multi-task learning of hate speech and sarcasm detection, and suicidal posts detection. Another baseline concerns the diversity of each node's connections. We employ an entropy-based metric to compare with the D measure. This metric calculates the negative sum of the product of the attention weights and their logarithms, as expressed by: Diversity Score $= -\sum_{i=1}^{N} w_i \log(w_i)$. As such, nodes with more dispersed attention weights (indicating uncertainty about their relationships) exhibit higher diversity scores. Conversely, nodes with concentrated attention (stronger relationships with fewer nodes) have lower diversity scores. To calculate the diversity, a similarity matrix is constructed by computing the dot product between node embeddings. This matrix represents the pairwise similarity between nodes. Subsequently, attention weights are derived by scaling the similarity matrix and applying a softmax function along the final dimension. This normalization ensures that the attention weights for each node sum to 1.

Experiments were carried out on a Google Colab instance equipped with a T4 GPU with 15.0 GB of GPU RAM, 51.0 GB of system RAM, and 235.7 GB of disk storage. The implementation, trained models and data is available on GitHub

Chapter 4

Results and discussion

4.1 Hate Speech Classification Results

Tables 4.1 and 4.2 present the model's classification performance across a balanced test set and a more challenging validation set. The test set, derived from the Slur Dataset, includes a balanced composition with 40% hate speech examples in a total of 1000 comments. Both sets are classified between hate speech (Class 0) and normal speech (Class 1). For HateXplain, the third class, for offensive speech, is grouped with the normal speech class for the analysis.

Model	Accuracy	Precision			Recall			F1-Score		
		Hateful	Normal	Macro Avg	Hateful	Normal	Macro Avg	Hateful	Normal	Macro Avg
HRG	0.74	0.75	0.70	0.73	0.91	0.42	0.66	0.82	0.53	0.67
HSRG	**0.76**	0.89	0.45	0.67	0.80	0.63	0.72	0.84	0.53	0.68
BERT	**0.84**	0.79	0.91	0.85	0.93	0.74	0.84	0.86	0.82	0.84
BERT with Sarcasm	**0.80**	0.75	0.89	0.82	0.92	0.69	0.80	0.83	0.78	0.80
BERT HateXplain	0.33	0.37	0.81	0.59	0.22	0.25	0.23	0.28	0.38	0.33
TF-IDF SVM	0.46	0.59	0.50	0.54	0.95	0.07	0.51	0.72	0.12	0.42
Word2Vec SVM	0.57	0.59	0.47	0.53	0.83	0.20	0.52	0.69	0.28	0.49

Table 4.1: Model Performance Comparison - Test Set

BERT-based models emerge as the clear performance leaders, maintaining remarkable consistency across all metrics for both classes. This superior performance can be attributed to their powerful pre-trained language understanding capabilities and robust fine-tuning process. The traditional machine learning approaches (TF-IDF Random Forest and Word2Vec) occupy an interesting middle ground, with Random Forest achieving respectable performance (0.83 accuracy) while being more interpretable than deep learning approaches. The TF-IDF KNN and Word2Vec SVM models have lower accuracy (0.75) and show more variability in their recall and F1-scores, especially for hate speech, reflecting less consistent performance in detecting hate speech compared to the top-performing models.

The Hate Rhetoric Graphormer (HRG) model shows promising results with the Diversity Loss, employing binary cross-entropy (BCE) with logit loss and a diversity loss component to encourage nuanced classification performance across classes. On the test

set, it achieves a baseline accuracy of 74%, but the true insight lies in its class-specific performance patterns. The HRG is sensitive to non-hate speech (Class 0), with a precision of 0.75 coupled with an impressive recall of 0.91, yielding a robust F1-score of 0.82. This performance profile suggests the model has developed strong discriminative features for identifying benign content. However, the model's behaviour shifts dramatically when confronting hate speech (Class 1), where we observe a precision-recall trade-off: maintaining reasonable precision (0.70) but struggling with recall (0.42). This asymmetry points to a conservative classification strategy for hate speech, potentially missing subtle forms of harmful content.

The Hate and Sarcasm Rhetoric Graphormer (HSRG) presents an improvement, pushing overall accuracy to 76%. Its most striking characteristic is the exceptional precision (0.89) for non-hate speech detection, suggesting highly reliable positive predictions for benign content. The integration of sarcasm detection capabilities appears to have enhanced the model's ability to capture contextual nuances, reflected in the improved hate speech recall (0.63) compared to the base HRG. This advancement comes at a cost to precision (0.45), indicating a more aggressive classification strategy that may generate more false positives but catches a wider spectrum of potentially harmful content.

