

Hierarchical Clustering integrated with Transformer model for XMC

*Bachelor Thesis Project-2 to be submitted in the partial
fulfillment of the
requirements for the degree*

Bachelor of Technology

IN

Mechanical Engineering

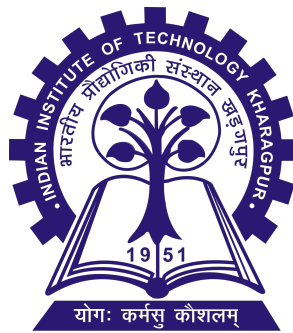
by

Bade Naga Manikanta Praveen

(21ME3AI32)

Under the guidance of

Prof. Mahesh Mohan M R



Department of Artificial Intelligence
Indian Institute of Technology Kharagpur
29th November, 2024



Department of Artificial Intelligence,
Indian Institute of Technology
Kharagpur, West Bengal, India - 721302

CERTIFICATE

This is to certify that we have examined the thesis entitled **Hierarchical Clustering integrated with Transformer model for XMC**, submitted by **Bade naga manikanta praveen (21ME3AI32)** an undergraduate student of **Department of Mechanical Engineering** in partial fulfillment for the award of degree of Bachelor of Technology. We hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfillment for the undergraduate Degree for which it has been submitted. The thesis has fulfilled all the requirements as per the regulations of the Institute and has reached the standard needed for submission.

Guide's Signature

Date: _____

ACKNOWLEDGEMENTS

I am profoundly grateful to my advisor, Prof. Mahesh Mohan M R, for introducing me to the captivating field of Extreme Multilabel classification. His extensive knowledge and thoughtful guidance have been essential to the advancement and success of my research.

I also extend my sincere thanks to the faculty and staff, whose assistance has been invaluable on my academic journey. Their openness to sharing insights and offering advice has greatly enriched my educational experience.

A special note of gratitude goes to my project teammates Mahim Jain, Gaurav Pathak, Prakhar Verma for their support and collaboration. Their feedback and perspectives have been a source of encouragement and personal growth.

Lastly, I owe my deepest appreciation to my family. Their steadfast support, patience, and belief in my potential have been my constant strength. This achievement is as much a tribute to them as it is to my own efforts.

ABSTRACT

Extreme multilabel classification (XMC) addresses scenarios in which each instance may be associated with a very large set of possible labels—often in the hundreds of thousands or millions—posing severe challenges in scalability, efficiency, and prediction latency. XMC is pivotal for managing e-commerce product catalogs with thousands to millions of labels, but the tail label problem—where rare categories suffer from sparse training data—poses significant challenges.

In this work, we tackle the tail label problem by proposing a hierarchical clustering framework integrated with a T5 transformer model. We structure the label space into a tree using hierarchical K-Means clustering and guide the T5 model to output required label. Our work is evaluated on a real-world e-commerce dataset, aiming to predict product modules from descriptions, retailer names, and prices. The results show that hierarchical clustering notably enhances tail label prediction over a standard T5 model, highlighting a promising path for building XMC systems.

This research advances the field of XMC by addressing the critical issue of tail label prediction, contributing to the development of scalable and high-accuracy systems capable of managing millions of labels. The findings demonstrate practical benefits for e-commerce applications, including enhanced user personalization, improved search relevance, and more effective product discovery. By leveraging a real-world dataset, this thesis tackles one of the key challenges facing modern XMC tasks, offering insights into solutions for large-scale, real-world deployments.

Contents

1	Extreme Multilabel Classification(XMC)	1
1.1	Introduction	1
1.2	Motivation	2
1.3	Challenges of Extreme Multilabel Classification	3
2	Literature review	6
2.1	Compressed Sensing Methods	6
2.2	Linear Algebra Tools	9
2.3	Tree based Methods	13
2.4	Deep learning methods	20
2.5	Transformers	21
2.6	T5 Transformer	25
3	Solving a E-commerce Problem	30
3.1	Problem Statement and its significance	30
3.2	Analysis of Dataset	33
3.3	Data Preprocessing	37
4	Methodology and Experimentation	38
4.1	Analysis and Observations	38
4.2	XMC with only T5 Transformer	39
4.3	Introduction to Hierarchial K means clustering	41
4.4	XMC with Hierarchial K means clustering + T5	45
5	Results and Discussion	48
5.1	Impact of Hierarchical Clustering	48
5.2	Evaluation Metrics	48

5.3	Discussion	49
5.4	Generalisation capability towards Tail labels	49
5.5	Scalability	49
5.6	Practical Impact	50
5.7	Limitations and Challenges	50
5.8	Broader Applicability	51
5.9	Conclusion	51
Bibliography		51

Chapter 1

Extreme Multilabel Classification(XMC)

1.1 Introduction

Extreme Multilabel Classification (XMC) is the task of learning a predictive model that can assign a relevant subset of labels to an instance from an extraordinarily large label set. The formal objective is to learn a function $f : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$ that, given an input space $\mathcal{X} \subseteq \mathbb{R}^d$ and a label space $\mathcal{Y} = \{1, 2, \dots, L\}$, where L can range from hundreds of thousands to millions, predicts the set of most pertinent labels for any given instance. When there are extremely sparse label distributions, the high dimensionality of the output space poses several unique challenges, including label imbalance, computational scalability, and prediction accuracy.

Unlike traditional multilabel classification tasks, XMC calls for algorithms that can work efficiently with limited memory and time. With the vast majority of labels appearing infrequently and a small fraction occurring very frequently, label distributions typically follow a power-law pattern. This heavy-tailed distribution complicates learning meaningful representations for infrequent labels. Additionally, the label correlations in XMC are very complicated and nontrivial, which means that sophisticated methods are required to capture cross-label dependencies.

Several families of methods have been developed to address XMC issues. Embedding-based approaches attempt to project the label space into a lower-dimensional manifold, whereas tree-based approaches recursively divide the label space to accommodate logarithmic time predictions. Despite their simplicity, one-vs-rest techniques often rely on complex regularization and parallelization techniques to remain effec-

tive at extreme scales. Recently, deep learning models and attention mechanisms have been studied to exploit the representational power of neural architectures in capturing high-order interactions among labels.

In general, XMC offers a special mix of opportunities and challenges by pushing the boundaries of learning algorithms, scalability, and theoretical comprehension in extreme multilabel learning scenarios.

1.2 Motivation

XMC has become an important field of research because of its direct relevance to large-scale real-world applications where there may be hundreds of thousands or even millions of possible labels. Modern e-commerce sites like Amazon use XMC techniques to recommend products to users from a large catalog based on their browsing history, search terms, and past purchases. Predicting a small subset of relevant products from a large inventory is crucial to increase user satisfaction and revenue.

In a similar vein, web search engines such as Google and Bing rely on XMC to tag and retrieve relevant documents from large corpora. Since queries are often mapped to a large label space with millions of potential intents or document categories, accurate and computationally efficient classification models are needed. Additionally, in order to properly customize search results in individualized search experiences, user queries must be precisely matched with pertinent labels.

On content-rich websites like Wikipedia, XMC facilitates the automatic annotation and categorization of articles. Because Wikipedia has so many articles and categories, XMC models are crucial to maintaining high-quality automated categorization. A Wikipedia article may be associated with more than one category. In large knowledge bases, this makes it easier to organize, navigate, and retrieve information.

The increasing complexity and volume of digital content, along with the need for personalized and real-time experiences, underscore the need to develop C algorithms. For these applications, models must be able to handle extreme label spaces, operate under stringent time and memory constraints, and maintain high precision even on infrequent or tail labels. Because of this, XMC research is not only theoretically intriguing, but also crucial to the creation of scalable and intelligent systems in a wide range of industries.

1.3 Challenges of Extreme Multilabel Classification

XMC is distinct from regular multilabel learning as a result of a series of different challenges, in spite of its great promise of application. Generally, two fundamental problems arise: the natural issues created by the long-tail distribution of the labels and the training challenges associated with extreme sets of labels.

Training Challenges: The size of the label space being gigantic poses particular challenges in training classifiers in the XMC setting. When L is extremely large, conventional multilabel methods such as one-vs-rest classification are impractical since they would have to train and store L models. These methods are computationally intensive and require much memory and storage space, even when practical. In addition, full-batch training is very costly as a result of frequent failure by conventional optimization methods to scale gracefully.



Figure 1.1: Large label space causing Training Challenges (Indo ML poster)

The extreme sparsity of label assignments is another key concern. An example in XMC is typically associated with a few labels out of a huge label set. This sparsity degrades the learning signal and complicates the optimization of objective functions, particularly when there are ambiguous or noisy examples involved. In addition, the training process is skewed by the imbalanced supply of positive and negative labels per example, which often makes models predict common labels excessively and miss out on rare but important ones.

To reduce computational burdens, sampling approaches such as hard negative mining and negative sampling have been devised. The ability of the model to generalize is directly affected by the non-trivial process of selecting informative negatives from millions of possible outcomes. Additionally, attentive regularization methods and scalable optimization techniques that can handle high-dimensional, sparse gradients and model parameters are often needed for successful training.

Tail Label Challenges: Handling tail labels, or labels that are very rare across the training data, is another XMC signature challenge. Empirical evidence shows that label frequencies in XMC tasks tend to follow a power-law distribution, i.e., a large number of labels appear only in a small number of instances. Good overall system performance, especially in applications requiring high recall or individualized recommendations, relies on learning compact classifiers for such tail labels.

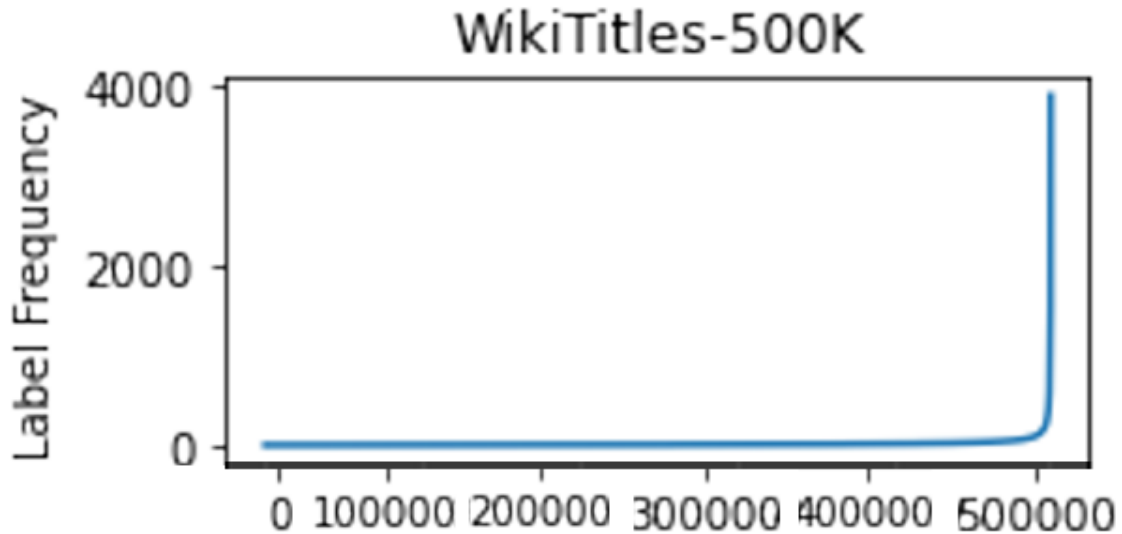


Figure 1.2: Tail label problem in wikipedia Dataset (Dasgupta et al. 2023)

Lack of good examples in tail labels prevents models from generalizing and detecting valuable patterns. Popular learning algorithms don't work well on infrequent labels since they are head-label-biased and optimized for frequent labels. Additionally, subtleties that can be correctly predicted for tail labels might get lost inadvertently by embeddings or dimensionality reduction algorithms based on global label similarity.

