# WikiTopic: A General Article Classifier

David Colonna
*Department of Computer Science*
*ETH Zürich*
dcolonna12.dc@gmail.com

Kajetan Pyszkowski
*Department of Computer Science*
*ETH Zürich*
kajetan.pyszkowski@gmail.com

Veit Bilingger
*Department of Computer Science*
*ETH Zürich*
veit.billinger@gmail.com

Gion Steggmaan
*Department of Computer Science*
*ETH Zürich*
gion.stegman@gmail.com

***Abstract*—This research explores the use of sophisticated machine learning methods to categorize a dataset of 1,000,000 Wikipedia articles. The primary aim is to design a proficient model that arranges these articles into unified clusters, highlighting hidden correlations in Wikipedia's vast information space. Early efforts using similarity metrics faced computational challenges, prompting a shift to sentence transformers for embedding article content. The integration of the HDBSCAN algorithm effectively determined density-centric clusters. The methodology yielded about 400 distinct clusters and pinpointed roughly 20,000 anomalous categories. Preliminary tests using BERT and ALBERT were not as successful as anticipated. Yet, transitioning to models with larger attention spans yielded promising outcomes, achieving an accuracy of 0.54 and F1 score of 0.80 on our evaluation dataset.**

## 1. Introduction

Wikipedia, with its vast repository of articles, serves as a vital information source for millions globally. Yet, the efficient categorization of these articles into precise topics remains a challenge, primarily due to the vastness and diversity of the content. In practice, machine learning has shown the capability to navigate vast data repositories, leading us to explore these techniques for Wikipedia article classification. In particular, the recent advancements in LLMs have motivated us to explore models with larger window sizes. This paper focuses on evaluation models for the classification of Wikipedia articles into distinct categories using the power of advanced neural network architectures.

To this end, we use the capabilities of BERT (Bidirectional Encoder Representations from Transformers) [1], a state-of-the-art natural language processing model, and its little brother ALBERT [2], known for their contextual understanding of text to serve **as a baseline**. Our experiments leverage an extensive dataset containing around 1 Mio. Wikipedia articles, each annotated with its respective categories. Such a rich dataset, we believe, provides an excellent opportunity to refine and validate our methodologies.

The task of refining the dataset for multilabel classification proved more challenging than expected. Initially, we assumed that extracting and parsing the entire Wikipedia dump would be straightforward. However, we quickly encountered RAM and disk storage constraints, which explains our limitation to 1 million articles. Moreover, post-extraction, we identified a massive 1.2 million unique categories. To create a generalized classification method for Wikipedia articles, it was necessary to drastically reduce these category numbers.

Our early attempts to cluster the categories using traditional matrix-based methods proved to be resource-intensive. As a result, we shifted our approach: we moved the categories to an embedding space, reduced their dimensionality, and then clustered them. This approach identified 420 categories, which, upon manual review, appeared consistent. Regrettably, due to our inability to parse the entire Wikipedia dataset, we encountered a significant number of outliers. We speculate that these outliers notably influenced our concluding outcomes.

Upon testing with BERT [1] and ALBERT [2], **our accuracy plateaued at 0.45**. Given that these models are restricted by an attention window of 512 tokens, we sought models that accommodate larger windows. Notably, the self-attention mechanism in conventional transformers scales with the sequence length in a quadratic manner. This led us to the Longformer, designed expressly to tackle this limitation. As anticipated, utilizing the Longformer demonstrated that expanding the window size correspondingly improved accuracy, **up to 0.54**.

## 2. Literature Review

Recent studies have delved into diverse techniques for classifying Wikipedia articles. Notably, Wu, Zongda, et al. presented an innovative method for Wikipedia semantic matching in their work, *"An Efficient Wikipedia Semantic Matching Approach to Text Document Classification"* [3]. This method leverages heuristic selection rules to rapidly pinpoint pertinent concepts from Wikipedia's semantic realm, removing the requirement to traverse the entire space. Furthermore, it utilizes the semantic representations of textual documents to gauge similarities, facilitating accurate classification.

