Enhanced Multi-Label Question Tagging on Stack Overflow: A Two-Stage Clustering and DeBERTa-Based Approach

Isun Chehreh
IASBS
Zanjan, Iran
a.chehreh@iasbs.ac.ir

Farzaneh Saadati
University of Georgia
Athens, Georgia
farzaneh.saadati@uga.edu

Ebrahim Ansari
IASBS
Zanjan, Iran
ansari@iasbs.ac.ir

Bahram Sadeghi Bigham
Alzahra University
Tehran, Iran
b_sadeghi_b@alzahra.ac.ir

***Abstract*—This paper introduces a novel method for automatically classifying questions with multiple labels, using data specifically sourced from Stack Overflow. Traditional tagging methods frequently face challenges due to the complexity and semantic diversity of these questions, resulting in inconsistent and sometimes inaccurate results. The process starts with preprocessing to remove any unwanted elements. Next, we convert the questions into meaningful representations using SMPNet. The semantic vectors obtained are then processed using UMAP to help us understand the overall structure of the data and make it easier to cluster similar items.**

After dimensionality reduction with UMAP, we use the K-Means method to group the questions into clusters, with the best number of groups determined by the Silhouette Score. Finally, a fine-tuned DeBERTa model is trained for each cluster to accurately predict the appropriate tags. Our approach significantly outperforms traditional methods, achieving 2% improvement over the best baseline. This strategy improves model efficiency by narrowing the focus to specific subsets of data.

I. Introduction

Stack Overflow is a popular platform where programmers at all skill levels go to find answers to coding problems. Each user can ask questions and get help from others who might have solutions. This website is continuously growing, and as of November 2022, four new questions were being added every minute, with over 34 million answers and 100 million monthly visits from developers. This growth has made effective content organization a challenge [1] [2].

Tags in Stack Overflow are important tools that describe the subjects of questions, helping to sort them into clear and specific groups. Tags are crucial for directing questions, improving searches, helping users navigate, and enhancing overall organization. Given the large number of users and questions, tagging has become one of the most efficient methods for organizing content [3] [4]. However, assigning appropriate tags manually is not without challenges. Users with different levels of knowledge sometimes add irrelevant tags, leading to inconsistencies in tagging, which, in turn, affect search precision and content organization.

Automatically assigning tags to questions on Stack Overflow is a significant challenge in the field of Natural Language Processing (NLP). The platform's vast and diverse content makes automatic tag assignment a complex task. Despite the clear advantages of automation, such as reducing human errors and ensuring consistency, accurately modeling the rich semantic context of questions remains a daunting task [1]. Previous studies have shown that adding appropriate tags can significantly enhance information retrieval, improving both precision and recall in search results [1]. However, existing methods often fail to fully leverage the context and depth of information present in questions and answers.

In this study, we introduce a method to automatically tag questions on Stack Overflow using the pre-trained DeBERTa model [5]. DeBERTa (Decoding-Enhanced BERT with Disentangled Attention) is an advanced NLP model built on BERT [6] and RoBERTa [7]. DeBERTa stands out among NLP models due to its advanced attention mechanisms, which allow it to capture rich semantic information from text. This capability makes it particularly well-suited for handling the complexity of Stack Overflow's diverse content, enabling more accurate and contextually relevant tag assignments. Our multi-stage proposed method uses these advantages to make tagging more efficient and accurate, which helps in better organizing and finding information on Stack Overflow.

The contribution of this paper is structured as follows: Section II succinctly presents an overview of related research; Section III elaborates on the proposed methodology; Section IV discusses an in-depth analysis with an explanation of the experimental results; and Section V concludes this paper.

II. Related Work

Saha et al. [8] developed a method for predicting tags using the Support Vector Machines (SVM) algorithm. Their approach involves two main steps. First, an algorithm generates a list of possible tags by examining the given question. Then, a second algorithm checks the degree of similarity between the question and the model. Xia et al. [9] introduced TagCombine, a system that automatically suggests tags by combining three special methods. These methods help improve tagging for items without tags by using techniques such as ranking based on similarity, ranking for multiple labels, and ranking based on tag terms.