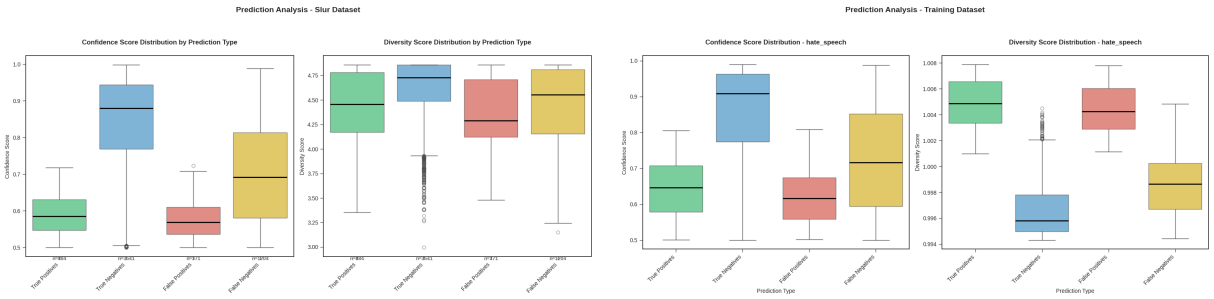

Figure 4.1: Boxplots for HRG (left) and HSRG (right)

In terms of confidence scores, as seen in Figure 4.1, HSRG exhibits high confidence in True Negatives on the Training Dataset (median close to 0.9), indicating reliable non-hate speech predictions, while True Positives show slightly lower confidence. Conversely, HRG on the Slur Dataset maintains high confidence for True Negatives but demonstrates lower confidence overall, suggesting challenges in identifying hate speech. The diversity scores reveal that HSRG has more consistent predictions, particularly for non-hate speech, while HRG shows greater variability in the Slur Dataset, with True Positives displaying the highest diversity.

Overall, HSRG performs better in terms of confidence, especially in recognizing non-hate speech, whereas HRG exhibits lower confidence and higher diversity, indicating difficulties in addressing the complexity of hate speech in the Slur Dataset. This suggests HSRG's stability in predictions, while HRG requires enhancements to better manage the variability present in the Slur Dataset.

The validation results in Table 4.1 are obtained for challenging, imbalanced 4chan dataset containing 809 comments, 22% of which are hate speech.

Model	Accuracy	Precision			Recall			F1-Score		
		Hateful	Normal	Macro Avg	Hateful	Normal	Macro Avg	Hateful	Normal	Macro Avg
HRG	0.27	0.21	0.71	0.46	0.83	0.12	0.47	0.33	0.20	0.27
HSRG	0.24	0.17	0.55	0.36	0.63	0.13	0.38	0.26	0.21	0.24
BERT	0.37	0.25	0.91	0.58	0.92	0.22	0.57	0.39	0.35	0.37
TF-IDF SVM	0.50	022	0.78	0.82	0.83	0.18	0.50	0.35	0.29	0.32
Word2Vec SVM	0.43	0.24	0.82	0.53	0.72	0.35	0.53	0.36	0.49	0.42

Table 4.2: Model Performance Comparison - Validation Set

As seen in Table 4.1, the BERT model with sarcasm detection as an auxiliary task shows better F1-scores, indicating a reduction in false positives. Interestingly, it also shows a slight performance drop compared to the standard BERT approach. The same occurs with HSRG in the validation set. This indicates that the base BERT model is able to implicitly learn to handle such subtleties without the need for an explicit sarcasm detection component. Therefore, the increase in model complexity from the addition of a sarcasm classifier is not warranted, given its effects on performance.

Key insights from these findings include:

- Sarcasm detection minimally impacts performance, making the added model complexity unnecessary.
- BERT's superior accuracy shows its embeddings properly capture semantics.
- The performance degradation in the validation set points to important challenges in base BERT models (HRG, HSRG and BERT) generalization in 4chan data that warrant further investigation.

4.1.1 Qualitative Analysis

Although evaluating the models' performance on the Slur Corpus provides an indicator of their quality, it does not fully capture the complexities of online discourse. Online conversations are highly topical and change rapidly. Additionally, communities like 4chan may have unique language patterns, further complicating a trained model's ability to accurately identify hate speech. To address this, we constructed a small test dataset with entries from 4chan to assess the models' adaptability. It is composed of 42% hate speech instances. The validation set was sourced from 4chan data and subsequently validated by the author to ensure its relevance to the study's focus on hate speech detection.

