Comparative Study of Classical and Neural Retrieval Methods for FANDOM Wikis

Aaron Kutzer and **Florian M. Müller** and **Till Dirschnabel** and
Tommy Nguyen and **Zhìwēi Zhān** and **Anja Reusch** and **Julius Gonsior**
Dresden University of Technology
Dresden, Germany

Abstract

This paper investigates the effectiveness of traditional information retrieval methods and the neural retrieval method ColBERT in the context of locating relevant passages within fandom wikis. To facilitate this study, we developed a new dataset called FANDOM-QA, consisting of query and answering passage pairs. Additionally, we provide a novel analysis of the operational mechanisms employed by ColBERT, shedding light on its functioning and capabilities.

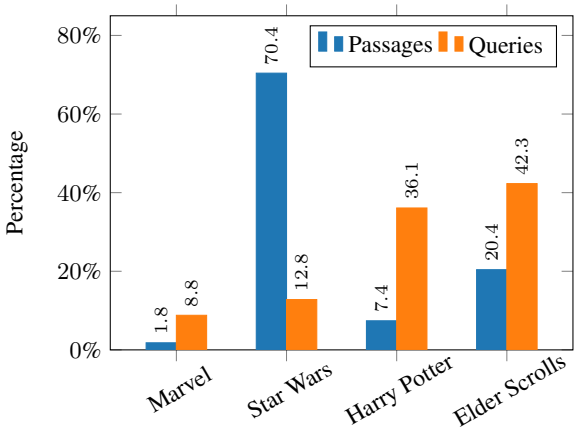


Figure 1: Percentage of Passages and Queries of each Wiki in the FANDOM-QA dataset

1 Introduction

Natural language processing has developed at an astonishing rate in recent years. An important task of natural language processing is information retrieval which consists of finding relevant information to answer a given questions. Classical retrieval methods do not have semantic understanding of language (Hambarde and Proenca, 2023), severely limiting the information retrieval, this changed with neural retrieval.

Our research consists of implementing a semantic retrieval system performing extractive question answering on fandom wikis sourced from fandom.com. Both classical and neural retrieval methods were tested on their effectiveness for this task. For this, we first extracted and cleaned publicly available fandom wiki exports and secondly generated our own training and validation datasets, which were used for training and evaluation of the retrieval methods. Furthermore, this paper discusses and evaluates optimisation methods for training datasets and hyperparameters and visualization methods to gain a better understanding of ColBERT. We start by introducing the datasets and their construction (Chapter 2), then we explain our retrieval methods and their training (Chapter 3.2.3). In Chapter 4, we present the results obtained in terms of metrics, hyperparameters and multi-stage fine-tuning. Finally in Chapter 5, we introduce visualisation methods to better understand Colbert and find the answer in the extracted passage, and discuss the observations made.

2 Datasets

For training and validation of the retrieval system, we utilized two datasets: the publicly available MS MARCOv2 (Bajaj et al., 2018) dataset and our custom dataset, FANDOM-QA.

The MS MARCO dataset was used in the first stage of multi-stage fine-tuning in our neural retrieval models and to validate our implementation against the original implementation, while the FANDOM-QA dataset was later used in the second stage of multi-stage fine-tuning. In this second stage, either the entire dataset or a subset of the dataset corresponding to a particular wiki were used.

Figure 1 and 14 provide information about the size of the datasets and the distribution of the fandom wikis in the FANDOM-QA dataset.

2.1 FANDOM-QA

The FANDOM-QA dataset was constructed by extracting all passages from the specific fandom

wikis: DC Comics, Elder Scrolls, Harry Potter, Marvel, Star Wars, and The Witcher using the the assistance of the *wikiextractor* tool (Attardi, 2015). Next, we generated positive and negative queries for all passages. Positive questions are designed to have answers that can be found in the passage, while negative questions cannot be answered using the information in the passage. To expedite this process, we utilized publicly available GPT-frontends using the *gpt4free*[1] repository.