Building on TagCombine, Short et al. [10] propose a hybrid tag recommendation system, combining text-based techniques from TagCombine with network-based recommendation approaches, which is called NetTagCombine. This approach combines traditional text analysis methodologies, such as TF-IDF, with network-oriented strategies so that a better prediction about tags may be carried out more precisely. Saini and Tripathi [11] suggested a way to predict tags for posts on Stack Overflow, focusing on the 1,000 most common tags. They used a method called multi-label classification, specifically a one-vs-all approach with a linear SVM. They tried other methods like Naive Bayes and unique feature extraction.

Alreshedy et al. [12] proposed a classifier to predict programming languages from questions on Stack Overflow, using the XGBoost algorithm. They cleaned the data, which included titles, bodies, and code snippets, and then processed it using TF-IDF, with a minimum document frequency set to 10. Jain and Lodhavia [13] studied how to predict tags for questions on Stack Overflow using two methods: a random forest classifier and a k-nearest neighbor algorithm. They found that the k-nearest neighbor method worked better than the random forest method for this task.

Khezrian et al. [14] performed tag prediction using Tag-BERT, which was based on the BERT framework. BERT is a language representation model that utilizes a bidirectional Transformer architecture, allowing it to consider both left and right contexts simultaneously. During the pre-training phase, the model learns masked language modeling and next-sentence prediction tasks that build deep contextualized embeddings for words. In this research, the pretraining and embedding operations were carried out by the BERT module. Output from BERT was transferred to a textual model for further processing through convolutional neural networks (CNNs) and deep neural network (DNN) layers. Xu et al. [15] proposed the Post2Vec architecture that utilizes the embedding of words with Word2Vec and predicts tags on Stack Overflow posts. The preprocessing, input layer, feature extraction layers, and feature combination module with tag prediction comprise the main parts of the Post2Vec model.

Devine and Blincoe [16] suggested using Extreme Multi-label Classification (XMLC) for predicting tags on Stack Overflow without needing labeled data. They tested fourteen different pre-trained models and found that the MPNet model, which was partly trained on data without tags, worked the best. Their research also showed that models trained on specific types of text, like titles or question-answer pairs, can be very good at classifying data without any prior training, especially when there isn't much-tagged data available for a particular topic.

He et al. [17] developed the PTM4Tag model that enhances the recommendations of tags for posts on Stack Overflow using pre-trained models. The proposed model mainly operates in three steps: pre-processing, feature extraction, and classification. In the pre-processing stage, the raw data from Stack Overflow is organized into titles, descriptions, and code snippets. These are then processed during the feature extraction stage using pre-trained models, such as BERT, to generate feature vectors that are combined through a fusion layer into a unified representation of the post. In the classification stage, this is finally converted into a tag vector that shows the likelihood of every tag. Utilizing a multi-layer neural network aligns the post representation with the most appropriate tags, using different pre-trained models for each of the components.

Erjon et al. [18] applied multiple sentence embeddings based on BERT to represent Stack Overflow questions. They used a k-nearest neighbor multi-label classifier to predict tags by exploiting similarities in feature vector representations. Among the methods tested, RoBERTa-Sentence proved to be the most effective for sentence embedding, demonstrating superior performance in the tagging task. Subramani et al. [19] explored tag prediction for Stack Overflow questions using deep learning models. They worked with the StackSample dataset, focusing on the top 50 tags. Their approach involved training MLP, LSTM, and GRU models, with the results showing that the LSTM and GRU models outperformed the MLP model.

Chehreh et al. [20] developed a multi-stage method for tagging Stack Overflow questions. The process starts with NLP techniques to preprocess the questions, followed by embedding them using the MPNet-based sentence transformer model. First, candidate tags are extracted using the YAKE algorithm, and second, candidate tags are identified through one of the multi-label k-nearest neighbors, multi-label Random Forest, or cosine similarity methods. In the next stage, overlapping tags are selected and scored, with a genetic algorithm normalizing features and assigning weights to identify the top five tags.

III. Approach

This section starts by introducing the dataset used in this research and then explains the method we propose in detail. It outlines the key characteristics of the data and the rationale behind the chosen approach, setting the foundation for the subsequent analysis and implementation.

A. Dataset Analysis

Stack Overflow has become a crucial resource for developers and researchers, resulting in the development of various datasets and tools for studying its content. The Stack Overflow dataset comes in different versions, and these datasets are useful for many NLP tasks. The version we're using can be found on Kaggle [21].