Sentence	True Label	ChatGPT	Word2Vec	BERT	BERT HateXplain	HRG	HSRG
If Assad is killing that many Arabs he is the good guy in my book	**H**	**H**	N	N	O	**H**	**H**
Killing Muslims is bad	N	N	N	N	N	N	N
Imagine thinking that killing Muslims is bad	H	H	N	N	N	N	**H**
F*ck no, women are scatterbrained nut cases. I always learned exponentially more from my male teachers	**H**	**H**	N	**H**	O	N	**H**
kek tr*nnys should be euthanized, disgusting	**H**	**H**	N	N	**H**	**H**	**H**
Arabs all deserve to die in the most painful way. They are mindless beasts and a plague upon the world and you're saying the same doesn't apply to Jews	**H**	**H**	N	N	**H**	N	**H**
Semites killing Semites who the hell cares	**H**	**H**	N	N	N	N	**H**
transgender	N	N	H	N	N	N	N
You think blacks and hispanics are subhuman jews, out to get everyone? And most white men are soyboys... How do you hide your belief in everyday life?	N	H	N	N	H	N	N
How's thanksgiving looking tomorrow for you? Will you not talk politics to avoid the autistic screeching of your family members or are you part of an actually based family who loves Trump through better or worse including his recent spat with lavar ball?	N	H	N	N	O	N	N
African	N	N	H	N	N	N	N
the world is gonna get very n*ggly soon	**H**	**H**	N	N	**H**	**H**	**H**

Table 4.3: Comparison of hate speech classification across different models with censored examples, where H: Hate Speech, N: Normal text, O: Offensive content (specific to HateXplain model)

It is notable that Word2Vec classifiers often misidentify mentions of marginalized groups, even without slurs, as hate speech. This occurs because Word2Vec captures word associations from its training data, which can still lead to biases if words related to marginalized groups frequently co-occur with negative contexts. As a result, the model incorrectly associates the mere mention of a marginalized group, such as transgender, with hate speech. This issue underscores the challenge of accurately capturing the nuanced and context-specific meanings necessary for reliable hate speech detection, even with a balanced training dataset. Consequently, advanced contextual embeddings and bias mitigation strategies are essential to improve the performance and fairness of these classifiers.

Even though BERT, a contextual embedding method, does not incur this same error, it has other issues with identifying hate speech. Some of these problems are mitigated by BERT-HateXplain, which shows a improved performance. However, it still misses less

explicit hate rhetoric, as seen in Table 4.1.1.

The graph-based models (HRG and HSRG) demonstrate interesting complementary strengths. However, both models occasionally fail to detect hate speech in complex narrative contexts, suggesting that current graph representations may not fully capture long-range semantic dependencies.

ChatGPT's strong (but not perfect) performance across various hate speech examples suggests that large-scale language models effectively learn subtle linguistic patterns indicative of harmful content, despite some false positives. This capability comes at the cost of interpretability — a crucial consideration for deployment in sensitive social contexts that can be remedied with asking for rationales.

The complex nature of online discourse, particularly in communities with distinct linguistic patterns, continues to pose significant challenges for automated hate speech detection. Our results suggest that while current models show promise, robust solutions may require architectural innovations that combine the semantic power of large language models with the interpretability of graph-based approaches.

4.1.2 Diversity awareness

The inclusion of the diversity for the training of both HRG and HSRG showed notable improvements, as seen in Table 4.1.2

	HRG	HSRG
Cross-Entropy Loss	0.42	0.37
Diversity Aware Loss	0.28	0.40
Diversity Aware Loss (D-measure)	0.26	0.24

Table 4.4: Effects of a diversity-aware classifier in the loss in the test dataset

The use of entropy to calculate diversity shown in the second row captures the level of uncertainty in node relationships, minimising the loss when compared to the Graphormer without diversity awareness. This aspect is useful to understand node influence within a graph. The HRG and HSRG results shown in Table 4.1 include diversity awareness based on the D-measure in their training loss, as their performance is superior performance to their counterparts.