To ensure effective evaluation of our neural retrieval method, we partitioned the dataset into two subsets: training and validation. The training subset constituted 89% of the data, while the validation contained 11%.

It is important to note that these steps were performed separately for each wiki, and the resulting datasets were then merged to create the final FANDOM-QA dataset. This ensured that our models could be trained and evaluated on specific fandom wikis or all collected wikis, while preventing information leakage between the training and evaluation datasets. We also wanted to evaluate our model on a wiki on which the model had not been trained. We used the Witcher dataset for this, and consequently did not include it in the Fandom-QA dataset. A few random samples of the dataset can be found in Figure 13.

We observed some problems with query generation using GPT. One obstacle was observed during training in the second stage of the multi-stage fine-tuning. A high proportion of the words in the negative query also appeared in the passage. To our surprise, this led to a drastic decrease in the performance of our model, possibly due to the loss function we used. To overcome this obstacle, we shuffled the negative queries in the FANDOM-QA training dataset to obtain a dataset where the positive and negative queries for a passage are more diverse.

3 Retrieval Methods

3.1 Classical Retrieval Methods

We used Term Frequency - Inverse Document Frequency (TF-IDF), a classic retrieval method, as a baseline model for comparison and for the reranking retrieval method described in Chapter 3.2.3. TF-IDF quantifies the importance of terms in a collection of documents by determining the frequency of a term (TF) within a document and adjusting it based on the inverse document frequency (IDF) across the collection. Thus, terms that occur more frequently within a particular document, but are relatively rare across the collection, are given high weights. By representing the query and the passages as vectors containing a TF-IDF value for each term that occurs in the entire corpus, the passages can be ranked using cosine similarity. However, TF-IDF has significant limitations. It assumes that the importance of a term depends solely on its frequency and treats each term as an independent entity, ignoring contextual factors such as the order or proximity of terms within a document, as well as synonyms. As a result, TF-IDF cannot capture semantic meaning, a problem that neural approaches seek to overcome.

3.2 Neural Retrieval Methods

For our neural retrieval approach, we adopted the ColBERT (Khattab and Zaharia, 2020) model, which utilizes Transformer models to generate vector embeddings for word tokens in a text sequence. These embeddings capture the contextual information of the tokens within the sequence and are then used to calculate the similarity between two sequences.

3.2.1 ColBERT Model

The ColBERT model takes a query and a document passage as input. These strings are then tokenized using the Transformer model's associated tokenizer, resulting in sequences of query tokens ($q = q_1 q_2 \dots q_m$) and document tokens ($d = d_1 d_2 \dots d_n$). Next we prepend a `[CLS]` token at the beginning and append a `[SEP]` token at the end of both sequences and add either a `[Q]` or `[D]` token after the `[CLS]` token to encode the input sequence type (query or document/passage). The query and document sequences are constrained to maximum lengths of N_q and N_d tokens, respectively. If the query is shorter than N_q tokens, we pad the sequence with `[MASK]` tokens until it reaches length N_q. The document sequence will not be padded if shorter than N_d and will be of length $L_d = min(n + 3, N_d)$. Thus, the input sequences for the ColBERT model are as follows: $q =$ `[CLS][Q]`$q_1 q_2 \dots q_m$`[SEP][MASK]...[MASK]` and $d =$ `[CLS][D]`$d_1 d_2 \dots d_{L_d-3}$`[SEP]`, respectively. These tokenized sequences are then passed through a Transformer model's Encoder, with experiments conducted using both BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019)

[1] https://github.com/xtekky/gpt4free

architectures. The resulting output is a sequence $E = E_1 E_2 \dots E_k$ of high-dimensional vectors, where k corresponds to N_q or L_d, depending on the sequence being processed. These vectors are subsequently mapped to a lower dimensionality d using a linear transformation. To calculate the similarity between the query and document sequences, we employ the *sum of maximum similarity* function, as presented by the authors of ColBERT. However, instead of using the sum, we use the mean to obtain a more interpretable similarity score. The function is computed as follows:

$$S(q,d) := \frac{1}{N_q} \sum_{i=1}^{N_q} \max_{j=1,\dots,L_d} sim(E_{q_i}, E_{d_j})$$

We evaluate similarity between embedding vectors using both cosine similarity and negated squared L_2-norm. Originally, the vectors were normalized before applying these similarity measures. However, we also experimented with L_2-norm without normalization of the embedding vectors. The formulas for the similarity measures can be found in the Appendix D.