Label-aware loss functions, few-shot learning extensions, and meta-learning methods are among the specialized strategies that have been proposed to handle this issue. Some strategies take advantage of hierarchical label frameworks or external semantic details to reinforce learning, while other methods attempt to balance learning in a way of assigning higher priorities to tail labels. Exploration towards achieving reliable performance on the long tail is ongoing and unsolved despite these efforts.

Moreover, for the evaluation metrics in XMC to properly capture the importance of tail labels, these metrics should be chosen wisely. Unless specifically altered, popular metrics like precision@k or normalized discounted cumulative gain (nDCG) can disproportionately reward head labels. This necessitates the development and use of evaluation schemes that encourage accurate predictions over the entire range of label frequencies.

Chapter 2

Literature review

2.1 Compressed Sensing Methods

XMC problems are a genuine challenges because the number of labels is extremely large, normally in the tens of hundreds of thousands or even larger. Traditional one-versus-all paradigms for classification become computationally burdensome, and there is therefore a motivation to explore dimensionality reduction approaches. Compressed sensing is one such possible direction, a method originally formulated in the signal processing community. The essence of compressed sensing in XMC is projecting the high-dimensional label space into a lower-dimensional embedding, losing no important information necessary for reconstructing the original label vectors.

The fundamental notion in compressed sensing methods is that label vectors, although they may form a large output space, are sparse by nature. In other words, an example activates just a limited number of active labels. With sparsity comes efficient encoding and decoding. Normally, these methods pass through three general stages: compression, learning, and reconstruction.

During the compressing phase, the original label matrix is mapped into a significantly lower-dimensional space using linear projections or random transformations. Random matrices were utilized in the initial approaches to make sure with high probability that distances between each pair of sparse vectors approximately remain the same. The learning stage is training prediction models in the reduced space, which is computationally more efficient as it is smaller in size. Finally, at the time of reconstruction, the prediction in the embedded domain is mapped back to the original label space, usually through optimization techniques to retrieve the sparse signal.

The first work along this line employed random projections across the label matrix and utilized binary relevance methods in the compressed space. The original labels were later reconstructed by solving sparse optimization problems. Solving these optimization problems during the inference stage, however, proved to be a bottleneck, and thus research was steered towards more efficient alternatives.

Subsequent enhancements introduced more structured approaches to the compression process. As an example, rather than using random projections alone, singular value decomposition (SVD) was utilized to achieve orthogonal projection matrices that could capture the label correlations better. In particular, Principal Label Space Transformation (PLST) (Tai & Lin 2012) recommended applying SVD to project labels onto a lower-dimensional compressed subspace based on principal components to minimize information loss during compression. PLST significantly simplified the decoding phase as well, where decoding was merely a linear transformation with thresholding.

A notable upgrade to PLST was Conditional Principal Label Space Transformation. Since label correlations may not be enough for the best compression, CPLST (Chen & Lin 2012) added feature-label correlations to the embedding process. Employing canonical correlation analysis (CCA) along with SVD, CPLST aimed to identify a projection matrix that simultaneously optimizes encoding error and prediction error. Although good in theory, CPLST introduced additional computational cost, thus becoming less scalable to extremely large label spaces.

Parallel to these matrix decomposition-based approaches, other approaches inspired by data structures like Bloom filters (Bloom 1970) were explored. These methods proposed representing sets of labels as sparse binary vectors with several hash functions. A unique bit pattern was designated for each label, and the aggregate label set was represented by a bitwise OR of these patterns. Such representations significantly reduced the number of classification operations required. Naive applications of Bloom filters had the limitation of high false positives. To address this, robust variants incorporated methods such as label clustering and error correction processes, taking advantage of the fact that in real-world datasets, most labels do not often co-occur.

The robust Bloom filter (RBF) model greatly improved over typical Bloom filter models by partitioning the label set into mutually exclusive clusters. This clustering was done using graph-based clustering algorithms on the label co-occurrence graph,

following pruning of high-frequency "head" labels. The clusters were then encoded independently, so that failure of errors in one cluster did not affect the entire label set. RBF's decoding was a two-stage process: picking the correct cluster and probabilistic label selection within the cluster.

Another new compression-based approach was the Log-Time Log-Space Extreme Classification (LTLS) (Jasinska & Karampatziakis 2016) method. LTLS mapped the XMC task to a structured prediction task by encoding labels as trellis graph paths. Each label was encoded as a unique path, and the prediction task was reduced to finding the highest scoring paths through the graph. While space- and time-efficient, LTLS was bounded by classification accuracy, especially for large unbalanced datasets with a long tail of infrequent labels.

Hash-based approaches kept improving with the invention of Merged Averaged Classifiers via Hashing (MACH) (Huang et al. 2018). In MACH, the labels were independently merged into a small number of meta-classes utilizing many hash functions. Each meta-class classification task was much smaller in size and could be learned separately as well as in parallel. In prediction time, predictions from several hash functions were added to get final label predictions. This method made use of concepts from count-sketch data structures with the advantage of possessing an attractive tradeoff between accuracy, memory usage, and computation cost.

In a summary of compressed sensing techniques for XMC, it is clear that the thread running through all techniques is the use of sparsity and label correlation structure. Techniques differ in how they compress: from random projections alone to structured decompositions such as SVD and CCA, to hash-based representations of labels.

A comparative analysis shows that initial compressed sensing techniques, though theoretically interesting, were behind empirically in performance because of expensive decoding phases. Matrix decomposition methods such as PLST and CPLST were more feasible in their application through simpler decoding at the cost of increased preprocessing expense. Bloom filter methods, particularly strong methods, were more space-efficient but suffered from high predictive accuracy under tight assumptions of co-occurrence of labels. Graph-based encoding methods such as LTLS introduced new perspectives but suffered when used in large-scale real-world datasets.

More recent methods, such as MACH, have a promising methodology by combining the strengths of hashing and averaging to generate models that are scalable

yet competitive in predictive performance. Such methods make significant use of parallelization and are perfectly suited for distributed computing setups.

Although these gains have been achieved, challenges remain. Reconstruction error during decoding, especially for rare labels (tail labels), remains a key bottleneck. Furthermore, compressed sensing algorithms fall behind in fast-evolving label spaces with high rates of new label occurrence. Some of the potential directions for future work include dynamic embedding spaces, adaptive hashing techniques, and hybrid models that combine compressed sensing and deep learning models in order to better capture complex label relationships.

In short, compressed sensing-inspired methods have played a central role in driving the boundaries of XMC. By effectively reducing the dimensionality of the label space, they have provided tractable solutions to ostensibly intractable problems. Yet, as XMC task size and complexity continue to grow, advances in embedding design, reconstruction methods, and theoretical assurances will be required to realize the full power of compressed sensing in this domain. Linear algebra methods outperform Compressed Sensing Methods.

2.2 Linear Algebra Tools

Linear algebra is the cornerstone of many techniques in extreme multi-label classification (XMC), providing scalable solutions to deal with issues raised by high-dimensional label spaces. In contrast to compressed sensing or tree-based methods, linear algebra techniques tend to emphasize matrix factorization, low-rank approximations, and structured optimization to minimize computational complexity at the cost of maintaining predictive accuracy. These methods take advantage of the natural sparsity and label correlations in label matrices to allow for efficient inference and training even when handling millions of labels. This part delves into significant linear algebraic techniques, their mathematical expressions, and their impact on XMC research advancement.

One of the underlying methods is the *Logistic Embedding for Multi-Label Learning* (LEML) framework, which casts XMC as an empirical risk minimization problem under low-rank constraints. LEML considers the weight matrix $Z \in \mathbb{R}^{d \times L}$ controlling label predictions to be factored into lower-dimensional factors $U \in \mathbb{R}^{d \times k}$ and $V \in \mathbb{R}^{k \times L}$, where $k \ll L$. By applying rank constraint $\text{rank}(Z) \leq k$, LEML scales down the

parameters from dL to $k(d + L)$, rendering the problem tractable. The optimization target minimizes the logistic loss across seen labels along with regularization of U and V with Frobenius norms. This approach generalizes previous compressed sensing methods such as CPLST but eschews label space transformations in the explicit sense, learning latent embeddings directly via gradient-based optimization. A closed-form solution by singular value decomposition (SVD) arises when employing squared loss, showing equivalence to conditional principal label space transformation approaches under certain conditions.

The global low-rank assumption limitations were revealed with the introduction of *Sparse Local Embeddings for Extreme Classification* (SLEEC) (Bhatia et al. 2015). Seeing that tail labels destroy low-rank structures, SLEEC takes a distance-preserving approach. It builds embeddings $z_i \in \mathbb{R}^{\hat{L}}$ for every instance x_i such that distances between close label vectors y_i and y_j are maintained in the embedding space. This is done by minimizing $\|P_\Omega(Y^T Y) - P_\Omega(Z^T Z)\|_F^2$, where P_Ω projects onto the set of nearest neighbors. SLEEC groups training data into clusters and learns individual embeddings for each cluster to perform local low-rank approximations. In prediction, k-nearest neighbors in the embedding space select appropriate labels. The use of local as opposed to global structure for the method makes it efficient to deal with tail labels, although computational expense is incurred due to solving large-scale optimization through alternating direction method of multipliers (ADMM).

Based on low-rank principles, *REMBRANDT* (Randomized Efficient Matrix Bandit Optimization) (Mineiro & Karampatziakis 2015) utilizes randomized matrix factorization for dimensionality reduction of label space. Approximating the label matrix Y by randomized SVD, REMBRANDT builds embeddings that preserve principal correlations among labels and features. The approach optimizes $\min_W \|Y - XW\|_F^2$ with $\text{rank}(W) \leq k$, where $W = UV^T$. Randomized algorithms speed up computation by approximating Y in a lower-dimensional subspace prior to factorization, saving time complexity from $O(nL^2)$ to $O(nLk)$. Theoretical bounds relate the approximation error to the spectral decay of Y , providing robustness for datasets with very quickly decaying singular values.

Tackling the twin problems of sparsity and scalability, *PD-Sparse* (Primal-Dual Sparse Learning) proposes a margin-maximizing loss function combined with ℓ_1 -regularization. The goal $\min_W \sum_{i=1}^n L(w^T x_i, y_i) + \lambda \sum_{l=1}^L \|W_l\|_1$ utilizes a separation

ranking loss $L(z, y) = \max_{k_n \in N(y), k_p \in P(y)} (1 + z_{k_n} - z_{k_p})_+$, where $P(y)$ and $N(y)$ represent positive and negative labels. The ℓ_1 -norm enforces sparsity in primal (feature weights) and dual (support labels) spaces. PD-Sparse uses the fully corrective block coordinate Frank-Wolfe algorithm to deal with non-smooth objectives, with importance sampling speeding up the detection of violating labels. The method attains sublinear training time in terms of the number of labels and is thus appropriate for extreme-scale settings.