The quality of feature representation - for both nodes, graphs and subgraphs - is proved to be a critical factor in model performance. The addition of a simple diversity measure enhanced the model's performance. This suggests that future work should focus not only on classification algorithms but also on exploring graph-theoretic measures for

textual data representation and encoding, as well as similar graph representation learning tasks.

4.2 Suicidal Text Classification

For this task, the results of the Suicidal Rhetoric Graphormer are compared to BERT and reported LIWC results [32].

Model	Accuracy	Precision			Recall			F1-Score		
		Suicidal	Normal	Macro Avg	Suicidal	Normal	Macro Avg	Suicidal	Normal	Macro Avg
SRG	0.77	0.74	0.80	0.77	0.82	0.72	0.77	0.78	0.76	0.77
BERT	0.97	0.97	0.98	0.97	0.98	0.97	0.97	0.97	0.97	0.97
Reported LIWC	0.88			0.88			0.96			0.96

Table 4.5: Performance Comparison for Suicidal Posts Detection

BERT maintains its SOTA accuracy for this classification task as well, demonstrating the effectiveness of transformer-based models in detecting nuanced linguistic markers of suicidal ideation. While the Suicidal Rhetoric Graphormer (SRG) exhibits the lowest performance among the evaluated methods, its novel graph-based approach offers unique advantages in computational complexity and potential interpretability. The significant performance gap between SRG and BERT suggests that the current implementation of the graph-based model requires further refinement. However, the SRG's fundamental architecture presents intriguing optimization possibilities, including that of using LIWC dictionaries to improve its performance and the use of other word embeddings.

4.3 Discussion

The experimental results illuminate several crucial aspects of text classification. While BERT-based models achieved superior numerical performance in both case studies, the graph-based approaches here proposed offer distinct advantages in terms of interpretability and theoretical foundations that warrant deeper consideration. Furthermore, the empirical results show an improvement with the inclusion of diversity awareness in the training process.

The Rhetoric Graphormer architectures represent a step toward more interpretable text classification. Their graph-based nature provides several key advantages: more trans-

parent feature relationships, solid theoretical foundations, and hierarchical information processing. Unlike the opaque attention mechanisms of transformers, graph structures model relationships between linguistic elements more explicitly, allowing for an inspection of how the model forms its decisions. This transparency is crucial for hate speech detection, for example, where understanding the reasoning behind classifications can have significant social implications.

Furthermore, Graph Theory provides a rich mathematical framework for analysing model behaviour. Metrics such as node centrality, clustering coefficients, path analysis, and diversity offer concrete tools for understanding how information flows through the model, making it easier to identify and address failure modes. As the graph structure naturally captures hierarchical relationships in language, from word-level connections to broader semantic structures, graph-based models can provide more intuitive representations of linguistic patterns, including those associated with hate speech.

The HSRG's improved recall in comparison to the HRG (0.63 vs 0.42) demonstrates the value of incorporating sarcasm detection, but the still-modest performance suggests that deeper semantic understanding is needed. Further improvements may stem from introducing different language modelling techniques to the Graphormer or from incorporating information theory principles to optimize edge weights and node representations.

In a rapidly changing scenario with the advent of LLMs, text classification can also leverage these complex architectures with transfer learning. Not only that, but graph-based architectures can offer more interpretability via the distilling of knowledge, using LLM-generated embeddings as initial node features, LLM attention patterns to inform graph structure, and creating hybrid architectures that combine transformer and graph-based components [28]. Even though the black-box nature of LLMs makes them problematic for sensitive applications like hate speech detection, graph-based models can improve its interpretability by providing clear audit trails for decision-making. This would allow for direct intervention in the reasoning process and support formal verification of model properties, improving LLM issues like hallucinations.

These results and future directions suggest that while Transformer-based models currently achieve superior numerical performance, the path forward may lie in developing graph-based architectures that maintain explainability while closing the performance gap. By leveraging the mathematical foundations of graph theory and information theory, we can work toward classification systems that are both powerful, interpretable and explainable.

Chapter 5

Conclusion

Capturing rhetoric nuances is no simple task, as shown by the hate speech and suicidal posts detection. Despite the advances from contextual learning with Transformers architectures, important relationships within a graph can be lost using solely this approach. Our study demonstrates the potential of Graphormer in addressing this complex challenge.

Our findings indicate that the addition of diversity-awareness to Graphormer improves its classifying capabilities for hate speech. While BERT-based models achieved superior accuracy, our models offer similar results with distinct advantages in terms of interpretability and theoretical foundations that warrant deeper consideration.