3.2.2 ColBERT Training

During the training the BERT/RoBERTa encoder is fine-tuned and the last linear projection and the additional `[Q]`/`[D]` tokens are learned from scratch. For training on the MS MARCO dataset, the model receives batches of tuples in the form $\langle q_i^+, d_i^+, d_{i,1}^-, \dots, d_{i,9}^- \rangle$. Here, d_i^+ represents the answer passage for the query q_i^+, and the passages $d_{i,l}^-$ do not contain the answer. The FANDOM-QA dataset contains negative queries for a passage, rather than providing negative passages for a query. So when training on our FANDOM-QA dataset, the model receives batches of tuples in the form $\langle q_i^+, q_i^-, d_i^+ \rangle$, where q_i^+ is answered by d_i^+ while q_i^- is not. Despite the differences in data format, the training objective remains the same: maximizing the similarity between the answering query/passage pairs relative to the other given passages. The loss is calculated using the cross-entropy loss and is defined as follows for the MS MARCO dataset:

$$\ell = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{e^{N_q S(q_i^+, d_i^+)}}{e^{N_q S(q_i^+, d_i^+)} + \sum_{j=1}^{9} e^{N_q S(q_i^+, d_{i,j}^-)}}$$

For the FANDOM-QA dataset, the loss function becomes:

$$\ell = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{e^{N_q S(q_i^+, d_i^+)}}{e^{N_q S(q_i^+, d_i^+)} + e^{N_q S(q_i^-, d_i^+)}}$$

Sadly, it was observed during our experiments that it is necessary to manually scale the similarity scores by a factor of N_q in order to achieve better convergence during training (so we actually use the sum of maximum similarity for training). The reason behind this requirement is likely attributed to the range of cosine similarity scores, which typically fall between -1 and 1. Even in an ideal scenario where predicted similarities take the form of $\langle 1, -1, \dots, -1 \rangle$, applying the softmax function yields probabilities such as $\langle 0.451, 0.061, \dots, 0.061 \rangle$, which is similar to using extremly aggressive label smoothing (Szegedy et al., 2015). Unfortunately, we have not been successful in devising an alternative loss function to address this limitation.

We conducted experiments using Mean Squared Error (MSE) loss in combination with cosine similarity. In this setup, the target for the loss function was set to 1 for the pairs $\langle q, d^+ \rangle$ representing answering passages. Conversely, for the pairs $\langle q, d^- \rangle$ representing not answering passages, we enforced orthogonality by aiming for a cosine similarity of 0. However, this approach yielded inferior performance, as demonstrated in the subsequent analysis.

3.2.3 Retrieval with ColBERT

Due to the computational cost associated with calculating the similarity between a given query and all passages, we adopt the two methods from the original ColBERT paper: *re-ranking* and *end-to-end retrieval*. In the *re-ranking* approach, we employ a classical retrieval algorithm (in our case TF-IDF) to retrieve the top-k passages for a query. Subsequently, ColBERT is used to re-rank these k documents based on their similarity scores. On the other hand, *end-to-end retrieval* exclusively utilizes ColBERT. For a given query, we search for the top-$\hat{k}$ most similar document embedding vectors for each of the N_q query embedding vectors. In the original work this was done using the vector search framework `faiss`, however due to time limitation we used a simple precomputed document embedding matrix and exhaustively calculated the similarities. We retrieve the associated documents ($\leq \hat{k} N_q$ unique documents) and calculate the similarity between the query and these documents. Finally, we

return the top-k documents. The validation of our implementation with the original implementation can be found in the Appendix A.