The *Robust Extreme Multi-Label Learning* (REML) (Xu et al. 2016) approach recognizes that low-rank assumptions are valid only after head and tail labels are separated. REML factorizes label matrix Y into a low-rank component \hat{Y}_L and a sparse component \hat{Y}_S , minimizing $\min_{\hat{Y}_L, \hat{Y}_S} \|Y - \hat{Y}_L - \hat{Y}_S\|_F^2$ subject to $\text{rank}(\hat{Y}_L) \leq k$ and $\text{card}(\hat{Y}_S) \leq s$. The model $\hat{Y}_L = XUV$ and $\hat{Y}_S = XH$ combines low-rank and sparse predictors, optimized through alternating minimization. Clustering labels into evenly sized sets facilitates parallel processing, whereas ℓ_1 -regularization over H provides sparsity. The hybrid nature of REML is shown to exhibit enhanced performance in cases with heterogeneously distributed label frequencies.

Improving on SLEEC’s drawbacks, AnnexML incorporates label-aware clustering and ranking-based embeddings. In contrast to random k-means clustering, AnnexML builds a k-nearest neighbor graph (KNNG) (Dong et al. 2011) over label vectors and partitions it to maintain connectivity. Each partition’s embedding space is learned through a pairwise ranking loss that focuses on accurate neighbor orderings. Approximate nearest neighbor search in the embedding space returns candidate labels at prediction time, with a re-ranking step providing final output refinement. The technique’s reliance on ranking losses and graph-based clustering lessens its dependency on Euclidean assumptions and makes it better for predicting tail labels.

Feature dimensionality is another problem in XMC, which is solved by *DEFrag* (Deep Embeddings and Feature ReAgglomeration) (Jalan & Kar 2019). DEFrag agglomerates features hierarchically into balanced clusters, aggregating features in clusters to form meta-features. For a feature matrix $X \in \mathbb{R}^{n \times d}$, clusters $\{F_1, \dots, F_K\}$ produce aggregated features $\tilde{X}_{ik} = \sum_{j \in F_k} X_{ij}$. DEFrag builds binary trees through spherical k-means splits, bounding each leaf at most d_0 features. In contrast to PCA or random projections, sparsity and non-negativity are maintained under DEFrag, which are important for retainability in XMC text-based applications. The theoretical analysis demonstrates aggregated features of DEFrag can’t degrade the performance

of a linear classifier more than original features, offering dimensionality reduction grounded in principles.

Comparative study indicates trade-offs among these strategies. LEML and REMBRANDT perform well on cases with tight global label correspondences but poorly on tail labels. SLEEC and AnneXML (Tagami 2017) mitigate this by focusing on local structures but at the expense of increased computation. PD-Sparse and REML find a balance between sparsity and resilience and are effective on imbalanced datasets. DEFrag’s feature agglomeration acts as a complementary method, providing orthogonal improvements when dealing with high-dimensional inputs. Empirical tests on benchmarks such as Wiki-500K and Amazon-3M demonstrate SLEEC and AnnexML having better precision@k scores, whereas PD-Sparse and REML dominate propensity-weighted metrics by more effectively dealing with tail labels.

Recent developments combine linear algebra techniques with deep learning. Methods such as *DeepXML* employ linear models for negative sampling and shortlisting, and then neural networks for final ranking. This hybrid strategy brings the scalability of linear techniques together with the representational capability of deep learning to attain state-of-the-art performance. For example, ASTEC (Approximate Shortlister with Tree-based Classifiers) uses label clustering for candidate selection and fine-tunes deep classifiers over shortlisted labels. These strategies underscore the continuing applicability of linear algebra to the computational complexity inherent in XMC, even as deep learning dominates other machine learning domains.

Overall, linear algebra techniques offer invaluable techniques for addressing the extreme multi-label classification challenge. By means of low-rank approximations, sparse optimization, and creative embeddings, these techniques solve the two-fold challenge of scalability and precision. Though there is no universal technique that overpowers all contexts, the field has developed to provide a full toolkit that is flexible enough to suit varied data properties-whether balancing head and tail labels, coping with high-dimensional features, or leveraging local versus global patterns. Directions in the future can be towards closer linking with deep architectures, hyperparameter search automation, and theoretical guarantees for new hybrid models.

2.3 Tree based Methods

Tree-based approaches in extreme multi-label classification take advantage of the intrinsic hierarchical structures of data to accelerate the learning and prediction processes. The underlying approach in these approaches is to iteratively partition either the sample space or the label space, effectively shrinking the search space at prediction time. Final predictions in most instances are achieved by aggregating simple models such as one-vs-all classifiers or averaging-based approaches.

These tree-based methods generally fall into two broad categories based on the partitioning strategy. Label partitioning-based methods divide the set of labels into clusters for every level in the hierarchy, resulting in a structured multi-level representation. Every node in the hierarchy has a multi-label classifier that predicts the optimal subset of labels at a given level. The last clusters, i.e., the leaves of the tree, are a small set of labels, and they can be easily handled with simple classifiers like one-vs-all models.

The second approach is the sample or instance partitioning-based methods, in which the training samples are divided and not the labels. This gives a tree-like structure in which the samples are divided into various groups at each level. Incoming test points are processed through this tree according to branch decisions made from classifiers, ultimately reaching a leaf or terminal node. Leaves calculate the final prediction by averaging the labels of the samples included within that cluster. This algorithm generally has a k-nearest neighbor-like structure, especially regarding how final predictions are made.

A visual comparison of these two strategies emphasizes that while label partitioning strategies partition the output space, instance partitioning strategies partition the input space and therefore define different ways of addressing extreme classification tasks. Label partitioning strategies like top-k clustering try to alleviate label prediction by reducing the search space, while instance partitioning tries to group similar samples together and use the local neighborhood information for making the final predictions.

In general, tree-based methods offer a realistic framework for scaling very large multi-label problems by mitigating search and training difficulties. They also seamlessly integrate with shallow as well as deep-learning models, delivering astounding performance gains with structured prediction and efficient decision sequences.

Multi-Label Random Forest Classifier (MLRF) is an extension of the basic random forest approach to solve extreme multi-label classification problems. MLRF partitioning is almost instance-oriented and less label-oriented. At the time of constructing the tree, a standard measure like Gini impurity is used to direct the splitting decisions like vanilla decision trees. One of the most significant properties of MLRF is that tree growth goes on until the number of distinct or active labels at each node reaches a number which is logarithmic in terms of the number of labels. The method ensures label space complexity reduction in a structured manner at each step, and one can work with very large sets of labels in an efficient manner.

Both during training and during prediction, MLRF is identical to a regular random forest in its process, i.e., many trees are trained independently on bootstrapped samples, and their predictions are combined to provide the final prediction. This approach assists in generating good performance through ensemble averaging, but without sacrificing the simplicity and interpretability of decision trees. However, besides these strengths, MLRF also has some drawbacks that affect its performance, particularly when compared with more advanced methods applied for extreme classification.

One of the main weaknesses of MLRF is that it cannot consider label ranking. In nearly all most extreme multi-label scenarios, label priority by relevance is key to performance, yet MLRF addresses all labels as if they were equal without considering priority. This will cause suboptimal retrieval performance, particularly in those cases where proper label ordering matters most. Another drawback comes from the general tree shape constructed by MLRF. These trees are also not balanced, and therefore longer training and inference times occur. An unbalanced tree structure can possibly have some branches significantly longer than others, and this creates inefficiencies during model construction as well as during prediction time.

However, MLRF remains a solid baseline procedure because it is simple to apply, simple to implement, and in good accord with standard random forest routines. It offers a good point of departure for the application of tree-based ensemble algorithms to problems created by overwhelmingly large sets of labels for multi-label classification problems.

FastXML (Prabhu & Varma 2014) is a tree-based model tailored to solve the problems of extreme multi-label classification. Unlike other methods that conduct label-based partitioning, FastXML uses an instance-based partitioning strategy in

which the instances’ division at each node is informed by the normalized Discounted Cumulative Gain (nDCG) measure. In contrast to purity-based thresholds often employed by decision trees, nDCG reflects both label relevance and label ranking, hence a better objective for multi-label environments. nDCG’s value ranges between 0 and 1, with larger values representing a better partitioning of instances according to label relevance.

The tree building starts with the random allocation of data points into the left and right child nodes. Random partitions of this kind tend to produce a low nDCG score. Therefore, FastXML uses an alternating minimization process that repeatedly refines the split by maximizing the nDCG score. The partitioning of instances into left and right branches naturally creates a partition over the corresponding labels, strengthening the tree structure relative to the target labels. This recursive partitioning goes on until the number of distinct, active labels at a leaf node becomes logarithmic with the total number of labels to maintain leaf sizes.

One major strength of FastXML compared to techniques such as MLRF is its explicit label ranking consideration in the partitioning scheme. Additionally, FastXML empirically builds more balanced trees and offers significant training and inference efficiency boosts. These aspects counteract the typical flaws related to unbalanced tree structures and ranking-irrelevant splitting measures in previous research.

The optimization in every split node is the minimization of a loss function consisting of an L1 regularization and a term with the objective to maximize the nDCG score. The weights learned, associated with hyperplanes separating instances, are saved and then reused on inference time. This repeated optimization during both the training and test phases guarantees the model’s quality and applicability.

For prediction, FastXML combines the probabilities of labels over all trees in the ensemble. The ultimate top- k predictions for an instance are found by ranking labels by the combined probabilities from the leaves traversed by the instance. By this ensemble approach, FastXML provides stable performance on huge datasets with millions of labels.

One-vs-all strategies create a fundamental basis in extreme multi-label classification, and DiSMEC (Babbar & Schölkopf 2017) is an extremely efficient variation of the strategy. DiSMEC attempts to eliminate scalability problems through a sparse and distributed training mechanism. Under one-vs-all’s framework, a separate binary classifier is learned per label, each with the objective of discrimination between the

presence or absence of the respective label. DiSMEC uses an L2 regularization term to prevent overfitting and maintain the generalization ability of the classifiers.

The training task of DiSMEC is to minimize a loss function with a balance of the regularization penalty and the empirical loss. Specifically, it minimizes the summation of the squared L2 norm of the weight vector and a hinge-loss term that favors correct label discrimination by a margin. Each binary classifier is learned independently on the feature vectors to obtain a dedicated weight vector belonging to its corresponding label. The sparsity of the learned weight vectors is critical to make the model computationally efficient, even for datasets with millions of labels.

During inference, the feature vector of a novel instance is dotted with the weight vector of each label to estimate the probability that the label should be associated with the instance. The approach allows for direct estimation of the relevance of each label, and thus the prediction process is straightforward conceptually and interpretable. However, inference time for DiSMEC grows linearly with the label set size, which can be a performance bottleneck when faced with extremely large label spaces.

While its inference time is higher than tree-based algorithms like FastXML, DiSMEC is more likely to yield better predictive accuracy. Its label independence on individual binary classifiers allows it to better learn label-specific decision boundaries, leading to improved performance, particularly in datasets where fine-grained label distinctions are important. The speed-accuracy trade-off is a characteristic of DiSMEC, making it the go-to choice when predictive performance is more important than computational efficiency. Overall, DiSMEC illustrates that one-vs-all formulations, if properly optimized to consider sparsity and distributed training, are still highly competitive in the extreme multi-label classification domain.