Additionally, the domain of Natural Language Processing (NLP) has seen the emergence of robust models, chief among them being BERT, which revolutionized NLP tasks with its bidirectional attention mechanism and pre-training on large corpora, allowing for fine-tuning various downstream tasks with minimal task-specific parameters. The model's ability to capture context from both directions and its subsequent generalization capabilities rendered it a state-of-the-art solution across several benchmarks. However, with its robustness came increased computational and memory demands. To address this, ALBERT (A Lite BERT) was proposed by Lan et al. (2019). ALBERT reconfigures the architecture of BERT to reduce parameter count and increase training speed, achieving this by factorizing the embedding layer and sharing parameters across the hidden layers. While maintaining or even surpassing the performance of its predecessor, ALBERT showcased the potential of achieving state-of-the-art results with optimized and efficient architectures. Both models serve as foundational pillars in the ongoing evolution of transformer-based NLP methodologies.

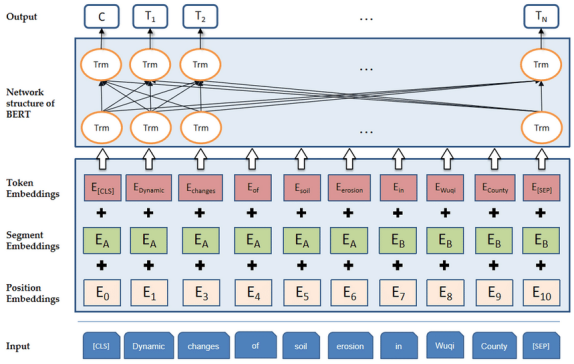


Figure 1: BERT's fundamental architecture.

Subsequent to the advancements brought about by BERT and ALBERT, the NLP community directed its attention to the inherent limitations of transformers concerning long text sequences. The quadratic scaling of self-attention mechanisms with sequence length restricted the application of models like BERT to relatively short text segments. Addressing this challenge, Beltagy et al. (2020) introduced Longformer [4], a novel transformer architecture designed to handle extended sequences efficiently. The primary innovation of Longformer lies in its combination of local windowed attention with global attention, effectively reducing the time complexity from quadratic to linear. This enables the processing of document lengths that were previously infeasible for traditional transformer models. In addition to its architectural innovations, Longformer demonstrated competitive performance across a range of tasks, proving that it's possible to manage longer sequences without compromising on the quality of outcomes.

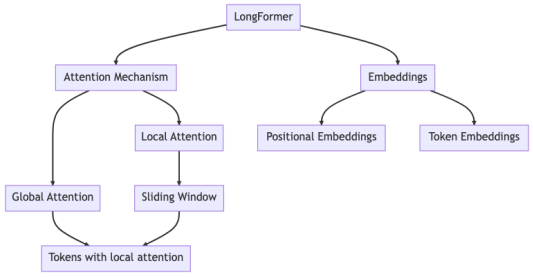


Figure 2: LongFormer Structure

## 3. Data

Our initial dataset is derived from the May 2023 Wikipedia dump files, encompassing the entire corpus of 6,775,344 English articles (Wikimedia Foundation, 2023). Enclosed within a compressed 21 GB archive, this repository exclusively comprises textual content, omitting images or multimedia elements, as well as links to other articles.

### 3.1 Data Extraction

In our data extraction process, we employ the *parserfromhell* Python package [5], which allows for extracting pertinent information from the dump files. For each article, this package parses essential details such as the title, textual content, and associated categories. These categories serve as the backbone of topical classifications on Wikipedia, providing the semantic labels which are crucial for the training of our classification models.

Nonetheless, the straightforward application of parserfromhell encountered computational bottlenecks. It operates sequentially, demanding over 10 minutes for every 10,000 articles. The tool's memory usage also poses a problem, necessitating an impractical 512GB of RAM to parse the complete dump.

To surmount these challenges, we harnessed the power of parallelization using Google Cloud virtual machines. Fragmenting the data across multiple instances enabled us to significantly reduce the time and memory required for generating an adequately sized parsed subset. In the end, our efforts culminated in the successful parsing of more than 1 million

articles, accomplished over a span of 17 hours and at a cloud computing cost of **$175**.