4 Results

This chapter showcases the results of our conducted experiments on ColBERT with the aim of (1) assessing the significance of various hyperparameters in relation to the retrieval performance, and (2) evaluating the comparative advantages of training the model using all fandoms versus specific ones, as well as its ability to generalize to unseen wikis. To address task (1), we conducted an extensive experimental phase of hyperparameter optimization. The performance of each model was evaluated using selected metrics (Chapter 4.1). Subsequently, we fine-tuned the model with the best hyperparameter settings on our FANDOM-QA dataset and the Harry Potter dataset (Chapter 4.3).

4.1 Metrics

We placed emphasis on computing Recall@k and MRR@k, which are two commonly used model evaluation metrics that are crucial for assessing the information retrieval capability and ranking accuracy of a model and are also used in the official ColBERT paper, allowing for comparison. In both cases, @k represents the truncation value, beyond which a passage is deemed a false negative or, in the case of MRR, assigned a reciprocal rank of 0.

$$\text{Recall@}k = \frac{TP@k}{P}$$

$$\text{MRR@}k = \frac{1}{|Q|}\sum_{i=1}^{|Q|} r \begin{cases} r = \frac{1}{\text{rank}_i} & \text{if rank}_i \leq k \\ r = 0 & \text{otherwise} \end{cases}$$

In the MS MARCOv2 dataset, it is important to note that a single query q_i^+ can have multiple relevant documents $d_{i,j}^+$. Therefore, we opted to use recall instead of accuracy to account for this variability. When calculating MRR@k, we consider the rank of the highest-ranked relevant document $d_{i,l}^+$. In contrast, the FANDOM-QA dataset consists of a single answering passage per query, rendering the calculation of accuracy or recall indistinguishable in this context, so we sticked with recall for its validation.

4.2 Hyperparameter Search

To determine the optimal hyperparameter configuration for our ColBERT model, we conducted a manual sequential optimization strategy. We systematically changed specific hyperparameters while keeping others constant. During the search, all models were trained on MS MARCOv2. Additional details regarding the training process can be found in Appendix B.

The investigated hyperparameters included the embedding dimension (d), with values of 8, 16, 24, 32, 64, and 128. We compared three similarity calculation methods: cosine similarity, $L2$ distance, and normalized $L2$ distance similarity. We tested training with either cross-entropy or mean squared error loss. The impact of varying the number of query embedding vectors (N_q) was also explored, with values of 32 and 64. Furthermore, the performance of BERT and RoBERTa architectures as backbone models was compared. Additionally, the influence of reducing the ratio $\hat{k} = \frac{k}{\lambda}$ on the end-to-end retrieval performance was examined. Detailed results for each hyperparameter configuration can be found in Figure 6 for end-to-end retrieval and Figure 7 for re-ranking.

The evaluation results indicate that the original configuration used in the ColBERT paper is mostly optimal in terms of hyperparameter settings. However, a slight improvement in performance was observed with an increased value of N_q. The best performing setting determined by our search is as follows: 128 embedding dimensions, cosine similarity, cross-entropy loss function, BERT as the backbone model, 64 embedding query vectors, and $\hat{k} = k/2$. Interestingly, decreasing the value of $\hat{k}$ did not significantly impact the retrieval process, particularly for the top 10 results which receive the most focus in our retrieval process, as users are less likely to explore lower-ranked results. Therefore, we set $\hat{k} = k/5$ to reduce the number of passages to rank and consequently save inference time.

Interestingly, during the evaluation phase, the model using a BERT backbone consistently outperformed the model with a RoBERTa backbone, despite RoBERTa being a superior trained BERT model. To strengthen this observation, we conducted tests on RoBERTa with various hyperparameter settings. However, all of these settings resulted in poorer performance compared to the BERT backbone. Regrettably, we are unable to explain the underlying cause of this behavior.