The Rhetoric Graphormers benefit from the contextual learning from Transformer architectures and a more explicit modelling of the relationships between linguistic elements. The improvement in performance from a simple diversity measure can be attributed to its capacity to encode feature relationships that better preserve structural information. This improved node representation helps with the understanding of the reasoning behind complex classification tasks. By integrating these representations with LLMs, we can achieve more interpretable systems that maintain the LLM's high performance while providing clearer insights into their classification decisions.

Furthermore, the improvements in the diversity-aware Graphormers demonstrate that graph-theoretic measures such as diversity can improve model's performance in graphs. Diversity-aware Graphormers should be tested in other areas, especially those where Graphormer already achiever SOTA performances - including recommendation systems [65], molecule modelling [71], image reconstruction [31], and extractive multimodal summarization [23]. The rich mathematical framework for analysing model behaviour offered by Graph Theory should be leveraged to improve current models. The dynamics of complex networks can be further explored with properties like diffusion and with the dynamical graph construction mentioned in Section 2.1.2.

In conclusion, our Graphormer-based approach shows promise in tackling the challenging task of hate speech detection. The proposed models should be optimized, and the impact of different centrality measures on their performance should be further investigated. Future research directions should also focus on exploring the use of Graphormer-

based architectures as complementary explainability tools for LLMs and in other graph-based tasks, such as recommendation systems and customer service chatbots.

5.1 Publications

This work was published and presented at the 2025 IEE International Conference on Communications (ICC) under the "Social Networks" call for papers [4].

Bibliography

[1] Ibrahim Abu Farha, Silviu Vlad Oprea, Steven Wilson, and Walid Magdy. SemEval-2022 task 6: iSarcasmEval, intended sarcasm detection in English and Arabic. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 802–814, Seattle, United States, July 2022. Association for Computational Linguistics.

[2] Sinan Aral, Lev Muchnik, and Arun Sundararajan. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks. *Proc. Natl. Acad. Sci. U. S. A.*, 106(51):21544–21549, December 2009.

[3] Albert-Laszlo Barabasi. *Network Science*. Cambridge University Press, Cambridge, England, July 2016.

[4] Allana Tavares Bastos and Martín Gómez Ravetti. Diversity-aware graphormer: Insights from hate speech detection on 4chan. In *ICC 2025 - IEEE International Conference on Communications*, pages 7146–7151, 2025.

[5] Giorgos Bouritsas, Andreas Loukas, Nikolaos Karalias, and Michael Bronstein. Partition and Code: Learning how to compress graphs. In *Advances in Neural Information Processing Systems*, volume 34, pages 18603–18619. Curran Associates, Inc., 2021.

[6] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, September 2018.

[7] Laura C. Carpi, Tiago A. Schieber, Panos M. Pardalos, Gemma Marfany, Cristina Masoller, Albert Díaz-Guilera, and Martín G. Ravetti. Assessing diversity in multiplex networks. *Scientific Reports*, 9(1):4511, March 2019.

[8] Duanbing Chen, Linyuan Lü, Ming-Sheng Shang, Yi-Cheng Zhang, and Tao Zhou. Identifying influential nodes in complex networks. *Physica A: Statistical Mechanics and its Applications*, 391(4):1777–1787, Feb 2012.

[9] Wangqun Chen, Fuqiang Lin, Guowei Li, and Bo Liu. A survey of automatic sarcasm detection: Fundamental theories, formulation, datasets, detection methods, and opportunities. *Neurocomputing*, 578:127428, April 2024.

[10] Yongqiang Chen, Yatao Bian, Bo Han, and James Cheng. How Interpretable Are Interpretable Graph Neural Networks? In *Proceedings of the 41st International Conference on Machine Learning*, pages 6413–6456. PMLR, July 2024.

[11] Rudi Cilibrasi. *Statistical Inference Through Data Compression*. doctoral, University of Amsterdam, October 2006.

[12] Thomas Colley and Martin Moore. The challenges of studying 4chan and the Alt-Right: 'Come on in the water's fine'. *New Media & Society*, 24(1):5–30, January 2022.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019.

[14] Luis Espinosa Anke, Thierry Declerck, Dagmar Gromann, Ziqi Zhang, Lei Luo, Dagmar Gromann, Luis Espinosa Anke, and Thierry Declerck. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semant. Web*, 10(5):925–945, jan 2019.