The resultant dataset encompasses 1,039,224 articles associated with 1,211,556 distinct categories. With an average article length of 1185 words, the dataset comprises substantial textual material ideal for subsequent modelling endeavours. However, the extensive vocabulary of categories introduces complexities in the realm of traditional classification techniques. In light of this challenge, our experimentation pivots towards the effective clustering of categories, laying the groundwork for a scalable approach to topic modelling.

## 3.2 Data Curation

Due to the unwieldy size of the dataset, we removed all articles that served as redirects. We then delved deeper into curation using embedding spaces and clustering techniques (see **Section 4.1**). Our category clustering efforts led us to approximately 400 clusters, each encapsulating articles with shared thematic similarities. It's worth noting that our approach also identified around 20,000 categories as outliers, highlighting the presence of diverse and less common topics within the dataset. These outliers, while posing additional modelling challenges, underscore the richness and diversity of content within Wikipedia's vast repository.

# 4. Methods and Experiments

## 4.1 Category Clustering

### 4.1.1 Initial Efforts

In our quest to effectively cluster an extensive dataset of 1,211,556 distinct categories from Wikipedia, we started of in various directions. Initially, we attempted to employ similarity metrics, such as Jaccard distance and Levenshtein distance, to cluster category terms based on their titles. However, the computational demands of hierarchical agglomerative methods proved to be a challenge, requiring over 24 hours to process just 5000 categories.

### 4.1.2 Semantic Vector Representations

Recognizing the need for a more efficient and effective approach, we started using sentence transformers trained on Wikipedia data (Reimers & Gurevych, 2019) [6] to embed category titles. This shift marked a turning point in our methodology, significantly enhancing both the efficiency and quality of results. By inputting the vast array of 1,211k categories into the BERT-based model, we harnessed the power of semantic vector representations. To streamline the process, we employed UMAP to reduce the embeddings' dimensionality to 100, further optimizing the approach.

### 4.1.3 HDBSCAN

At the heart of our success lies the HDBSCAN algorithm. HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) is an advanced clustering algorithm that extends the principles of the classic DBSCAN (Density-Based Spatial Clustering of Applications with Noise) method. Unlike traditional clustering techniques that assume clusters to be of similar sizes or shapes, HDBSCAN identifies clusters of varying densities, making it particularly good at detecting clusters that other algorithms might miss. The "hierarchical" aspect of HDBSCAN allows it to examine clusters on a spectrum of scales, ensuring that even nested or overlapping clusters are identified. One of the standout features of HDBSCAN is its ability to discern noise or data points that don't belong to any cluster, which can be crucial for many real-world applications.

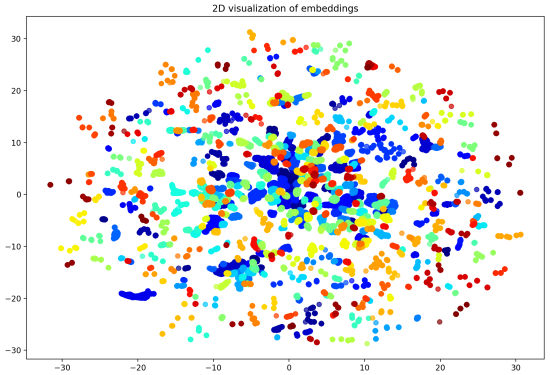


Figure 3: UMAP cluster embeddings visualization w/o the noise.

After applying these new techniques we started seeing some real progress. By leveraging the semantic embeddings generated through the BERT-based model and harnessing the power of HDBSCAN, we achieved a breakthrough in our clustering efforts. We succeeded in categorizing the extensive set of 1,211,556 categories into approximately 400 coherent clusters. Additionally, the methodology identified around 20,000 outlier categories.

### 4.1.4 Outlier Categories

Our hypothesis about why we have so many outliers is linked with our early issues of parsing the whole Wikipedia dataset. As we spent much time trying to parse the articles more efficiently, we finally settled on only parsing around 10% of the dataset. However, due to the way the Wikipedia dump is structured and how our parsing method works, we could not choose the subset of articles we wanted to parse. It is possible that this created a bias in the selected articles, which could

explain the presence of so many categories with only one corresponding article, hence the outliers.

Another hypothesis explaining the huge number of outlier categories could be the structuring system of Wikipedia. We could not find any conclusive information in this regard, but it is possible that even taking in its globality, there are more categories than articles on Wikipedia.