4.3 Multi-Stage Fine-Tuning

After identifying the best hyperparameters in Chapter 4.2, we conducted a Multi-Stage Fine-Tuning process to train a robust retrieval model for our FANDOM-QA dataset. In the first stage, we preadapted ColBERT on MS MARCOv2, as the dataset provides fundamental semantic knowledge for ColBERT in retrieving correct answers to queries. In the second stage, we performed fine-tuning seperatly on the FANDOM-QA and Harry Potter datasets. We trained those two models separatly, since we wanted to test if it is necessary to train a model for each wiki seperatly or if a more general model would be similarly capable. We should mention that we not only trained models with 128-dimensional embeddings but also with 24-dimensional embeddings. We made this decission, since choosing 24 dimensions reduces the required storage for the precomputed document embeddings by a factor of 5, while maintaining reasonable retrieval performance. Additionally, we increased N_q to 48 to enhance performance with smaller additional computational cost compared to 64. During this training, a learning rate of 3e-6 and a batch size (B) of 42 were employed for improved stability.

The results for the first stage, along with a comparison to the classical retrieval method TF-IDF, can be seen in Figure 2. It is evident that all ColBERT settings perform significantly better than TF-IDF, with 128-dimensional embeddings and end-to-end retrieval yielding the best recall. Interestingly, the re-rank method outperforms end-to-end retrieval for 24-dimensional embeddings. Due to time limitations, we were unable to investigate the cause of this behavior further, but it presents an intriguing avenue for future research.

The end-to-end retrieval performances of the second stage models on the Harry Potter dataset, as seen in Figure 3, show similar performance between the models fine-tuned on FANDOM-QA and Harry Potter, with a slight lead observed for the FANDOM-QA model. However, when comparing these models to the preadapted model, a significant performance difference was observed. This difference could be attributed to the fine-tuned models being trained on well-formulated questions and performing on a smaller information pool, while the preadapted model dealt with more diverse and extensive topics. The fine-tuned models may have also adopted certain formulations frequently used by GPT, leading to improved performance.

Figure 2: Retrieval Performance on MS MARCOv2 (Eval)

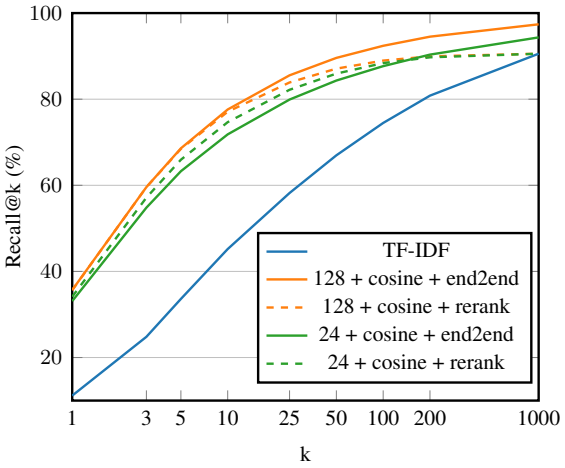
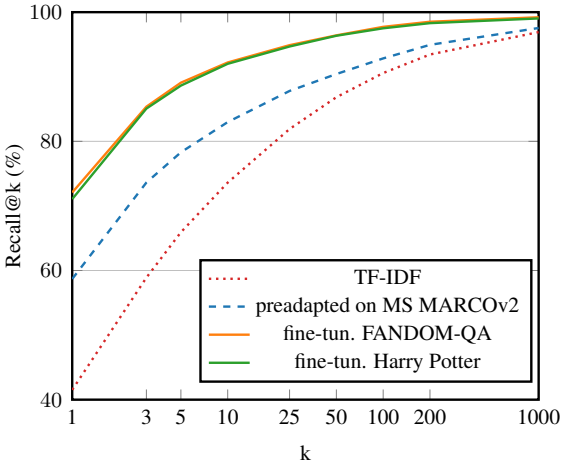


Figure 3: End-to-End Retrieval on Harry Potter ($d = 24$, $sim =$ cosine)



We also evaluated the generalization capabilities of the models to unseen data, by utilizing the Witcher dataset. The results, seen in Figure 8, indicated that the more general model performed better than the model trained solely on Harry Potter. However, the performance of both models was still quite similar, suggesting again a potential flaw in the autogenerated questions generated by GPT-3, since repetitive formulation in the fine-tuned models may have led to an excessive performance in this context.