[15] Paula Fortuna and Sérgio Nunes. A Survey on Automatic Detection of Hate Speech in Text. *ACM Comput. Surv.*, 51(4):85:1–85:30, July 2018.

[16] Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3):215–239, Jan 1978.

[17] Cristopher G S Freitas, Andre L L Aquino, Heitor S Ramos, Alejandro C Frery, and Osvaldo A Rosso. A detailed characterization of complex networks using information theory. *Sci. Rep.*, 9(1):16689, November 2019.

[18] Lei Gao, Alexis Kuppersmith, and Ruihong Huang. Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In Greg Kondrak and Taro Watanabe, editors, *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 774–782, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing.

[19] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc.

[20] Liam Hebert, Lukasz Golab, and Robin Cohen. Predicting hateful discussions on reddit using graph transformer networks and communal context. In *2022*

IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), pages 9–17, 2022.

[21] Petter Holme and Jari Saramäki. Temporal networks. *Phys. Rep.*, 519(3):97–125, October 2012.

[22] Amir Reza Jafari, Guanlin Li, Praboda Rajapaksha, Reza Farahbakhsh, and Noel Crespi. Fine-Grained Emotions Influence on Implicit Hate Speech Detection. *IEEE Access*, 11:105330–105343, 2023.

[23] Xiankai Jiang and Jingqiang Chen. Heterogeneous graphormer for extractive multimodal summarization. *Journal of Intelligent Information Systems*, September 2024.

[24] Zhiying Jiang, Matthew Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, and Jimmy Lin. "low-resource" text classification: A parameter-free classification method with compressors. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, page 6810–6828. Association for Computational Linguistics, July 2023.

[25] Kibae Kim and Jörn Altmann. Effect of homophily on network formation. *Commun. Nonlinear Sci. Numer. Simul.*, 44:482–494, March 2017.

[26] Jana Kurrek, Haji Mohammad Saleem, and Derek Ruths. Towards a comprehensive taxonomy and large-scale annotated corpus for online slur usage. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 138–149, Online, Nov 2020. Association for Computational Linguistics.

[27] A Li, S P Cornelius, Y-Y Liu, L Wang, and A-L Barabási. The fundamental advantages of temporal networks. *Science*, 358(6366):1042–1046, November 2017.

[28] Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. A Survey of Graph Meets Large Language Model: Progress and Future Directions. In *Proceedings of the Thirty-ThirdInternational Joint Conference on Artificial Intelligence*, pages 8123–8131, Jeju, South Korea, August 2024. International Joint Conferences on Artificial Intelligence Organization.

[29] Yusheng Li, Yilun Shang, and Yiting Yang. Clustering coefficients of large networks. *Information Sciences*, 382–383:350–358, Mar 2017.

[30] Silas Lima Filho, Mônica Ferreira Da Silva, and Jonice Oliveira. Textual Analysis of Facebook Communities Related to Depression. In Sílvia Araújo, Micaela Aguiar, and Liana Ermakova, editors, *Digital Humanities Looking at the World*, pages 231–239. Springer Nature Switzerland, Cham, 2024.

[31] Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphormer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12939–12948, October 2021.

[32] Jingfang Liu and Mengshi Shi. A Hybrid Feature Selection and Ensemble Approach to Identify Depressed Users in Online Social Media. *Frontiers in Psychology*, 12, January 2022.

[33] Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Controllability of complex networks. *Nature*, 473(7346):167–173, May 2011.

[34] Daniel M Low, Laurie Rumker, Tanya Talkar, John Torous, Guillermo Cecchi, and Satrajit S Ghosh. Natural Language Processing Reveals Vulnerable Mental Health Support Groups and Heightened Health Anxiety on Reddit During COVID-19: Observational Study. *Journal of Medical Internet Research*, 22(10):e22635, October 2020.

[35] Rijul Magu, Kshitij Joshi, and Jiebo Luo. Detecting the hate code on social media. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1):608–611, May 2017.

[36] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. Hatexplain: A benchmark dataset for explainable hate speech detection. *arXiv preprint arXiv:2012.10289*, 2020.

[37] Yashar Mehdad and Joel Tetreault. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303, Los Angeles, September 2016. Association for Computational Linguistics.

[38] Attila Mester, Andrei Pop, Bogdan-Eduard-Mădălin Mursa, Horea Greblă, Laura Dioşan, and Camelia Chira. Network analysis based on important node selection and community detection. *Mathematics*, 9(18):2294, September 2021.