With regards to extreme multi-label classification, Parabel (Prabhu et al. 2018) proposes a label tree partitioning methodology that strikes the right balance between high predictive accuracy and training efficiency as well as inference efficiency. In contrast to conventional one-vs-all approaches such as DiSMEC and PPDSparse (Yen et al. 2017), which typically incur high computational costs, Parabel is very efficient in managing large label sets by cleverly reducing the number of training points each classifier needs to process. This is done by arranging the labels into a balanced binary tree, such that the hierarchy assists in grouping similar labels together according to their feature representations.

The fundamental concept in Parabel(Prabhu et al., Parabel) is to recursively divide the set of labels into two different groups to create a binary tree structure. The division goes on until every node in the tree has at least a predefined minimum number of labels, represented as M . At a leaf node, the algorithm stops, and two kinds of binary classifiers are trained at every internal node to help determine whether a sample moves to the left child node or right child node. At every leaf node, a collection of one-vs-all classifiers are trained, each dedicated to predicting the existence of a certain label from the group of labels that node represents.

At prediction, an input example traverses the tree by making a choice at each internal node, guided by the learned classifiers, until arriving at a leaf. Having arrived at a leaf, a label ranking is computed using the scores from the corresponding one-vs-all classifiers. In order to support effective label splits, Parabel describes each label as a normalized vector based on the distribution of training samples that it is paired with. The label splitting is defined as a clustering problem, resolved with a two-means objective augmented by an extra balancing term. The clustering promotes balanced splits and uses cosine similarity so that the obtained partitions are semantically coherent.

This partitioning and clustering process is recursively applied to each node so that a balanced tree architecture is ensured. Partitioning stops when a node’s label set becomes small enough and therefore treating nodes with fewer than M labels as leaves and reducing the problem to multi-label classification at that time. After tree construction, Parabel also needs training classifiers predicting the probability of an instance traversing a specific branch if its parent node is active, expressed as a Maximum A Posteriori (MAP) estimation problem. A binary variable is added to each node here to represent whether a label should be turned on, depending on the child’s relationship with its parent node.

At the leaf node, an independent optimization is performed where one-vs-all classifiers are learned to output the final label probabilities for an instance that arrives at the leaf. While the optimization problems throughout the tree are independent, this does not mean that Parabel is oblivious to label correlations. Rather, it takes such dependencies into account explicitly when conducting the initial label clustering, clustering labels that tend to co-occur together.

In prediction mode, the problem simplifies to determining the top- k labels with highest probability for an instance. The instance’s chance of being given a label is

calculated as the product of probabilities along the path from the root node to the leaf node containing the label. In particular, the probability for a label y_t is the product of transition probabilities for the path from the root to the leaf at which y_t appears. Beam search techniques, such as greedy Breadth-First Search (BFS), are applicable to determine the most likely paths and therefore the top- k most confident labels accordingly.

By this hierarchical decomposition of the label space and effective binary classification methods, Parabel brings about both scalability and high accuracy, and it is thus a viable contender among algorithms for extreme multi-label classification problems.

The CRAFTML (Siblini et al. 2018) approach proposes an innovative method of extreme multi-label classification based on the merging of random forests and a speedy partitioning process. In contrast to conventional algorithms like FastXML, CRAFTML is built around a k -means based partition strategy instead of being optimized to aim directly at nDCG. An important departure from FastXML is the use of k -way splits instead of binary splits in building the tree, for improved diversity as well as efficiency. Moreover, CRAFTML uses random forest techniques wherein random projections are used to project over both feature and label spaces with a view to enhancing diversity among trees.

The algorithm builds a forest consisting of m_F trees where each tree is constructed recursively through k -ary partitioning. The recursive subdivision at each node proceeds until one of three things happens: the number of instances in the node drops below some threshold $n_{textleaf}$, all of the features are the same among instances, or all of the instances have the same labels. After the tree is completed, each leaf node stores the mean label vector for the instances which it was given, to serve as a foundation for future prediction.

At training time, CRAFTML decomposes the process of node training into three separate steps. In the first step, the label and feature vectors for instances within a node are projected into a lower-dimensional representation via a random projection matrix. This projection matrix is either created from a Standard Gaussian Distribution or by sparse orthogonal projections. After dimensionality reduction, k -means clustering is used to divide the projected instances into k clusters. The clustering centroids are initialized by the k -means++ method, which is a better clustering quality than random initialization alone. For improving the clustering, the centroids are

recalculated by averaging vectors in each temporary cluster before assigning instances based on proximity to these recalculated centroids.

Consequently, a k -means classifier is learned at every internal node, partitioning the instances into k subsets of child nodes. At prediction time, an instance traverses the tree based on decisions of the learned k -means classifiers at each level. When reaching a leaf node, prediction for the instance is computed based on the mean label vector stored in that leaf. Combining the outputs of many trees in the forest leads to more precise end predictions, with the advantage of the ensemble effect and the various partitionings developed during training.

ExtremeRegression (XReg) (Prabhu et al. 2020) is a method that is specifically designed for XMC, or label prediction for an extremely large number of labels for each sample. XReg provides solutions to difficulties encountered by straightforward multilabel classification and presents effective solutions when the number of labels is extremely high and the inter-label relations are intricate. For extreme multilabel tasks, the general algorithms do not have the capacity to cope with the computational cost and the sparsity of the label distributions. XReg, on the other hand, applies regression techniques adopted from the multilabel scenario to simultaneously predict multiple labels while managing computational complexity.

The general goal of XReg is to transform the multilabel task into a regression task where a label is considered as a continuous output. This approach can easily deal with cases that have numerous labels by employing regression models that predict one label at a time, possibly from a common feature space. This approach sidesteps exposing the model to the constraint of directly dealing with high-dimensional label spaces. Apart from the computation efficiency, XReg also minimizes redundancy in the traditional multilabel methods, for example, one-vs-rest or binary relevance, as it solves the problem in a more comprehensive way.

The other important feature of XReg is that it can scale with increasing growing labels and therefore can be applied to real-world problems where thousands of labels per instance are to be predicted. As opposed to other methods, one per label or label-ranking-based, which is inefficient, XReg offers a single model that can predict multiple labels simultaneously, leading to an enormous improvement in prediction time as well as model quality.

Further, XReg can naturally learn label correlations through exploiting the dependence among labels with regression-based methods. This means it performs best

with interdependence-label datasets, which an independent solution would be unable to achieve under the intrinsic organization. With label dependencies as a part of the regression model, XReg ensures that it does not just generate correct predictions for the appropriate labels but is also aware of interdependence between the labels and so generates better, more understandable predictions.

Briefly, ExtremeRegression presents a different method of handling XMC issues by transforming them into regression issues. It is a more effective, scalable, and precise mechanism of predicting multiple labels, especially when handling challenging and interrelated label sets. Its capacity for handling label dependency and minimizing computation costs makes it an ideal application for large-scale multilabel classification issues across different fields.

2.4 Deep learning methods

The XML-CNN (Liu et al. 2017)(eXtended Multi-Level Convolutional Neural Network) model is a specially designed neural network architecture for extreme multi-label text classification. It effectively extracts both local and global semantic representations from the text by utilizing multi-level convolution and dynamic max pooling. Unlike other conventional models based on bag-of-words alone or RNNs, XML-CNN takes advantage of the convolutional structure to represent different n-gram features at different scales.

From a given input text sequence described by an embedding matrix $X \in \mathbb{R}^{l \times d}$, where l denotes the sequence length and d is the dimension of the embeddings, the model uses several convolutional filters with different window sizes to extract local features. A filter $w \in \mathbb{R}^{h \times d}$, where h denotes the window size, is used over a window of h words to generate a feature. Mathematically, the feature c_i for the i -th window is calculated as:

$$c_i = f(w \cdot X_{i:i+h-1} + b)$$

Here, b is a bias term and f is a non-linear activation function, usually the ReLU function.

Following convolution, dynamic max-pooling is performed across the sequence in order to find the most useful features in the various regions:

$$p = \max\{c_1, c_2, \dots, c_{l-h+1}\}$$

This ensures that the model is strong to different sequence lengths and emphasizes the most important features.

The pooled features across various filters and window sizes are concatenated to create an overall document representation vector z . This vector is then input into a fully connected layer and a sigmoid activation for multi-label classification:

$$\hat{y} = \sigma(Wz + b')$$

where W and b' are the final layer's weights and biases, and σ is the element-wise sigmoid function.

To learn the model, the binary cross-entropy loss is minimized:

$$\mathcal{L} = - \sum_{i=1}^m [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

where y_i are the actual labels and \hat{y}_i are the predicted probabilities for each label.

Therefore, XML-CNN efficiently integrates convolutional operations, dynamic pooling, and fully connected layers to address the issues of extreme multi-label text classification tasks.

2.5 Transformers

2.5.1 Introduction

(Vaswani et al. 2017) RNNs, particularly LSTMs and GRUs, have long been considered the state-of-the-art for sequence modeling and transduction tasks, including language modeling and machine translation. These models have been extensively used to process sequential data by capturing temporal dependencies, and many advancements have been made to push the limits of recurrent networks, especially in the context of encoder-decoder architectures.

One of the key limitations of recurrent models is their inherently sequential nature, which processes data one step at a time. This sequential computation is particularly problematic when handling long sequences, as it precludes parallelization within training examples. This limits the batching of examples due to memory constraints. While various optimizations, such as factorization tricks and conditional

computation, have improved the computational efficiency of RNN-based models, the fundamental sequential computation constraint remains.

In recent years, attention mechanisms have gained prominence as a powerful tool for modeling dependencies in sequence data. These mechanisms allow models to capture relationships between input and output sequences regardless of their distance, making them highly effective in tasks like machine translation and language modeling. However, most attention mechanisms are still used in conjunction with recurrent models. The Transformer model, however, entirely eliminates recurrence and relies exclusively on attention mechanisms. By doing so, Transformers can model global dependencies between input and output sequences in parallel, resulting in significantly faster and more efficient training. This architecture has also proven effective for time series data, where parallelization can greatly enhance model performance and training time.

2.5.1.1 Encoder

Each encoder layer has two sub-layers. The multi-head self-attention mechanism, and the position wise fully connected feed-forward network. Residual Connection and Normalization is added in each sub-layer. The attention mechanism allows the model to weigh and "pay attention" on different parts of the input sequence based on the context. The FFN network consists of two linear layers with ReLU activation function in between. Adding residual information yields stronger gradients, helps remember positional encoding and normalization reduces bias, prevents weight explosion.

2.5.2 Model Architecture

2.5.2.1 Decoder

The decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to encoder layer, residual connections and sub-layer normalization is added. The self-attention is modified in the decoder stack to prevent positions from attending to subsequent positions. This masking ensures that the predictions for position i can depend only on the known outputs at positions less than i . The decoder operates in an autoregressive manner, kickstarting its process with a start token. After the decoder, the word or token with the highest probability is the final output.