In the end, we chose not to take into account the outliers as they would just be considered noise in our subsequent classification task and would potentially confuse our classification model. This is definitely a tradeoff as we lose a certain percentage of our classification labels, but we decided it was worth it as we would save a lot of time.

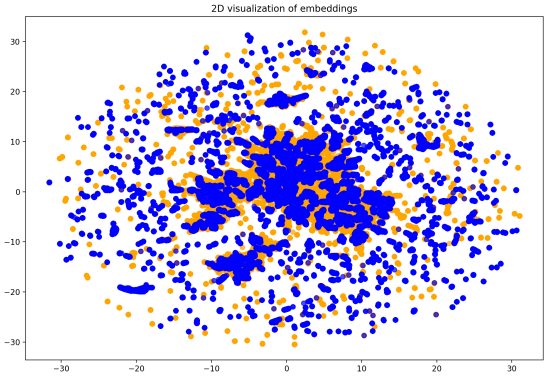


Figure 4: UMAP cluster vs noise visualisation.

## 4.2 Model Fine-Tuning

First, in order to make our dataset work with any model we had to build a custom dataset. Our implementation takes the **clustered** Wikipedia categories and uses them to generate training samples for the different models by encoding the categories into label vectors. The text of each article is tokenized (with the corresponding model's tokenizer) to create the model input.

### 4.2.1 BERT

Once we obtained the resulting clusters from HDBSCAN we decided first to try and use the pre-trained BERT model and fine-tune it in a downstream task.

BERT (Bidirectional Encoder Representations from Transformers) is a technique in natural language processing created by Google in 2018. Based on the Transformer architecture, BERT utilizes attention mechanisms to learn deep contextual representations from unlabeled texts in a bidirectional manner. During pretraining, BERT develops versatile language understanding through two unsupervised tasks - Masked Language Modeling and Next Sentence Prediction. This gives BERT an advantage over previous models limited to unidirectional context. Built with transfer learning in mind, pre-trained BERT models can be fine-tuned on downstream NLP tasks through the simple addition of task-specific layers, achieving good results. The pre-trained models come in two sizes - BERT Base and BERT Large, with 110M and 340M parameters respectively.

During our efforts to fine-tune BERT for multi-label classification of Wikipedia articles, we faced considerable challenges regarding long training times, insufficient accuracy, and high resource requirements. Given the enormous size of the dataset, consisting of over 1 million articles, training the BERT base model took an impractically long time even on high-powered GPU systems. Despite training for multiple days, we could only achieve an **accuracy of around 40.06%** on the validation set. The model's perplexity clearly indicated difficulty in converging on the complex dataset.

### 4.2.2 ALBERT

Given these challenges with BERT, we transitioned to using the ALBERT model, a successor to BERT. ALBERT, which stands for *A Lite BERT*, maintains the same powerful contextual embeddings as BERT but with crucial modifications to reduce parameter count and enhance training speed. By factorizing the embedding layer and sharing parameters across all the layers, ALBERT's footprint is significantly lighter, which directly addressed our concerns about prolonged training times and excessive resource demands. ALBERT's architecture naturally resists overfitting, especially on big datasets, resulting in more consistent and reliable performance. With this new method, we achieved a significantly faster training time of around 15 hours instead of 75 hours. Unfortunately, we only managed to **attain 45% accuracy** on our data.

### 4.2.3 LongFormer

After these two tries, we decided to switch our model to the LongFormer in order to obtain better results. The Longformer is made for processing lengthy documents, addressing the sequence length limitation faced by models like BERT, which caps at 512 tokens. What sets Longformer apart is its unique attention mechanism. Instead of the traditional self-attention where every token considers all others—resulting in quadratic computational complexity—Longformer judiciously combines local and global attention. Most tokens in Longformer focus only on a limited window of adjacent tokens, but select tokens can attend to the entirety of the document, capturing broader contexts. Initialized with the knowledge-rich base of RoBERTa, Longformer mixes the strengths of pre-trained models with the capability to handle extensive sequences.