The dataset includes 10% of the text from questions and answers on the Stack Overflow website and is divided into three separate files. The questions file holds the ID, title, body, creation date, closure date, score, and owner ID for all non-deleted Stack Overflow questions. The answers file contains the body, creation date, score, and owner ID for each answer, with the ParentId column connecting it to the corresponding question in the questions file. Furthermore, the tags file provides the ID and tags for each question [21].

This dataset includes 1,264,218 questions and 3,750,995 tags, with 37,036 unique tags. Each question in this dataset is associated with up to 5 tags, with an average of 2.9 tags per question. The 20 most common tags in this dataset are shown in a word cloud in Fig. 1.

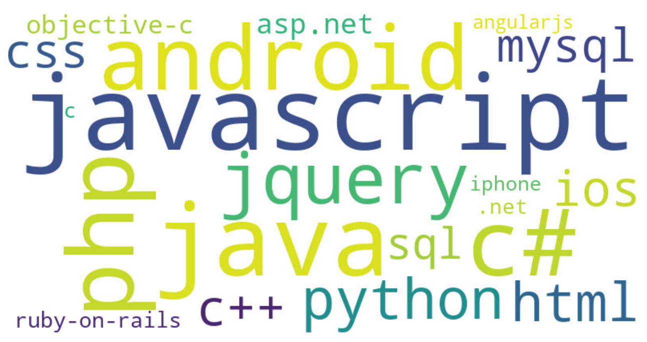

Fig. 1. Illustrates the 20 most frequent tags of the dataset by word cloud

B. Proposed Method

The structure and main steps of the proposed method are visualized in Fig. 2. This figure provides a clear overview of the approach, highlighting the main components and workflow involved in the method.

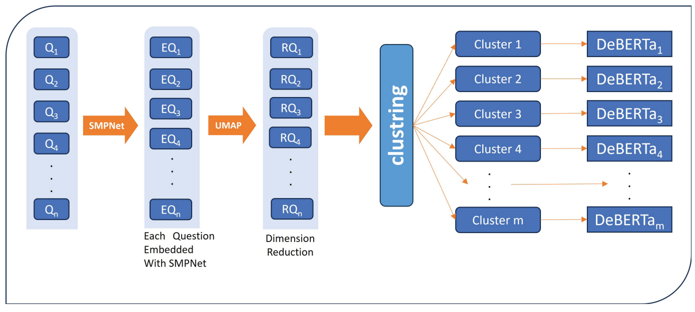

Fig. 2. Illustrates the proposed method

The primary goal of this approach is to develop a multi-label classification model that can correctly predict the right tags for each question. Since questions can have different meanings and need various labels, this method uses a two-step process: first, clustering questions by how similar they are in meaning, and then adjusting separate models for each cluster. This plan helps the model work better and faster because each model deals with a more similar part of the data.

In the first stage, the questions need to be pre-processed to remove any extra noise and to transform the text into a form that is effective for clustering and classification. During this stage, the text is converted to lowercase to reduce case sensitivity. Additionally, URLs and HTML tags are removed to clean the text of extra noise. Punctuation and numbers are also removed, leaving only letters and spaces. After this, the text is tokenized, and common and insignificant words that do not affect meaning (stopwords) are removed. Finally, words are lemmatized to return them to their base form, ensuring that all variations of a word are considered the same unit.

After pre-processing, the questions are converted into semantic vectors. For this purpose, we use SMPNet [22], a sentence transformer model that is based on MPNet [23] and fine-tuned on one billion sentence pairs from various NLP tasks.

MPNet is a version of transformer models [24] that leverages the advantages of BERT and XLNet [25]. XLNet is a general autoregressive pretraining model. XLNet models bi-directional context using a training strategy called Permuted Language Modeling (PLAM). This technique allows the model to learn dependencies across left and right contexts by considering all the possible permutations of the input factorization. MPNet model uses PLM to more effectively capture dependencies between predicted tokens. Additionally, MPNet integrates auxiliary positional information as input, allowing the model to process entire sentences and reduce positional discrepancies that exist in XLNet.

MPNet has undergone pre-training on a substantial text corpus exceeding 160 GB. SMPNet, employing Siamese and triplet network architectures akin to SBERT [26], derives significant semantic embeddings from sentences. SBERT, the first sentence transformer model, reduced the time required to find the most similar pair from 10,000 sentences to about 5 seconds by making modifications to BERT. Thus, while both SBERT and SMPNet use similar structures, SMPNet is more accurate and efficient, using newer methods compared to SBERT. This model can turn semantically similar sentences into vectors that are close to each other.