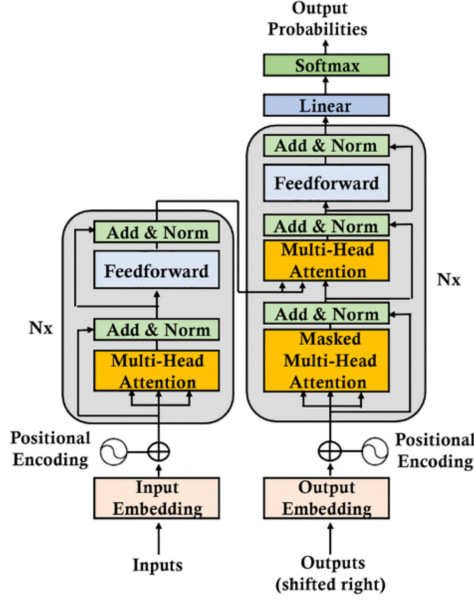


Figure 2.1: The Transformer: Model Architecture (Vaswani et al. 2017)

2.5.2.2 Attention

Scaled Dot-Product Attention: Linear transformations of the whole input sequence embeddings are done to generate the key, query and value matrices. A dot product matrix multiplication is performed between the queries and keys, resulting in the creation of a score matrix. The scores are then scaled down by dividing them by the square root of the dimension of the query and key vectors. Softmax function is applied to the adjusted scores to obtain the attention weights which emphasizes higher scores while diminishing lower scores. Now the weights derived from the softmax function are multiplied by the value vector, resulting in an output vector.

$$Q = X_{\text{Embedding}} \times W_Q$$

$$K = X_{\text{Embedding}} \times W_K$$

$$V = X_{\text{Embedding}} \times W_V$$

$$\text{Attention Score} = \text{softmax} \left(\frac{QK^\top}{\sqrt{d_k}} \right)$$

$$\text{Attention Output}(Q, K, V) = \text{softmax} \left(\frac{QK^\top}{\sqrt{d_k}} \right) V$$

Multi-Head Attention: All of the process starts by generating queries, keys and values h times by using different weight matrices. This process happens separately in each of these smaller stages or 'heads' and is done in parallel. Finally the attention outputs of each of these heads are concatenated. The concatenated output is then multiplied by another weight matrix to bring it the same dimension as that of the attention input.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

Masked Multi-Head Attention: During training, we give the correct target sentence to the decoder and it predicts the output so that it is as similar as possible to the target sequence. But, if the decoder has access to every word in the target sequence, the model will not be able to learn its parameters properly. The decoder would know what the next words should be and it will harm the ability of the model to predict unseen words. To prevent this, we use masking. For example, if the decoder is supposed to produce the 4th output token, we should mask every word from index 5 until the EOS token.

Encoder-Decoder Multi-Head Attention In this attention layer we take key and value vector from the Encoder output and the query vector from the Decoder output. By linear transformations, we make two copies of the output of the encoder and consider them to be key and value vectors. The output of the previous decoder layer, i.e. the query vector contributes to the calculation of attention weights. This allows every position in the decoder to attend over all positions in the input sequence, integrating information from both encoder and decoder.

2.5.2.3 Position-wise Feed-Forward Networks:

This consists of two linear transformations with a ReLU activation in between. While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is d_{model} , and the inner-layer has dimensionality d_{ff} .

2.5.2.4 Embeddings and Softmax:

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model} . We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities.

2.5.2.5 Positional Encoding:

Position and order of words are crucial in any language, but the Transformer architecture itself doesn't inherently recognize the position of each word in a sequence. To address this, we introduce positional encoding, which adds information about each word's position within a sentence. This encoding is a d -dimensional vector that helps the model understand the relative or absolute position of tokens in the sequence.

The positional encoding must satisfy several criteria: it should provide a unique encoding for each time step (word's position), maintain consistent distances between time steps across sentences of varying lengths, be generalized to longer sentences without additional effort, have bounded values, and be deterministic. Sinusoidal functions for the positional encodings. Finally, the input to the multi-head self attention is given as:

$$\psi'(w_t) = \psi(w_t) + \vec{p}_t$$

Here, $\psi(w_t)$ is the embedding for a particular word w_t and \vec{p}_t is the positional encoding for that word in the sentence.

2.6 T5 Transformer

2.6.1 Introduction

(Raffel et al.,) The T5 (Text-to-Text Transfer Transformer) introduced by Google researchers in 2020, redefines natural language processing (NLP) by unifying diverse tasks under a single framework: text-to-text. Traditional NLP models often require task-specific architectures (e.g., BERT for classification, GPT for generation), complicating deployment across applications. T5 addresses this by reframing all tasks—translation, summarization, question answering, and classification—as text generation problems. For example, translating "Hello" to German becomes input "translate English to German: Hello" with output "Hallo". This approach simplifies

the NLP pipeline, enabling a single model to handle multiple tasks through task-specific prefixes.

The model explores the limits of transfer learning by pre-training on a massive, diverse corpus (the Colossal Clean Crawled Corpus, C4) and fine-tuning on downstream tasks. Building on predecessors like BERT and GPT, T5 adopts an encoder-decoder Transformer architecture, bridging the gap between encoder-only (e.g., BERT) and decoder-only (e.g., GPT) models. The authors systematically analyze factors like model size, training objectives, and dataset composition, providing insights into scalable NLP systems.

2.6.2 Model Architecture

2.6.2.1 Encoder-Decoder Structure

The T5 model employs an encoder-decoder Transformer architecture, distinguishing itself from encoder-only (BERT) or decoder-only (GPT) designs by combining both components for flexible text generation. The encoder processes input text to generate context-aware representations through stacked self-attention layers, while the decoder autoregressively produces output text by attending to both the encoder’s output and previously generated tokens. Central to T5’s design is its text-to-text framework, which unifies diverse NLP tasks—such as translation, classification, and summarization—by formatting all inputs and outputs as text. During pre-training, T5 adopts a denoising objective inspired by masked language modeling, where random spans of tokens are masked, and the model predicts the missing spans sequentially. This span-based masking encourages longer-range coherence compared to BERT’s word-level approach. This architecture balances versatility, scalability, and efficiency, enabling state-of-the-art results across NLP tasks.

2.6.2.2 Unified Text-to-Text Framework

(Raffel et al. 2020) A key innovation of T5 is its text-to-text approach. Unlike other models that use task-specific architectures, T5 treats both input and output as text strings. This unified framework simplifies transfer learning and allows the same model, hyperparameters, and loss function to be applied across diverse tasks. T5 employs a masked span prediction objective during pretraining. Firstly, masked spans are replaced with sentinel tokens. Then the model predicts the full masked spans rather

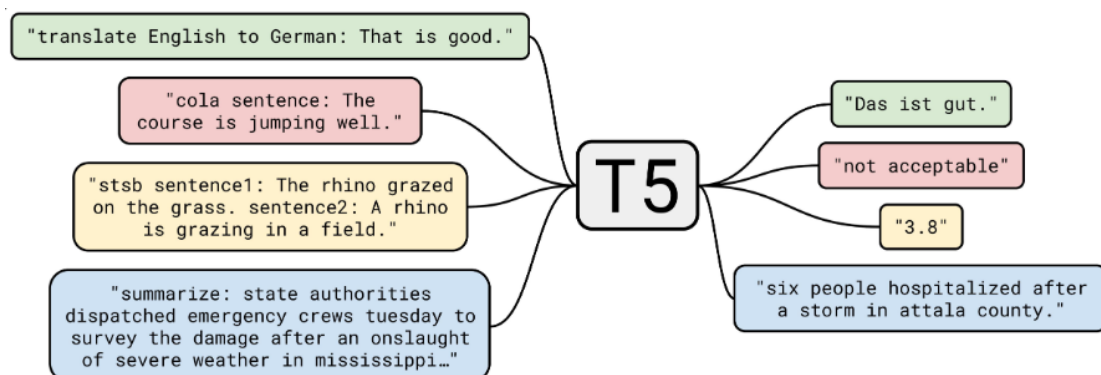


Figure 2.2: T5 (Text to Text) Transformer (Raffel et al. 2020)

than individual tokens. This approach differs from BERT’s token-level masking and aligns more closely with BART’s span-level masking, but with unique enhancements.

The T5 architecture handles tasks like classification, translation, summarization, and question answering under a unified framework. It uses large-scale pretraining to capture general language patterns, which can be fine-tuned for specific tasks. It also captures long-range dependencies effectively due to its Transformer-based design and demonstrates strong performance across languages via cross-lingual transfer learning. T5 was trained on the C4 dataset for one month using teacher forcing and maximum likelihood estimation. Compared to BERT base, T5 required only one-fourth as many pretraining steps while achieving comparable or superior results. T5 can be applied successfully in machine translation, text summarization, question answering and text classification. Its ability to handle diverse tasks under a single framework has made it widely adopted in research and industry.

The T5 model is pre-trained on the C4 dataset, a cleaned and filtered version of the Common Crawl corpus, where aggressive text filtering (removing code, duplicates, and placeholder text) significantly enhances dataset quality and model performance. For transfer learning, the authors employ multi-task learning during fine-tuning, training the model on multiple tasks simultaneously to improve generalization, and introduce adapter layers—small, task-specific modules inserted into the pre-trained network—to enable parameter-efficient adaptation without retraining the entire model. Key experiments highlight the superiority of the denoising objective (span masking) over autoregressive (GPT-style) and deshuffling approaches, demonstrating its ability to capture longer-range dependencies. Scaling experiments reveal that larger models

(e.g., 11B parameters) achieve state-of-the-art results but with diminishing returns relative to their computational cost, while zero-shot learning tests show modest capabilities, such as inferring tasks from prompts. Despite its success, T5 faces limitations: training the 11B model demands immense resources, limiting accessibility, and the text-to-text framework occasionally struggles with structured outputs (e.g., parsing), underscoring trade-offs between versatility and task-specific performance.

Fine-Tuning : T5 is fine-tuned for each task after pretraining by having task-specific output and input examples. Every task is mapped to a text-to-text task. For instance, for summarization, the input may be a string such as "summarize: The quick brown fox jumps over the lazy dog." and the output would be a shorter form such as "A quick brown fox jumps over a lazy dog." For machine translation, the input may be "translate English to German: How are you?" and the output would be "Wie geht es dir?" For sentiment analysis, the problem is reformulated as text generation where the input "classify sentiment: The movie was fantastic!" has the output "positive." For question answering, the input can be "question: What is the capital of France? context: France is a country in Europe. The capital of France is Paris." and the output would be "Paris."

Changes to Base Transformers :T5 makes a few significant changes to the base transformer model in an effort to better accommodate text-to-text tasks. One of the changes involves the incorporation of relative positional encodings, which, as compared to fixed positional encodings, enables the model to better generalize across different sequence lengths. Another variant is pre-normalization, where layer normalization is introduced before attention and feedforward layers for enhancing gradient flow and training stability. T5 also employs a reduced training task, the unified span corruption task, which makes it efficient for a large number of downstream tasks.

Applications of T5 :T5's performance and adaptability make it efficient for various NLP applications. It can be utilized in text summarization, where it condenses text from documents, articles, or reports. In machine translation, T5 can translate between languages. For question answering, it can generate or extract the answer from a provided context. T5 can also do text classification tasks like sentiment analysis, spam filtering, or topic classification. In addition to this, it can also be utilized in the creation of conversational AI and dialog systems.

T5 Advantages :There are various advantages of T5. The monolithic nature of its architecture eliminates task-specific architectures, and it can hence be fine-tuned

to carry out any task. The model is scalable in size, ranging from small up to 11 billion parameters, and can be fine-tuned to serve variable computational requirements. T5 is on par with the cutting edge on many NLP evaluations, and has been found to do well. It can easily be fine-tuned to novel tasks by reformulating the input and output.

Challenges: Despite its power, there are also some weaknesses in T5. Its bigger sizes like T5-3B and T5-11B do demand a huge amount of computations, which could act as an impediment in making them available for small organisations. Inference speed for generation models like T5 can be slower than in classification approaches for some tasks. Fine-tuning T5 into specialized or specific tasks also needs a considerable quantity of data to improve performance, which in itself is a disadvantage where there is not much data.