In conclusion, the end-to-end retrieval method is superior to the re-ranking approach. For retrieval on fandoms, a more general model (all wikis) appears to perform as well as more specialized (single wiki) models while exhibiting better generalization capability to unseen data. However, further research with more diverse data should be conducted to validate these findings.

5 Model Interpretability and Explainability

As part of our study, we aimed to understand how our ColBERT model trained on MS MARCOv2 works, assess its strengths and weaknesses, and determine the location of the answer within the extracted passage. To achieve this, we have highlighted the tokens in the passage that are highly relevant to ColBERT (Chapter 5.1), improved this highlighting method using Kernel Density Estimation to locate the answer in the passage (Chapter 5.2), and summarised the main observations in (Chapter 5.3). Finally, in Appendix E, we conducted small-scale experiments to evaluate the respective contributions of the word embeddings and the transformer architecture to the performance of ColBERT.

5.1 Visualization of the Relevance of Passage Tokens

To find relevant tokens in the passage, we first note that ColBERT uses the similarity between query vectors and passage vectors to determine the correct passage. In addition, by visualising the similarity between passage tokens and query tokens as in Figure 10, we found that the embedded query vector E_{q_i} at index i encodes a large part of the information of the query token q_i at index i. The same relationship was found between the embedded passage vector E_{d_j} and the passage token d_j. Therefore, a passage token d_j is considered relevant to ColBERT regarding a question q if the associated vector E_{d_j} contributes to a high similarity score $S(q, d)$. This occurs when E_{d_j} exhibits the highest similarity to one or multiple query vectors E_{q_i}. We identified the $\tilde{k} = 2$ most similar passage vectors for each query vector, although the similarity score S only considers the top 1 to strike a balance between marking a sufficient amount of information without assigning high relevance to each passage token. The relevance R of a passage token d_j can be determined by the $absolute\ count$ of query vectors that exhibit high similarity to E_{d_j}.

$$R_{abs}(q, d_l) = |Q(q, d_l)| \quad \text{, where} \quad Q(q, d_l) =$$

$$\{E_{q_i} \mid sim(E_{q_i}, E_{d_l}) \in \underset{j=1,\dots,L_d}{max_{\tilde{k}}}\ sim(E_{q_i}, E_{d_j})\}$$

and $max_{\tilde{k}}$ returns the highest $\tilde{k}$ elements computed while iterating over index j. Since the relevance of a passage vector depends not only on its frequency among the top $\tilde{k}$ similarities of the query vectors, but also on the value of each individual similarity, we have extended the similarity score R_{abs} to R_{acc}:

$$R_{acc}(q, d_l) = \sum_{E_{q_i} \in Q(q,d_l)} sim(E_{q_i}, E_{d_l})$$

5.2 Locating the Answer in the Passage

Although ColBERT identifies the passage containing the answer to the given question by comparing the passage embedding with the query embedding, in most cases the answer itself cannot be identified directly by comparing these two embeddings using R_{abs} or R_{acc}. In most cases, the most relevant tokens of the passage are not the answer itself, but rather the tokens that lead to the answer. When comparing the query 'who won the football championship in 2006?' and the passage 'the football championship in the year 2006 was a great sports event that was won by italy.' of Figure 9 the highest relevance will be found in the words 'the', 'football', 'championship', '2006', 'was', 'won' and 'by' instead of the actual answer 'italy', because the actual answer is not relevant to decide whether the question is answered in the passage. ColBERT was not trained to find the answer, but rather to determine whether the answer is contained within the passage. Replacing 'italy' with any other word would have a minimal effect on the "probability" of the passage providing an answer to the question. Our approach to solving this problem is to use Kernel Density Estimation (Weglarczyk, 2018), by assuming that the answer is likely to be located in the areas of the passage that contain a higher concentration of relevant tokens with respect to the question.