After obtaining the semantic vectors from SMPNet, these vectors are further processed using UMAP [27] for dimensionality reduction. UMAP helps in preserving the global structure of the data while reducing the dimensionality, making it more efficient for clustering. By reducing the dimensionality of the vectors, UMAP enables the K-Means clustering algorithm to work more effectively, especially in terms of both computational efficiency and the quality of the clusters formed. The proportion of dimensions before and after applying UMAP is illustrated in Fig. 3. This pie chart highlights the significant reduction in dimensions achieved through UMAP.

After converting the sentences into lower-dimensional vectors with UMAP, they are clustered using the K-Means algorithm [28]. Determining the optimal number of clusters is a key challenge in clustering. An incorrect choice can result in inadequate separation of sentences and, consequently, complicate the classification models. To tackle this issue, we utilize the Silhouette Score criterion [29]. The Silhouette Score is a metric for evaluating clustering quality, combining intra-cluster and inter-cluster information. Specifically, the Silhouette Score rewards clustering solutions that show both compactness within individual clusters and clear separation between clusters. This metric generally takes a value between -1 and 1; a value close to 1 indicates good clusters, while a value close to -1 indicates poor clusters. The number of clusters with the highest Silhouette Score is selected as the optimal number of clusters.

In Fig. 4, the calculated Silhouette Scores for different numbers of clusters, ranging from 10 to 50, are displayed. This chart helps us select the optimal number of clusters, which corresponds to the highest Silhouette Score.

Once the clusters are identified, each cluster is examined separately. For each cluster, a DeBERTa model is fine-tuned to learn the appropriate labels for the questions in that cluster effectively. This model has been improved by introducing disentangled attention and an enhanced masked decoder. In the disentangled attention mechanism, each word is indexed with two vectors, one for its content and another for its position in the sentence. This separation allows the model to pay more effective attention to both the semantic and positional relationships of words. DeBERTa's enhanced masked decoder also considers the absolute positions of words in the masked word prediction process, which helps the model better distinguish differences in sentences with similar words. DeBERTa outperforms BERT and RoBERTa in various NLP tasks and has achieved significant results, especially in text identification and classification. In this stage, the questions and their tags are fed into the DeBERTa model, which is specifically trained for that cluster.

This approach offers several key advantages. First, models are trained on smaller groups of similar questions instead of the whole dataset, which can have very different meanings. This makes the models more accurate. Second, by splitting the data into different groups and training separate models, we can create smaller and better models. This reduces the time it takes to make predictions and makes the system work more efficiently.

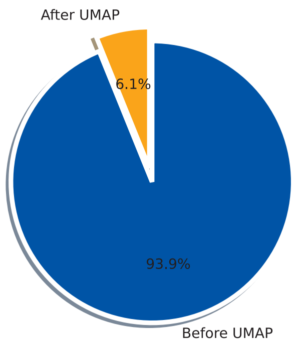

Fig. 3. Proportion of Dimensions Before and After UMAP

IV. EXPERIMENTAL RESULTS

This section details the results of our experiments in benchmarking the effectiveness of the proposed method. We compared our model against baseline approaches using various metrics that emphasize efficiency improvements. These results will help confirm that our method can capture the subtlety arising in multi-label classification tasks while question tagging and, therefore, find wider applications. To ensure our model is strong during training and testing, we removed tags that appeared less than 100 times in the data. We then evaluated our model's performance using Precision, Recall, and F1-Score.

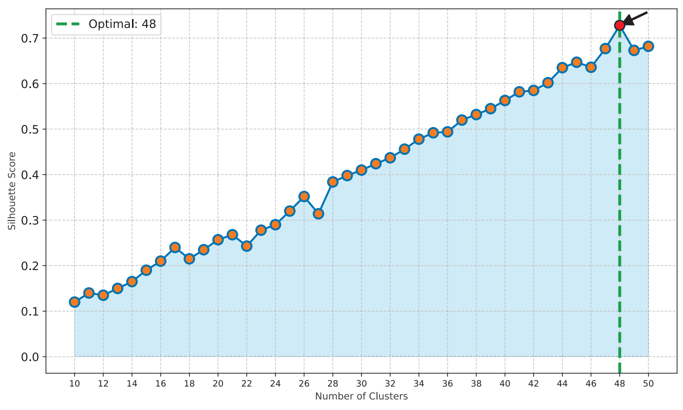

Fig. 4. Silhouette Scores Across Cluster Counts (10-50)

Our dataset includes questions from Stack Overflow, with each question having three main parts: the title, the question itself, and any code that goes with it. These parts are important for our analysis.