Chapter 3

Solving a E-commerce Problem

3.1 Problem Statement and its significance

3.1.1 Introduction

The rapid growth of e-commerce platforms has transformed consumer shopping habits, with billions of products and millions of transactions occurring daily. Despite this impressive expansion, one significant challenge faced by both sellers and platforms is the maintenance of precise and comprehensive product metadata. This metadata is typically organized as attribute-value pairs, which are essential for ensuring efficient product searches, accurate personalized recommendations, and an overall enhanced shopping experience for customers. These attribute-value pairs serve as the foundation for many key features, from filtering search results to tailoring product suggestions based on customer preferences. However, ensuring the consistency and accuracy of this structured data is far from trivial. With an ever-expanding catalog of items, it becomes increasingly difficult to ensure that every product is accurately described, classified, and tagged with the correct attributes. As e-commerce continues to evolve, the challenge of managing product metadata grows, requiring platforms and sellers to invest in advanced data management techniques, automated systems, and artificial intelligence tools to improve the accuracy and consistency of product attributes. Addressing this challenge effectively is essential for sustaining competitive advantage in the e-commerce space and delivering a seamless and satisfying shopping experience to consumers.

3.1.2 Problem Statement

Attribute-Value Prediction From E-Commerce Product Descriptions: Despite their importance, product descriptions provided by sellers are often incomplete or unstructured, leading to missing crucial attributes. This lack of structured data hampers the platform’s ability to:

- Optimize search engines for relevant query retrieval.
- Provide precise and personalized product recommendations.
- Enable effective question-answering systems for customers.
- Improve the overall user experience through detailed and organized product catalogs.

Given the massive scale of modern e-commerce catalogs managed by platforms like Amazon, Walmart, and Alibaba, manual curation of attribute-value pairs is impractical. Automating this process through machine learning models that can extract and predict attribute-value pairs from unstructured product descriptions becomes a necessity.

3.1.3 Objective

The objective of this task is to develop a machine learning model that can automatically predict attribute-value pairs for a given product based on:

- **Product Title:** A concise summary of the product.
- **Product Description:** Additional unstructured textual information.
- **Optional Metadata:** Details about the store and manufacturer (which are not mandatory for use).

The model must predict:

- **Category Values:** The hierarchical levels of categories to which the product belongs.

3.1.4 Key Challenges

- **Unstructured Data:** A significant challenge in e-commerce is dealing with unstructured data, especially product descriptions. These descriptions often contain irrelevant or extraneous information and lack consistent formatting. As a result, extracting meaningful insights or categorizing products accurately becomes difficult. The lack of structure complicates tasks such as search optimization, data categorization, and personalized recommendations.
- **Scale:** E-commerce platforms manage vast catalogs containing millions of unique products. This scale presents a unique challenge in handling large datasets efficiently. Models used to process product information must be capable of managing and analyzing extensive data sets, ensuring quick retrieval of relevant product information while maintaining performance. Handling such large-scale data requires sophisticated algorithms and scalable infrastructure to ensure efficiency.

3.1.5 significance of solving the problem

A robust solution to the challenge of maintaining accurate and comprehensive product metadata in e-commerce platforms can have far-reaching positive effects. First, structured metadata significantly improves the search and discovery process, enabling customers to find the most relevant products quickly. This is essential in an environment where users are often overwhelmed by the sheer number of options available. Additionally, the accurate prediction and management of product attributes enhance the personalization of recommendations, directly impacting sales. Customers are more likely to purchase products that are tailored to their preferences, leading to higher conversion rates and customer satisfaction. Moreover, automating catalog management by reducing the need for manual data entry minimizes human error and operational costs. By streamlining this process, platforms can ensure that product information remains up to date and consistent, reducing the workload on staff. Furthermore, providing detailed and accurate product information builds trust with customers, fostering a sense of reliability and transparency. When customers can easily access all relevant product details, they are more likely to feel confident in their purchasing decisions, improving the overall shopping experience. In conclusion, solving

the problem of product metadata management not only improves platform performance but also enhances customer engagement, satisfaction, and long-term loyalty, all of which are crucial for sustained success in the competitive e-commerce market.

3.2 Analysis of Dataset

- **Total Records:** 746,502
- **Training Dataset:** 561,838 samples
 - **Training:** 449,470 samples
 - **Validation:** 112,368 samples
- **Test Dataset:** 184,664 samples

3.2.1 Features (X)

- Description
- Retailer
- Price

3.2.2 Labels (Y)

- Supergroup
- Group
- Module

3.2.3 Important Characteristics

- Presence of extensive hierarchical labels enables structured and organized prediction.
- Textual descriptions are often sparse and noisy, necessitating the use of advanced language modeling techniques.

- Metadata features such as *Retailer* and *Price* play a vital role in resolving ambiguities.
- The label distribution follows a long-tail pattern, resulting in a highly imbalanced dataset.

3.2.4 Input features

Description	Retailer	Price
Seachem Entice- Natural Scent & Flavor Enhance...	Seachem Laboratories, Inc.	10.75
OSHA Danger Sign - Hazardous Pool Chemicals ...	SignMission	7.99
Harpy Motors Automotive 12oz Aerosol Spray Pai...	Harpy Motors	46.98
Hubbell CS8165C Locking Plug, 50 amp, 480V, 3 ...	Hubbell	107.17
Halex, 1-1/2 in. Rigid Conduit Locknut , 96195...	Jensen (Home Improvement)	6.91
ORRAINE HOME FASHIONS Adirondack Tier Curtain...	Lorraine Home Fashions (Home)	9.99
Design Toscano QL1936 9 in. Queen Cleopatra on...	Design Toscano	51.90
APC Smart UPS XL 2200 SU2200XL Compatible Repl...	UPS Battery Center	199.99
Whats Up Nails - Bow Vinyl Stencils for Christ...	Whats Up Nails	3.75
Intro-Tech Automotive LX-25-R Ultimate Reflect...	Intro-Tech	57.25
3 Front 2 Rear Full Leveling Lift Kit -Street ...	STREET DIRT TRACK	73.90
LINKWELL Black Queen Bee Pillow Cover 18x18 in...	LINKWELL Home Decor	10.99
Seaguar STS Salmon Fishing Line, Strong and Ab...	Seaguar	22.99
3dRose What Happens Stays at GrammyS Two Tone ...	3EROS	16.92
Dorman 939-178 15 x 6 In. Steel Wheel Compatib...	Dorman Products	68.99

Figure 3.1: sample of Input data

The dataset consists of three primary features used for training the model:

- **Description:**
 - text representing the item details.

- Often contains informal, abbreviated, or noisy information.
- **Example:** Seachem Entice- Natural Scent & Flavor Enhancer for Fish Food, Marine & Freshwater Aquariums 250ml
- **Challenges:**
 - * Requires robust language modeling to understand incomplete or abbreviated words.
 - * Needs to capture subtle semantic meaning to distinguish similar products.
- **Retailer:**
 - Name of the store or seller.
 - **Example:** Seachem Laboratories, Inc
 - **Importance:**
 - * Provides auxiliary information regarding product type or specialization (e.g., an organic-focused store likely sells eco-friendly items).
 - * Helps resolve ambiguities when the description alone is unclear.
- **Price:**
 - Numeric value representing the product cost.
 - **Example:** 10.75
 - **Importance:**
 - * Helps differentiate between similar items at different price points (e.g., "car air freshener" vs "car essential oil" might differ significantly in price).
 - * Acts as a soft signal for certain product categories (luxury vs economy products).

3.2.5 Output labels

Supergroup	Group	Module
Pet Supplies	Fish & Aquatic Pets	Aquarium Water Treatments
Industrial & Scientific	Occupational Health & Safety Products	Safety Signs & Signals
Automotive	Paint & Paint Supplies	Paints & Primers
Tools & Home Improvement	Electrical	Plugs
Tools & Home Improvement	Electrical	Electrical Boxes, Conduits & Fittings
Home & Kitchen	Home Dcor Products	Window Treatments
Home & Kitchen	Home Dcor Products	Home Dcor Accents
Electronics	Computers & Accessories	Computer Accessories & Peripherals
Beauty & Personal Care	Foot, Hand & Nail Care	Nail Art & Polish
Automotive	Interior Accessories	Sun Protection
Automotive	Replacement Parts	Shocks, Struts & Suspension
Home & Kitchen	Bedding	Decorative Pillows, Inserts & Covers
Sports & Outdoors	Hunting & Fishing	Fishing
Home & Kitchen	Kitchen & Dining	Dining & Entertaining
Automotive	Tires & Wheels	Wheels

Figure 3.2: sample of Output data

Each training sample is mapped to a hierarchical label structure consisting of three levels:

- **Supergroup:** High-level broad category, such as *homecare* or *food ambient*.
- **Group:** Intermediate level offering more refined categorization within the supergroup.
- **Module:** The most granular level representing the final specific class.

Brand information is provided but was not used for training or evaluation in this work.

3.2.6 Observations

- The level of label specificity varies based on the product and data volume; certain labels are highly detailed , whereas others remain broad .
- Some modules represent rare or infrequent categories, making them valuable benchmarks for assessing model performance on underrepresented classes.
- The hierarchical labeling framework allows leveraging inter-label connections, enhancing model generalization, especially in scenarios with limited training data.

3.3 Data Preprocessing

- The product data, along with hierarchical classification labels, was sourced from JSON files.
- Each product sample in the training dataset (561,838 entries) was annotated with module, group, and supergroup labels.
- The dataset encompassed fields such as product descriptions, retailer information, and pricing details.

Chapter 4

Methodology and Experimentation

4.1 Analysis and Observations

1. The **module** represents the most specific category in our hierarchy.
2. Each label in the specific category (i.e., module) has a **one-to-one correspondence** with the higher-level categories: *SuperGroup* and *Group*. This one-to-one mapping eliminates the need to predict the *SuperGroup* and *Group* categories separately.
3. Therefore, during the training phase, we will construct a mapping from each module to its corresponding higher-level categories. In other words, we will store the relationship from the most specific category upward.

Example:

Bluetooth Earphones \Rightarrow {Headphones(Group), Electronics(SuperGroup)}

4. During inference, whenever our model outputs a label like “Bluetooth Earphones” (since the model is designed to predict only the most specific category), we can use the precomputed mapping to retrieve its corresponding higher-level labels (e.g., Headphones(Group), Electronics(SuperGroup)).
5. As a result, our model’s prediction space is simplified: it only needs to output among the modules, significantly reducing complexity and improving efficiency.

4.2 XMC with only T5 Transformer

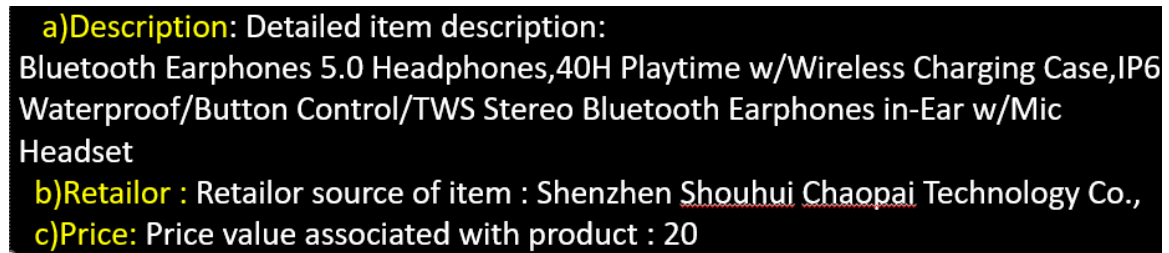
In this work, we fine-tune the **T5** model for the task of predicting the module (class) label based on product-related information. The fine-tuning process is carefully designed by framing the problem as a **sequence-to-sequence** task, leveraging the original design philosophy of the T5 (Text-To-Text Transfer Transformer) architecture.