Kernel Density Estimation is a statistical method that estimates the underlying probability density function of a random variable by placing a kernel function on each data point and summing them up to create a smooth density estimate. Given that the queries are encoded in N_q embedding vectors, calculating the top $\tilde{k}$ passage vectors for each query vector yields a total of $\tilde{k} \cdot N_q$ one-dimensional index points ranging from 0 to $L_d - 1$ and containing duplicates. The tokens `[CLS]`, `[D]` and `[SEP]` are ignored by KDE. Applying KDE to this data sample allows us to identify areas in the passage with a high density of relevant tokens, indicating not a high probability, but the relevance of that particular area in the broader context.

The assumption that the answer is likely to be found

in these dense areas of the passage is often correct, and therefore useful for quickly identifying the likely location of the answer in the passage, but not reliable. KDE sacrifices information by smoothing compared to using R_{abs} for visualisation. The three visualisations (Figure 11, 12) provide insight into the relevance of the retrieved information to ColBERT. While KDE is suitable for regular users looking for answers with easy to understand markings, R_{acc} is more suitable for gaining insight into the functioning of ColBERT and for troubleshooting.

5.3 Observations during Visualization

To further analyse our model, we extracted questions and passages from the FANDOM-QA dataset on which Colbert did particularly well and particularly badly, and did the same for TF-IDF. We then manually analysed the visualisation of the intersection of each of the 4 sets. In addition, we used single token similarity, as shown in Figure 10, which visualizes the similarity of the passage to the exemplary single tokens 'who', 'when' and 'where'. Colbert's understanding obviously goes beyond synonyms: the token 'in' occurs twice in the passage, but the similarity of 'when' is only high in relation to the first, and the similarity of 'where' is only high in relation to the second. The major strength of ColBERT lies in the context-aware embeddings (Chapter E), where tokens are no longer considered in isolation. The context 'germany' changes the word 'in' from a time-related word to a place-related word.
However, ColBERT encounters similar issues as single-word-based systems like TF-IDF. ColBERT often misses the point of the question and instead retrieves passages that provide extensive information about the content of the question. Passages that provide the answer but have a different core topic are overlooked. It also seems that not enough attention is paid to interrogative words such as 'who', even though these words are well understood by ColBERT, as Figure 10 demonstrates. This behavior can be explained by the fact that during the training phase, there are only a few passages that contain substantial information about the topic of the question but do not directly answer the question. Hence, ColBERT learns to find the right passage without paying attention to the specific type of question. Furthermore, each query vector contributes equally to the sum of similarities. Assuming a strict 1:1 relationship between query token and query vector, the interrogative word will not be given more weight than any other token in the query, but this assumption needs further investigation. We also found that the vectors of the `[MASK]` tokens also carry information, e.g. two `[MASK]` tokens have different similarities to the tokens from the passage. This shows at the very least a partial resolution of the 1:1 relationship between query token q_i and query vector E_{q_i}.

6 Conclusion

In this research, we focused on implementing a semantic retrieval system that performs extractive question answering on fandom wikis. Our system effectively cleansed the given fandom dataset and generated its own training and validation datasets. This approach allows for easy reuse and further development of the system. The final model produced satisfactory results. Pre-computation of model weights and document embedding vectors allowed for fast response retrieval. Multi-stage fine-tuning proved useful in achieving high recall rates. Our research showed that neural retrieval significantly outperformed classical retrieval methods, mainly due to the context awareness of ColBERT. The optimal hyperparameters we found are consistent with the original paper. Furthermore, the presented visualisation methods proved to be useful to gain insights into the functioning of CoLBERT, to troubleshoot problems and to find answers within the extracted passage.
Moving forward, future research should focus on improving question generation to obtain a more diverse range of questions. Furthermore, it is crucial to encourage the model to understand the intent behind the question, rather than just focusing on the topic itself. To achieve this, giving more weight to the question words and evaluating the model's performance based on this aspect would be a valuable area of research.