In the evaluation phase, our approach predicts up to five tags per question. For this evaluation to be fair, it is designed to suit the number of tags that each question may actually have. For instance, if a question has only three associated tags, we evaluate only the top three predicted tags from our approach. The results of these evaluations are presented in Table I.

TABLE I. EVALUATION RESULTS OF ALL TAGS

Approach	F1_score	Recall	Precision
Erjan et al. 2023	0.541	0.643	0.467
Xu et al. 2021	0.625	0.740	0.541
Chehreh et al. 2024	0.659	0.766	0.578
Our Approach	**0.679**	**0.788**	**0.597**

In this study, the impact of different training data sizes on model quality was also examined. The results indicate that increasing the training data size has a direct effect on improving model performance, but the extent of this impact varies across different stages. For instance, when the data volume increases from very low levels (such as 1/16 and 1/8) to larger amounts, a significant improvement in model quality is observed. This suggests that at these early stages, the model is highly dependent on having more data, and adding more data rapidly enhances its performance.

However, at higher levels, such as when the data volume increases from 3/4 to the full dataset, the changes in model quality gradually diminish. In other words, at these stages, the model already has sufficient data for learning, and adding more data has a lesser impact on improving its performance.

Thus, it can be concluded that during the early stages of increasing data, the changes in model quality are more pronounced. However, as the model approaches using the full dataset, these changes gradually decrease, and the additional data contributes less to improving the model's performance. This finding highlights the importance of having sufficient data during the early stages of training, while the efficiency of adding more data decreases in the later stages.

The results of this evaluation are illustrated in Fig. 5, which clearly show the changes in model quality based on the size of the training data.

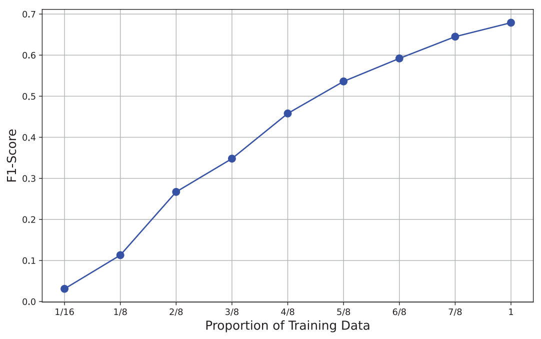

Fig. 5. Impact of Training Data Size on Model Quality (F1-Score)

As part of our experiments, we checked how dimensionality reduction affects our results, so we evaluated our method both without and with this dimensionality reduction. The findings showed that using dimensionality reduction made the system work much faster. Additionally, as shown in Table II, this method not only cuts down on processing time but also boosts the performance of our model.

TABLE II. COMPARISON OF MODEL PERFORMANCE WITH AND WITHOUT DIMENSIONALITY REDUCTION

Approach	F1_score	Recall	Precision
Without dimensionality reduction	0.665	0.773	0.583
With dimensionality reduction	**0.679**	**0.788**	**0.597**

V. CONCLUSION

This paper presents a novel and robust approach to the automatic multi-label classification of questions, specifically tailored for the dynamic and diverse environment of Stack Overflow. The complexity of accurately tagging questions on such a platform, where topics can span a wide range of domains, poses significant challenges that traditional methods have struggled to address. By implementing a two-stage strategy that first clusters questions based on semantic similarities and then fine-tunes dedicated models for each cluster, we have made substantial progress in enhancing both the precision and effectiveness of the tagging process.

The integration of advanced NLP models, such as SMPNet for generating meaningful semantic embeddings and DeBERTa for fine-tuned classification, has proven to be particularly advantageous. When combined with dimensionality reduction techniques like UMAP, these models facilitate a more manageable and accurate clustering process. Additionally, employing the Silhouette Score to determine the optimal number of clusters further enhances the reliability of our approach, ensuring that each model is trained on a well-defined and homogeneous subset of data.

Our experimental results demonstrate a significant improvement over baseline methods, both in terms of F1-score and in generalizing across various types of questions. The approach is scalable and can be adapted to other large-scale, content-rich platforms where effective content organization is at a premium.