4.2.1 Input Construction

The input to the T5 model is constructed by concatenating three key attributes from the dataset:

- **Description** (textual description of the product),
- **Retailor** (the name of the retailer or seller),
- **Price** (the numerical price information).

These three fields are combined into a single input string, separated appropriately to maintain clarity for the model



a)Description: Detailed item description:
Bluetooth Earphones 5.0 Headphones,40H Playtime w/Wireless Charging Case,IP6
Waterproof/Button Control/TWS Stereo Bluetooth Earphones in-Ear w/Mic
Headset
b)Retailor : Retailor source of item : Shenzhen Shouhui Chaopai Technology Co.,
c)Price: Price value associated with product : 20

Figure 4.1: Example



Retailer: {retailer} Price: {price} Description: {text}

Figure 4.2: Input format to T5

This unified input string is fed into the encoder of the T5 model.

4.2.2 Output Construction

The corresponding output for each input is the **module label**, representing the most specific category of the product. For instance, given the above input example, the model would be expected to output:

Bluetooth Earphones

Thus, the model learns to map a structured but concatenated text input into a concise text output (the target module), fully utilizing the sequence-to-sequence capabilities of T5.

4.2.3 Problem Framing

The fine-tuning task is framed as a **sequence-to-sequence prediction** problem. Specifically:

- **Input sequence:** Concatenated string of Description, Retailor, and Price.
- **Output sequence:** Module (specific category label).

This framing is advantageous because T5 was pre-trained using a variety of text-to-text tasks, making it naturally suited for problems where both inputs and outputs are sequences.

4.2.4 Training Details

During training:

- The model is initialized with the pre-trained weights of T5-Large.
- Standard cross-entropy loss is used, comparing the predicted token sequence with the ground truth module sequence.
- **Hyperparameters:** Batch Size: 32 samples per device; Learning Rate: 1×10^{-4} ; Optimizer: AdamW; Input text tokenized with a maximum length of 28 tokens.

4.2.5 Advantages

This approach has several advantages:

- Simplifies the multi-feature input problem into a unified text representation.
- Makes use of powerful pre-trained language representations from T5-Large.
- Avoids the need for complex feature engineering by treating the problem as pure text generation.
- Enables easy extension to incorporate additional metadata fields if necessary, simply by modifying the input string format.

Thus, by fine-tuning T5-Large in this manner, we aim to achieve high-quality predictions of the module labels based on diverse product information in a natural and effective way.

4.3 Introduction to Hierarchical K means clustering

Now in this section we think of a approach that can mitigate tail label problem and we will try to integrate with the T5 transformer

In order to efficiently structure the module labels and facilitate better semantic understanding for the downstream fine-tuning of the T5 transformer, we propose a **Hierarchical K-Means Clustering** based organization of the labels. This method not only captures semantic relationships among modules but also emphasizes fair representation of both head and tail labels, crucial for handling imbalanced label distributions.

Step 1: BERT Embedding Extraction

The first step involves generating semantic vector representations for all module labels. We use pre-trained **BERT** (Bidirectional Encoder Representations from Transformers) to extract high-dimensional embeddings for each module. Specifically, each module label (which is a short textual phrase) is passed through BERT, and the corresponding token representation is extracted as its dense embedding vector.

Step 2: Initialization with Root Node

Initially, we consider all module embeddings as part of a single **root node**. This root node contains the entire set of module vectors and serves as the starting point for the hierarchical clustering process.

Step 3: Controlled Node Branching to Address Tail Label Problem

At each node, rather than splitting into only two clusters (as in traditional binary hierarchical clustering), we opt for a higher branching factor. Specifically, we set the number of children to **6** at each node.

This design is crucial because:

- In binary clustering (two children per node), there is a strong risk that head labels dominate early splits, forcing tail labels to be merged unnaturally with unrelated head labels.
- By allowing more children (e.g., six), we provide **greater flexibility for tail labels** to find clusters of their own without being forced to merge prematurely.
- This significantly improves **semantic purity** within clusters and ensures that rare (tail) labels are adequately and fairly represented in the hierarchy.
- Consequently, tail labels receive stronger, more meaningful representation, which is essential for subsequent tasks that suffer from label imbalance.

Step 4: Recursive K-Means Clustering

At each node, **K-Means clustering** is applied with $k = 6$. The child nodes formed by clustering are then processed recursively:

1. If a node has more than 6 module vectors, it is eligible for further clustering.
2. K-Means clustering is applied again on this node’s embeddings.
3. The process repeats recursively until a node has fewer than 6 module vectors.
4. Nodes with fewer than 6 modules are designated as **leaf nodes**.

This recursive clustering strategy results in a multi-level hierarchical tree where each path from root to leaf captures increasingly finer-grained semantic distinctions among modules.

Step 5: Label Encoding through Prefix Codes

Once the hierarchical tree is built, we encode each module label using a unique sequence of integers that represents its path in the hierarchy.

For example:

- Suppose the path from the root to a module involves choosing child 0 at the root, then child 0 at the next level, and then reaching a leaf node.
- In that case, **Module 1** can be encoded as $[0, 0, 0]$.
- If another module, **Module 2**, shares the first two splits but diverges at the third level by choosing child 1, it would be encoded as $[0, 0, 1]$.

Thus, each module obtains a **unique hierarchical code** based on its position in the cluster tree.

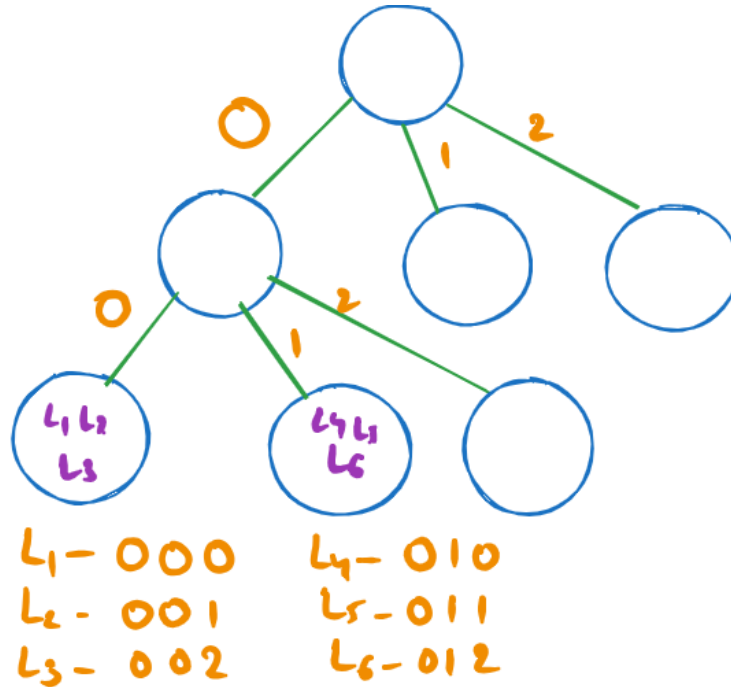


Figure 4.3: Visualisation of label encoding

Semantic Closeness via Prefix Matching

An important property of this encoding is that:

- Labels that share longer prefixes in their code sequences are **semantically closer**.
- For instance, modules $[0, 0, 0]$ and $[0, 0, 1]$ are much closer than modules $[0, 0, 0]$ and $[2, 1, 3]$.

Prefix matching thus directly captures the semantic similarity between different modules, making the code structure meaningful beyond simple identification.

Benefits for T5 Fine-Tuning

The hierarchical encoding obtained through this process plays a critical role when fine-tuning the T5 model for label prediction:

- The T5 model can now **focus more naturally on capturing semantic similarities** among labels, since similar modules will have highly overlapping prefix codes.
- **Tail labels** receive better representation, since they are no longer overshadowed by head labels in the clustering stage.
- During training, errors involving similar prefixes (e.g., predicting $[0,0,1]$ instead of $[0,0,0]$) are **semantically less severe** compared to errors involving completely different prefixes, thereby allowing the model to be graded more intelligently.
- Additionally, the hierarchical organization acts as an **implicit regularizer**, encouraging the model to learn general cluster structures before fine-grained label distinctions.

This combination of semantic structure, tail label protection, and hierarchical prefix encoding ultimately leads to **stronger generalization performance** and more robust label prediction in downstream tasks.

I have applied clustering on some other data set to ensure its robustness of working. Please find the figure 4.3 as for output some dataset before label encoding and figure 4.4 as clustered output of the dataset we are working with.

```
[246(agricultural machinery) 247(agricultural sciences) 248(agriculture)]
[ 650(astrology) 651(astronomy) 653(astronomy & space) 654(astronomy & space science) 11046(solar panels) 11353(star wars)]
[ 8601(paleobiology) 8602(paleontology)]
[ 439(animal ears) 440(animal husbandry) 441(animal psychology)]
[1280(biological & natural sciences) 1281(biological sciences) 1282(biology) 1285(biometrics) 1287(biostatistics) 1307(birds)]
[ 4266(environmental economics) 4268(environmental policy) 4269(environmental science) 4270(environmental studies)]
[ 262(air compressors) 267(air conditioning & heating) 268(air conditioning line repair tools) 269(air conditioning oils)]
[ 4219(engine & chassis parts) 4221(engine blocks) 4228(engine heaters & accessories) 4236(engine parts) 4237(engine filters) 4373(exhaust sys
[ 4358(exercise bikes) 4359(exercise machine accessories) 12370(treadmill belts) 12371(treadmill lubricants)]
[12929(water filtration & softeners) 12930(water garden kits) 12946(water quality & instrumentation) 12951(water temp)]
```

Figure 4.4: Clustered output on some other dataset before label encoding to ensure robustness

```
'bleach ammonia': array([1, 1, 0, 5, 0]),
'garden & flora': array([0, 5, 0]),
'stationery & printed material & services': array([1, 2, 1, 0]),
'homecare merchandise': array([1, 2, 3, 1]),
'skin conditioning moisturising': array([1, 4, 5, 1, 0]),
'wine still light table styles': array([3, 2, 0]),
'sugar candy': array([3, 0, 0, 0]),
'snacks chips crisps reconstituted extruded': array([5, 1, 1, 0]),
'skin cleansing & toning': array([1, 4, 5, 4]),
'meat products fresh': array([5, 4, 0]),
'dog food dry': array([5, 5, 0]),
'meat cuts joints whole fresh fw': array([5, 4, 3]),
'eggs egg products fresh': array([5, 0, 2, 0]),
'chocolate single variety': array([5, 2, 2, 0]),
'cough cold & other respiratory remedies & accessories': array([1, 0, 3, 0]),
'fruit orange fresh fw': array([5, 3, 0, 0]),
'cheese fresh fw': array([5, 0, 0, 0]),
```

Figure 4.5: Clustering output of the dataset we are working

4.4 XMC with Hierarchical K means clustering + T5

In order to further enhance the performance of module label prediction and particularly improve the handling of **tail labels**, we integrate the hierarchical clustering output directly into the T5 Large fine-tuning pipeline. This approach combines the semantic structure obtained from hierarchical clustering with the powerful sequence-to-sequence modeling capabilities of T5.