[39] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, September 2013.

[40] Michael T. Mills and Nikolaos G. Bourbakis. Graph-Based Methods for Natural Language Processing and Understanding—A Survey and Analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(1):59–71, January 2014.

[41] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for Graphs: An Overview from Architecture Perspective, February 2022.

[42] Vivi Nastase, Rada Mihalcea, and Dragomir R. Radev. A survey of graphs in natural language processing. *Natural Language Engineering*, 21(5):665–698, November 2015.

[43] Stuart Oldham, Ben Fulcher, Linden Parkes, Aurina Arnatkevic Iūtė, Chao Suo, and Alex Fornito. Consistency and differences between centrality measures across distinct classes of networks. *PLoS One*, 14(7):e0220061, July 2019.

[44] Juri Opitz. Gzip versus bag-of-words for text classification, August 2023. arXiv:2307.15002 [cs].

[45] World Health Organization. Suicide. https://www.who.int/news-room/fact-sheets/detail/suicide.

[46] Antonis Papasavva, Savvas Zannettou, Emiliano De Cristofaro, Gianluca Stringhini, and Jeremy Blackburn. Raiders of the Lost Kek: 3.5 Years of Augmented 4chan Posts from the Politically Incorrect Board. *Proceedings of the International AAAI Conference on Web and Social Media*, 14:885–894, May 2020.

[47] Vahid Parvaresh. Covertly communicated hate speech: A corpus-assisted pragmatic study. *Journal of Pragmatics*, 205:63–77, 2023.

[48] John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. Toxicity Detection: Does Context Really Matter? In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4296–4305, Online, July 2020. Association for Computational Linguistics.

[49] Phu Pham, Loan T. T. Nguyen, Witold Pedrycz, and Bay Vo. Deep learning, graph-based text representation and classification: A survey, perspectives and challenges. *Artificial Intelligence Review*, 56(6):4893–4927, June 2023.

[50] Mihaela Popa-Wyatt. Not All Slurs are Equal. *Phenomenology and Mind*, No 11:150–156 Pages, January 2017.

[51] Rolandos Alexandros Potamias, Georgios Siolas, and Andreas Georgios Stafylopatis. A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, 32(23):17309–17320, December 2020.

[52] Yiting Qu, Xinlei He, Shannon Pierson, Michael Backes, Yang Zhang, and Savvas Zannettou. On the evolution of (hateful) memes by means of multimodal contrastive learning. In *On the Evolution of (Hateful) Memes by Means of Multimodal Contrastive Learning*, page 293–310. IEEE Computer Society, Dec 2022.

[53] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, Feb 2003.

[54] Manoel Ribeiro, Pedro Calais, Yuri Santos, Virgílio Almeida, and Wagner Meira Jr. Characterizing and Detecting Hateful Users on Twitter. *Proceedings of the International AAAI Conference on Web and Social Media*, 12(1), June 2018.

[55] Thorsten Rings, Timo Bröhl, and Klaus Lehnertz. Network structure from a characterization of interactions in complex systems. *Sci. Rep.*, 12(1):11742, July 2022.

[56] Jo Robinson, Georgina Cox, Eleanor Bailey, Sarah Hetrick, Maria Rodrigues, Steve Fisher, and Helen Herrman. Social media and suicide prevention: A systematic review. *Early Intervention in Psychiatry*, 10(2):103–121, 2016.

[57] Tiago A. Schieber, Laura Carpi, Albert Díaz-Guilera, Panos M. Pardalos, Cristina Masoller, and Martín G. Ravetti. Quantification of network structural dissimilarities. *Nature Communications*, 8(1):1–10, January 2017.

[58] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks, October 2017.

[59] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain, April 2017. Association for Computational Linguistics.

[60] Sara Nadiv Soffer and Alexei Vázquez. Network clustering coefficient without degree-correlation biases. *Physical Review E*, 71(5):057101, May 2005.

[61] Abhinav Upadhyay. Abhinav-upadhyay/lz77_is_all_you_need, August 2023.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[63] N. Velásquez, R. Leahy, N. Johnson Restrepo, Y. Lupu, R. Sear, N. Gabriel, O. K. Jha, B. Goldberg, and N. F. Johnson. Online hate network spreads malicious COVID-19 content outside the control of individual social media platforms. *Scientific Reports*, 11(1):11549, June 2021.