REFERENCES

[1] V. Ithipathachai and M. Azizi, "Are tags' it?' analysis of the impact of tags on stackoverflow questions," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 2022, pp. 1483–1490.
[2] *Stack Overflow Information*, *(Example for web-links)*. [Online]. Available: https://usesignhouse.com/blog/stack-overflow-stats/#:~:text=As%20of%20November%202022%2C%20there%20are%20more%20than%20100%20million,and%2023%20million%20registered%20users.
[3] *Stack Overflow help center*, *(Example for web-links)*. [Online]. Available: https://stackoverflow.com/help/tagging
[4] L. Nie, Y. Li, F. Feng, X. Song, M. Wang, and Y. Wang, "Large-scale question tagging via joint question-topic embedding learning," *ACM Transactions on Information Systems (TOIS)*, vol. 38, no. 2, pp. 1–23, 2020.
[5] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," *arXiv preprint arXiv:2006.03654*, 2020.
[6] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
[7] Y. Liu, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
[8] A. K. Saha, R. K. Saha, and K. A. Schneider, "A discriminative model approach for suggesting tags automatically for stack overflow questions," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 73–76.
[9] X. Xia, D. Lo, X. Wang, and B. Zhou, "Tag recommendation in software information sites," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 287–296.
[10] L. Short, C. Wong, and D. Zeng, "Tag recommendations in stackoverflow," *San Francisco: Stanford University*, 2014.
[11] T. Saini and S. Tripathi, "Predicting tags for stack overflow questions using different classifiers," in *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*. IEEE, 2018, pp. 1–5.
[12] K. Alrashedy, D. Dharmaretnam, D. M. German, V. Srinivasan, and T. A. Gulliver, "Scc++: Predicting the programming language of questions and snippets of stack overflow," *Journal of Systems and Software*, vol. 162, p. 110505, 2020.
[13] V. Jain and J. Lodhavia, "Automatic question tagging using k-nearest neighbors and random forest," in *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*. IEEE, 2020, pp. 1–4.
[14] N. Khezrian, J. Habibi, and I. Annamoradnejad, "Tag recommendation for online q&a communities based on bert pre-training technique," *arXiv preprint arXiv:2010.04971*, 2020.
[15] B. Xu, T. Hoang, A. Sharma, C. Yang, X. Xia, and D. Lo, "Post2vec: Learning distributed representations of stack overflow posts," *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3423–3441, 2021.
[16] P. Devine and K. Blincoe, "Unsupervised extreme multi label classification of stack overflow posts," in *Proceedings of the 1st International Workshop on Natural Language-based Software Engineering*, 2022, pp. 1–8.

[17] J. He, B. Xu, Z. Yang, D. Han, C. Yang, and D. Lo, "Ptm4tag: sharpening tag recommendation of stack overflow posts with pre-trained models," in *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, 2022, pp. 1–11.

[18] E. Skenderi, S.-M. Laaksonen, J. Huhtamäki, and K. Stefanidis, "Assessing text representation methods on tag prediction task for stackoverflow," in *The 56th Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences, 2023, pp. 585–594.

[19] S. Subramani, S. Rajesh, K. Wankhede, and B. Wukkadada, "Predicting tags of stack overflow questions: A deep learning approach," in *2023 Somaiya International Conference on Technology and Information Management (SICTIM)*. IEEE, 2023, pp. 64–68.

[20] I. Chehreh, E. Ansari, and B. S. Bigham, "Advanced automated tagging for stack overflow: A multi-stage approach using deep learning and nlp techniques," in *2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*. IEEE, 2024, pp. 1–6.

[21] *Dataset link, (kaggle)*. [Online]. Available: www.kaggle.com/code/younday/auto-tagging-stack-overflow-questions/input

[22] *SBert, (kaggle)*. [Online]. Available: https://www.sbert.net/docs/sentence_transformer/pretrained_models.html

[23] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mpnet: Masked and permuted pre-training for language understanding," *Advances in neural information processing systems*, vol. 33, pp. 16 857–16 867, 2020.

[24] A. Vaswani, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[25] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[26] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[27] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[28] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[29] G. Vardakas, I. Papakostas, and A. Likas, "Deep clustering using the soft silhouette score: Towards compact and well-separated clusters," *arXiv preprint arXiv:2402.00608*, 2024.