The experimentation with the LongFormer confirmed our initial assumption that increasing the window size would increase the model's overall accuracy. (see Table 1)

| Window Size | Initial Accuracy | Max. Accuracy |
|---|---|---|
| 512 | 0,367 | 0.412 |
| 1024 | 0.409 | 0.443 |
| 2048 | 0.433 | 0.483 |
| 4096 | 0.485 | **0.548** |

Table 1: Table of accuracies for different window sizes.

#### 4.2.4 Loss Function

We decided to use the `BCEWithLogitsLoss` loss function provided by the PyTorch framework, designed for binary and multi-label classification tasks. Its exact formula can be described as follows:

$$L(x, y) = -\frac{1}{N}\sum_{\{i=1\}}^{N}[y_i * \log(x_i) + (1 - y_i) * \log(1 - x_i)] \quad (1)$$

Where:

- `L` is the loss function
- `N` is the batch size
- `x_i` is the predicted probability for class `i`
- `y_i` is the true binary label for class `i`

The PyTorch implementation combines the Binary Cross Entropy (BCE) loss with a sigmoid activation on raw logits, offering a more numerically stable approach than applying the sigmoid activation and BCE loss separately. By integrating the sigmoid operation within the loss computation, `BCEWithLogitsLoss` can prevent potential issues related to precision limitations, which can manifest when using separate operations. We opted for this loss function as it is well-known for ensuring more reliable training, reduces the likelihood of encountering "NaN" or "inf" values during optimization, and often leads to faster convergence.

#### 4.2.5 Optimizing Training

In our efforts to optimize training speed and efficiency, we turned to FP16 precision, which allowed us to increase our effective batch size, making the training process quicker. This was a promising adjustment, providing noticeable improvements. However, when we tried to push the boundaries even further by implementing INT8 precision, we faced challenges due to library incompatibilities. Despite our eagerness to utilize INT8, we still wanted to enhance our batch size and overall efficiency further, these technical issues hindered progress. It's unfortunate as leveraging INT8 we could've increased our effective batch size and potentially yielded better results.

Additionally, we opted for the Adam Optimizer for its adeptness in model optimization. It's a blend of RMSprop's and AdaGrad's best features, resulting in an algorithm that modifies the learning rate for each parameter on the go. Adam's self-adjusting learning rate feature ensures it quickly converges to optimal results. To further optimize our memory footprint, we transitioned our optimizer to Adafactor, trading off a bit of convergence speed for the ability to handle larger batch sizes. Its factored second-moment estimation allowed fitting batch sizes 2x larger in the same memory footprint. This significantly sped up iteration.

We also incorporated gradient accumulation strategies to further increase our batch size. Ultimately, we managed to achieve an effective batch size of 44 for ALBERT and 16 for LongFormer, all within a 16 GB virtual memory limitation during training.

## 5. Results

### 5.1 Preprocessing

Our initial experiments revealed the challenges in preprocessing and modelling the large-scale Wikipedia dataset. While simpler similarity metrics struggled with dimensionality, neural embeddings provided efficient semantic representations. HDBSCAN clustering produced meaningful topics and will facilitate the next phase of article classification. The scale of the category vocabulary was the biggest surprise. The 1 million articles mapped to over 600,000 unique categories - far more than anticipated. This highlighted the importance of scalable embedding and clustering techniques before tackling article classification.

Though we have not yet classified the whole dataset, the category modelling provides a strong proof of concept. Our pipeline represents a novel application of state-of-the-art NLP methods to Wikipedia's breadth of semantic topics. Extending the approach to model article text directly presents opportunities for further optimization and innovation.

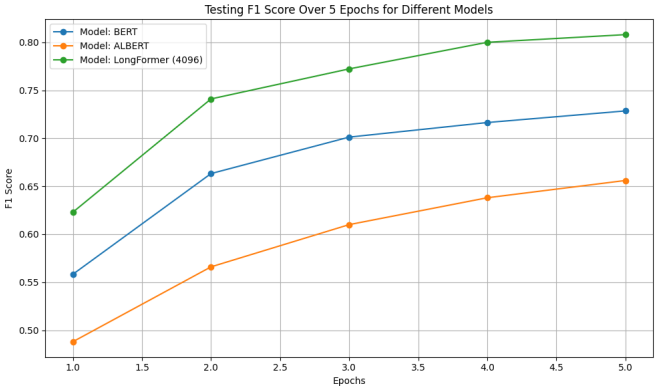


Figure 5: F1 scores of BERT, ALBERT and LongFormer (4096)