[64] Byron C. Wallace. Computational irony: A survey and new perspectives. *Artificial Intelligence Review*, 43(4):467–483, April 2015.

[65] Jing Wang and Jiangtao Ren. Graphormer based contrastive learning for recommendation. *Applied Soft Computing*, 159:111626, 2024.

[66] Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In Zeerak Waseem, Wendy Hui Kyong Chung, Dirk Hovy, and Joel Tetreault, editors, *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada, August 2017. Association for Computational Linguistics.

[67] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. *Graph Neural Networks for Natural Language Processing: A Survey*. Now Foundations and Trends, 2023.

[68] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Singapore, Singapore, 2022.

[69] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph Information Bottleneck. In *Advances in Neural Information Processing Systems*, volume 33, pages 20437–20448. Curran Associates, Inc., 2020.

[70] Chuanpeng Yang, Fuqing Zhu, Guihua Liu, Jizhong Han, and Songlin Hu. Multimodal hate speech detection via cross-domain knowledge transfer. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 4505–4514, New York, NY, USA, 2022. Association for Computing Machinery.

[71] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do Transformers Really Perform Badly for Graph Representation? In *Advances in Neural Information Processing Systems*, volume 34, pages 28877–28888. Curran Associates, Inc., 2021.

[72] Adrian Zbiciak and Tymon Markiewicz. A new extraordinary means of appeal in the polish criminal procedure: the basic principles of a fair trial and a complaint against a cassatory judgment. *Access to Justice in Eastern Europe*, 6(2):1–18, March 2023.

[73] Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the Expressive Power of GNNs via Graph Biconnectivity. In *The Eleventh International Conference on Learning Representations*, September 2022.

[74] Zhilu Zhang and Mert Sabuncu. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Appendix A

GZip-based hate speech detection

The compression method had a resurgence of popularity in 2023 with the paper [24]. This approach considers compression as learning, given that compression typically uses patterns to reduce representations. Not only that, but compression is a robust and simple method that improves when aggregating similar files [11]. Small variations in the pairwise compressed sizes yield great results, better than when separately compressing.

This compression usually compares pairs of compressed information with the Normalized Compression Distance (NCD). Clustering according to this compression-based similarity measure groups sequences with similar features. However, these features are not explicitly known, and an analysis of what the compressor does might not tell us which features are being expressed by the conglomerates of features analysed by the compressor [11]. Furthermore, the most different pairs become equidistant, with a maximum distance. This implicates in the loss of detail for classification.

In [24], the data is compressed using Gzip and distances between two compressed texts are measured using NCD. Then, the method leverages the better compression of shared common patterns in a class to employ a simple classifier, KNN. The results obtained do not beat SOTA, but are close to it - although the tie-breaker used in the original paper is very optimistic [44]. GZIP is an old, fast and reliable Lempel-Ziv type compressor (LZ77) with a 32-kilobyte window [11]. LZ77 is a lossless dictionary-based compression algorithm that uses a sliding window-based search buffer, which allows for a high level of compression for commonly reoccurring substrings. GZIP's deflate algorithm further compresses its input with another step with Huffman coding, that, unlike LZ77, is a statistical coding technique. The better performance of compression methods that use LZ77 compression (GZIP, LZMA and Zstandard) in comparison with Bzip, a method that includes a final compression with Huffman coding, points to the importance of LZ77's importance. Not all lossless compression algorithms perform as well as GZIP for text classification. LZ4, an optimized implementation of LZ77's principles, performs well, but not as well as GZIP. Huffman coding, on its own, performs the worst. Experimental results [61] show that this great performance with GZIP occurs due to its step of LZ77 compression.

To use GZip for hate speech detection, the text is first vectorized and then com-

pressed. The resulting compression is then used in a simple classifier, K Nearest Neighbours (KNN). The results with this method were encouraging, with 73% accuracy, with 77% accuracy for hate speech and 67% for normal speech. However, for the more challenging validation dataset, the model fails to identify normal speech, with 4 false positives in this class.

An issue that this method faces is the use of NCD, as it loses important details. Another issue is that the compression provided by LZ77 is flat, i. e. it does not consider the hierarchy since it determines disjoint clusters in dimensional representation. This structure can be very informative in communication as the context can dramatically change the meaning of the exact same text. However, the projection of the distance matrix information may get distorted with an increasing number of items.

Even so, this compression method for word representation constitutes a simple but effective method, that requires no pre-processing and no pre-training.