Input-Output Format for Fine-Tuning

The fine-tuning task is formulated as a classic **sequence-to-sequence problem**:

- **Input:** The input to the T5 model is the concatenation of three fields from the input data: **Description**, **Retailer**, and **Price**. These fields are concatenated into a single string to provide a rich textual context about the product.
- **Output:** The output is the **label encoding** generated through hierarchical clustering. Specifically, instead of outputting the raw module name, the model is trained to predict the corresponding sequence of integers (prefix code) that represents the module’s position in the cluster hierarchy.

Advantages of Using Clustered Label Encoding

The decision to use hierarchical cluster-based label encoding as the output for fine-tuning offers several significant advantages, especially in the context of improving tail label performance:

1. Semantic Structure in the Output Space:

- Since the label encodings are generated from hierarchical clustering of BERT embeddings, modules that are semantically closer share longer prefixes.
- This introduces a structured similarity in the output space, allowing the model to make semantically “close” mistakes rather than entirely unrelated ones.
- As a result, the T5 model is encouraged to learn the underlying semantic structure among labels, not just memorize isolated labels.

2. Improved Handling of Tail Labels:

- Tail labels, which are often underrepresented in the dataset, tend to form smaller clusters or unique branches in the hierarchical structure.
- By explicitly representing them with distinct but semantically meaningful prefix codes, the model learns to predict them not as rare isolated labels, but as natural extensions of more common categories.

- Therefore, tail labels gain **better representation and exposure during training**, reducing the bias towards head labels.

Training Strategy

The T5 Large model is fine-tuned using a supervised learning setup where the loss is computed based on the match between the predicted label sequence (prefix code) and the ground truth encoding obtained from the cluster hierarchy.

Special attention is given to:

- Ensuring balanced mini-batches that contain examples from both head and tail labels.
- Using hierarchical-aware evaluation metrics, where partial prefix matches are credited accordingly.
- **Hyperparameters:** Batch Size: 32 samples per device; Learning Rate: 1×10^{-4} ; Optimizer: AdamW; Input text tokenized with a maximum length of 28 tokens.

Impact on Tail Label Performance

By integrating hierarchical clustering into the fine-tuning process:

- Tail labels receive stronger training signals through unique but semantically meaningful encodings.
- The model moves beyond simple frequency-based learning and towards **semantic similarity learning**.
- This shift leads to a substantial improvement in recall and precision for rare module classes, which are typically the hardest to predict accurately.

Thus, the combination of BERT-based hierarchical clustering with T5 fine-tuning offers a powerful and effective strategy for robust label prediction, especially in domains characterized by highly skewed label distributions.

Chapter 5

Results and Discussion

5.1 Impact of Hierarchical Clustering

- Hierarchical clustering contributed to substantial improvements in module classification accuracy, particularly enhancing performance on rare (tail) labels, while preserving inference speed.
- By applying prefix tree constraints, the model was able to enforce structured outputs, ensuring both semantic coherence and robust generalization across the long-tail label distribution.
- These findings strongly support the notion that organizing labels hierarchically is an effective strategy for tackling the persistent tail label challenge in XMC.

5.2 Evaluation Metrics

Table 5.1: Comparison of evaluation metrics

Metric Name	Purpose
Accuracy	% of correct module predictions (overall)
Precision	Correctly predicted labels out of total predicted
Recall	Correctly predicted labels out of total true labels
F1-Score	Harmonic mean of Precision and Recall
Tail Accuracy	Accuracy computed only over rare/tail modules

Table 5.2: Comparison of Plain T5 vs. Hierarchical Clustering + T5

Metric	Plain T5	Hierarchical Clustering + T5
Accuracy	79.18%	83.8%
Precision	78.5%	82.6%
Recall	68.2%	74.1%
F1-Score	72.9%	78.0%
Tail Accuracy	40.7%	55.8%

5.3 Discussion

Integrating hierarchical clustering with the T5 model proves to be a powerful and reliable strategy for addressing the tail label challenge in Extreme Multi-Label Classification (XMC). Our experimental results consistently showed that organizing the label space hierarchically and enforcing constraints on output generation leads to notable improvements in prediction accuracy, especially for infrequent labels. we will further explore the broader impact of our results, examine both the advantages and limitations of the proposed methodology, and discuss potential avenues for extending this approach to other domains

5.4 Generalisation capability towards Tail labels

A key highlight from our experiments is the enhanced generalization ability of the model towards infrequent (tail) labels. Traditional deep learning models typically overfit to head labels due to their overwhelming frequency in training datasets. In contrast, hierarchical clustering mitigates this imbalance by preventing head labels from dominating the learning process. Grouping semantically similar labels enables the model to leverage knowledge from frequent labels and effectively transfer it to rarer ones. This strategy fosters smoother decision boundaries and yields richer representations for categories with limited samples.

5.5 Scalability

The hierarchical clustering method offers natural scalability to extremely large label spaces. Rather than handling each label as an isolated entity, the model predicts traversal paths within a hierarchy, substantially lowering computational demands.

scalability is vital for real-world applications in areas like e-commerce platforms, recommendation systems, and search engines, where the label space can easily expand to thousands or even millions of categories.

5.6 Practical Impact

The proposed method has significant potential to enhance the performance of large-scale systems:

- **Search Engines:** Enhanced label prediction improves document classification and retrieval, leading to more accurate and relevant search results.
- **Content Management Systems:** Reliable multi-label assignments support better structuring, filtering, and recommendation of digital assets.
- **E-commerce Platforms:** More precise product categorization boosts search relevance, strengthens personalized recommendations, and elevates customer satisfaction.

By effectively addressing the tail label challenge, platforms are empowered to serve niche segments and long-tail markets, thereby opening up new business opportunities and revenue streams.

5.7 Limitations and Challenges

While the hierarchical clustering method offers several benefits, it is not without drawbacks:

- **Memory Requirements:** Although the approach reduces the decoding search space, storing and managing the prefix tree introduces memory overhead, which can become substantial when dealing with extremely large label sets.
- **Dependency on Clustering Quality:** The overall success of the method is highly sensitive to the quality of the initial hierarchical clustering. Suboptimal clustering can negatively impact predictive performance.
- **Fixed Tree Structure:** The prefix tree, once established, remains static during training. Any inaccuracies or errors in the tree’s construction may be propagated during prediction, affecting the model’s reliability.

5.8 Broader Applicability

Although this study concentrated on product categorization within e-commerce, the underlying ideas of hierarchical clustering and constrained decoding have wide-ranging applications. Potential areas for extension include:

- **Scientific Paper Indexing:** Enabling accurate categorization of niche research fields that are often underrepresented.
- **Medical Diagnosis Systems:** Supporting rare disease identification, where limited training data often hampers performance.
- **Legal Document Classification:** Ensuring precise assignment of fine-grained, infrequent categories critical for legal workflows.

Enhancing performance on tail labels paves the way for democratizing access to information, guaranteeing that rare but important categories receive proper attention rather than being marginalized.

5.9 Conclusion

In conclusion, the integration of hierarchical clustering with the T5 model offers a compelling direction for improving Extreme Multi-Label Classification (XMC). This approach effectively balances scalability, predictive accuracy, and equitable treatment of underrepresented labels. Despite some remaining challenges, it establishes a solid groundwork for advancing future research in large-scale classification tasks and personalization systems.

Bibliography

- Babbar, R. & Schölkopf, B. (2017), Dismec: Distributed sparse machines for extreme multi-label classification, *in* ‘Proceedings of the tenth ACM international conference on web search and data mining’, pp. 721–729.
- Bhatia, K., Jain, H., Kar, P., Varma, M. & Jain, P. (2015), ‘Sparse local embeddings for extreme multi-label classification’, *Advances in neural information processing systems* **28**.
- Bloom, B. H. (1970), ‘Space/time trade-offs in hash coding with allowable errors’, *Communications of the ACM* **13**(7), 422–426.
- Chen, Y.-N. & Lin, H.-T. (2012), ‘Feature-aware label space dimension reduction for multi-label classification’, *Advances in neural information processing systems* **25**.
- Dasgupta, A., Katyan, S., Das, S. & Kumar, P. (2023), ‘Review of extreme multilabel classification’, *arXiv preprint arXiv:2302.05971* .
- Dong, W., Moses, C. & Li, K. (2011), Efficient k-nearest neighbor graph construction for generic similarity measures, *in* ‘Proceedings of the 20th international conference on World wide web’, pp. 577–586.
- Huang, Q., Shrivastava, A. & Wang, Y. (2018), ‘MACH: Embarrassingly parallel k -class classification in $o(d \log K)$ memory and $o(k \log K + d \log K)$ time, instead of $o(kd)$ ’.
- URL:** <https://openreview.net/forum?id=r1RQdCg0W>
- Jalan, A. & Kar, P. (2019), ‘Accelerating extreme classification via adaptive feature agglomeration’, *arXiv preprint arXiv:1905.11769* .

- Jasinska, K. & Karampatziakis, N. (2016), ‘Log-time and log-space extreme classification’, *arXiv preprint arXiv:1611.01964* .
- Liu, J., Chang, W.-C., Wu, Y. & Yang, Y. (2017), Deep learning for extreme multi-label text classification, *in* ‘Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval’, pp. 115–124.
- Mineiro, P. & Karampatziakis, N. (2015), Fast label embeddings via randomized linear algebra, *in* ‘Joint European conference on machine learning and knowledge discovery in databases’, Springer, pp. 37–51.
- Prabhu, Y., Kag, A., Harsola, S., Agrawal, R. & Varma, M. (2018), Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising, *in* ‘Proceedings of the 2018 World Wide Web Conference’, pp. 993–1002.
- Prabhu, Y., Kusupati, A., Gupta, N. & Varma, M. (2020), Extreme regression for dynamic search advertising, *in* ‘Proceedings of the 13th international conference on web search and data mining’, pp. 456–464.
- Prabhu, Y. & Varma, M. (2014), Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning, *in* ‘Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining’, pp. 263–272.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. & Liu, P. J. (2020), ‘Exploring the limits of transfer learning with a unified text-to-text transformer’, *Journal of machine learning research* **21**(140), 1–67.
- Siblini, W., Kuntz, P. & Meyer, F. (2018), Craftml, an efficient clustering-based random forest for extreme multi-label learning, *in* ‘International conference on machine learning’, PMLR, pp. 4664–4673.
- Tagami, Y. (2017), Annexml: Approximate nearest neighbor search for extreme multi-label classification, *in* ‘Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining’, pp. 455–464.
- Tai, F. & Lin, H.-T. (2012), ‘Multilabel classification with principal label space transformation’, *Neural Computation* **24**(9), 2508–2542.

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017), ‘Attention is all you need’, *Advances in neural information processing systems* **30**.
- Xu, C., Tao, D. & Xu, C. (2016), Robust extreme multi-label learning, *in* ‘Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining’, pp. 1275–1284.
- Yen, I. E., Huang, X., Dai, W., Ravikumar, P., Dhillon, I. & Xing, E. (2017), Ppdspare: A parallel primal-dual sparse method for extreme classification, *in* ‘Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining’, pp. 545–553.