### 5.2 Model Training

When attempting to use the BERT model, we encountered unexpected challenges. Despite its reputation for state-of-the-art performance, we achieved a very low accuracy of 0.41. Remarkably, this assessment took an extensive period of more than 30 hours to complete due to the substantial number of samples. The issues with accuracy could likely be attributed to

the sheer magnitude of the dataset, which might have strained the model's capacity to comprehend intricate relationships and patterns.

| Model | Max F1 Score | Max. Accuracy |
|---|---|---|
| BERT | 0.710 | 0.441 |
| ALBERT | 0.697 | 0.393 |
| LongFormer (4096) | **0.802** | **0.548** |

Table 2: Summary of the results over five epochs.

After transitioning from BERT to the ALBERT model, we observed a significant reduction in training time from 75 hours to approximately 15 hours. However, the accuracy achieved was only 39.93% (Table 2).

Our significant breakthrough came with the extended window sizes of LongFormer. Using the 4096-token window, we achieved an accuracy of 0.54, substantially surpassing our earlier benchmarks. For the 512-token window size, the results were comparable to those obtained with ALBERT and BERT. Interestingly, we observed a steady rise in accuracy, which appeared to be linearly correlated with the window size.

Our most notable advancement was observed in the F1 scores with the utilization of LongFormer's extended window sizes. With the 4096-token window, we achieved a notable F1 score of 0.802 (Table 2), exceeding our previous benchmarks by some margin. It's crucial to highlight that the elevated F1 score can be attributed to the presence of at least one unbalanced class in our dataset, specifically the outlier categories. In our evaluations, the evolution of the F1 scores for both ALBERT and BERT remained consistently close (Figure 6). This consistency underscores the robustness of ALBERT and BERT in handling our dataset, but it also highlights the potential ceiling we might have reached with these architectures, especially when compared to the advancements we observed with LongFormer.

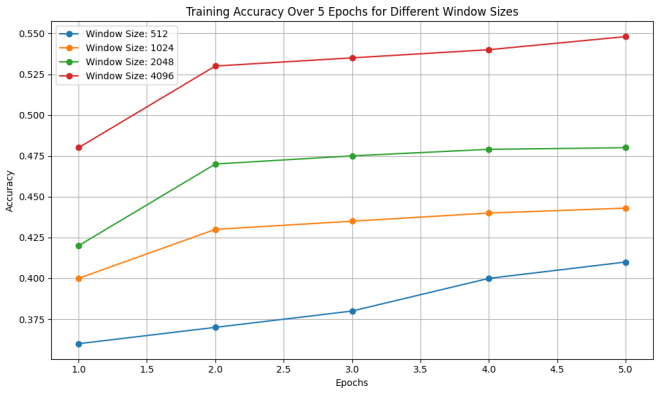


Figure 6: LongFormer accuracy over five epochs for different window sizes.

### 5.3 Visualization Insights

In order to better understand and interpret the high-dimensional embeddings and the resultant clusters, we incorporated various visualization techniques into our pipeline. During our initial efforts while looking into hierarchical clustering methods, we visualized the dendrogram of categories. However, we moved away from this visualization method due to computational constraints and practicality with big datasets. The dendrograms were valuable in understanding the tree-like structure of category relationships, showcasing granular subtopics under broader themes. After employing HDBSCAN, we plotted a histogram of cluster sizes (see Figure 4). This allowed us to identify how many categories typically fall into a cluster, and detect any anomalies such as overly large or minuscule clusters. All in all, through these visualization techniques, several insights emerged. For instance, we identified potential outlier categories that didn't fit neatly into any cluster and might represent niche or unique topics on Wikipedia.

## 6. Future Work

Here are a few of the ways our work could be expanded in the future.

### 6.1 Fine-Tuning on Advanced Models

The realm of NLP has seen an explosion of advanced models following the advent of Transformer architectures. While the experimentations with ALBERT and LongFormer provided valuable insights, there are other models, like Transformer-XL [7] or various iterations of GPT, that could be explored. Transformer-XL, for instance, has been designed with an extended context in mind, allowing it to remember longer sequences of data. This could be crucial when working with extensive Wikipedia articles that may span over the traditional token limits. On the other hand, GPT variants, known for their generative capabilities, might offer unique embeddings that could enhance classification accuracy.

### 6.2 Enhanced Preprocessing

Our initial efforts in preprocessing the Wikipedia dump indicated potential biases in the selection of articles, leading to a massive number of outliers. An enhanced preprocessing stage can help rectify this. One avenue to consider is diving deeper into the structural and semantic nuances of the Wikipedia categorization system. For instance, instead of a rudimentary parsing based on article categorization, a semantic similarity measure could be employed. Techniques like cosine similarity based on TF-IDF vectors or even more advanced embeddings could assist in clustering articles with similar content themes. This approach could bridge the gap where articles might be miscategorized or where categories are too sparse. Such a refined preprocessing technique can ensure a more balanced and

representative dataset, reducing noise and increasing the potential for model accuracy.

### 6.3 Model Ensembling

Finally, Model ensembling [8] is a potent technique that harnesses the power of multiple models to achieve better predictive performance. The underlying philosophy is that while a single model might have its inherent biases and limitations, a collective decision-making process across multiple models can compensate for individual weaknesses. For the Wikipedia corpus, predictions from models like BERT, ALBERT, LongFormer, and even others like Transformer-XL or GPT could be combined. Various ensembling techniques, like weighted averages, stacking, or majority voting, can be employed. It's also worth noting that each of these models may capture different nuances of the data due to their unique architectures and pretraining processes. Combining their strengths might result in a more robust classification system, effectively aggregating diverse perspectives on the data to achieve superior results.

## 7. Conclusion

Through the course of this extensive project, we gained first-hand experience in applying state-of-the-art natural language processing techniques to categorize Wikipedia's massive and diverse corpus of articles. Our experimentation with models like BERT [1], ALBERT [2], and LongFormer [4] illuminated the unique challenges associated with large-scale multi-label text classification across a heterogeneous textual dataset. Despite leveraging powerful contextual embedding approaches, the models struggled to achieve robust performance in terms of accuracy and training time efficiency over this voluminous data.

These outcomes underscored the importance of meticulous dataset preparation and preprocessing to improve the learnability of the models. As evidenced by the significant outlier categories, our initial parsing methodology had limitations that likely introduced dataset biases. Exploring alternative parsing and sampling techniques tailored to Wikipedia's structure and size could yield a more balanced training corpus.

Meanwhile, model architectures that handle lengthy text sequences better, like Transformer-XL [7], may be better suited for classifying encyclopedic articles. Techniques like gradient accumulation also proved vital in allowing large batch training with memory constraints. There remains substantial scope for innovation to enhance model convergence and scalability over corpora of this scale.

Moving forward, employing an ensemble approach by combining diverse models could potentially improve accuracy by aggregating their complementary strengths. Though substantial challenges persist, this project provided invaluable learnings regarding the real-world application of machine learning to knowledge management tasks. With continued methodical experimentation and innovation, significant progress can be made in effectively organizing and categorizing expansive text repositories, like Wikipedia.

### References


[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: pre-training of deep bidirectional transformers for language understanding," 2019.

[2] Z. Lan, M. Chen, et al., "Albert: a lite bert for self-supervised learning of language representations," 2020.

[3] Z. Wu, H. Zhu, et al., "An efficient wikipedia semantic matching approach to text document classification," *Inf. Sciences*, vol. 393, pp. 15–28, 2017, doi: https://doi.org/10.1016/j.ins.2017.02.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025517304292

[4] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: the long-document transformer," 2020.

[5] G. Contributors, "Mwparserfromhell," GitHub, 2023. (\url{https://github.com/earwig/mwparserfromhell})

[6] N. Reimers, and I. Gurevych, "Sentence-bert: sentence embeddings using siamese bert-networks," 2019.

[7] Z. Dai, Z. Yang, et al., "Transformer-xl: attentive language models beyond a fixed-length context," 2019.

[8] M. Alhamid, "Ensemble models, a guide to learning ensemble techniques in a simple walkthrough." [Online]. Available: https://towardsdatascience.com/ensemble-models-5a62d4f4cb0c