References

Giuseppe Attardi. 2015. Wikiextractor. https://github.com/attardi/wikiextractor.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Kailash A. Hambarde and Hugo Proenca. 2023. Information retrieval: Recent advances and beyond.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. Colbertv2: Effective and efficient retrieval via lightweight late interaction.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision.

Stanislaw Weglarczyk. 2018. Kernel density estimation and its application. *ITM Web of Conferences*, 23:00037.

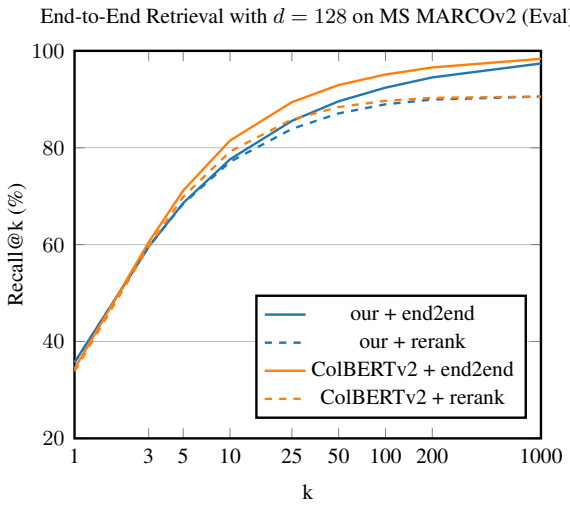


Figure 4: Comparison between our ColBERT implementation and the official ColBERTv2 checkpoint

A Validation of our implementation

To compare our implementation of ColBERT with the official implementation, we trained our model using the same hyperparameter configuration as the official ColBERTv2 model (Santhanam et al., 2022) on the MS MARCOv2 dataset. It is important to note that our training approach closely resembles ColBERTv1, so we did not expect identical performance to ColBERTv2. However, since only the weights of ColBERTv2 are publicly available, we compared our model to this version. Due to computational constraints, we were unable to compute passage embeddings for the entire set of 8.8 million passages in the MS MARCOv2 dataset for our application. Consequently, our evaluation was limited to the 1 million validation passages. As a result, a direct comparison with the official results was not feasible. Instead, we recalculated the performance metrics solely based on the validation passages. The comparison results are shown in Figure 4, where our model performed slightly worse, but still fairly similar to the offical ColBERTv2 checkpoint.

B Constant Settings for Neural Model Training and Hyperparameter Search

Throughout the training runs, we maintained a batch size B of 42 on MS MARCOv2 and 128 on FANDOM-QA, a dropout rate of 0.1, and utilized the AdamW optimizer with a learning rate of 3e-6. Additionally, we employed linear warmup of the learning rate during the first epoch. We increased the value of N_d to 320 compared to the official ColBERTv2 checkpoint, which used 180. Automatic mixed precision was applied during training and retrieval, with the embeddings being stored with 16-bit precision.

The hyperparameter search was performed only on MS MARCOv2, with the only difference being the use of a smaller batch size of 28 and a higher learning rate of 5e-6.

C Hardware specification

The training of every model was conducted on a single NVidia A100 GPU with 40GB of memory, which was made available through the High-Performance Computing (HPC) infrastructure provided by the ZIH (TU Dresden).

Training time had great variety, every trained model took 3 to 6 epochs to converge. The training of the preadapted models took about 12 to 18 hours. Fine-tuning on the FANDOM-QA dataset was done in under an hour, and on the Harry Potter dataset after mere minutes.

D Similarity Measures

We tested two differente similarity meassures

(1) *cosine similarity*:

$$sim_{cos}(E_{q_i}, E_{d_j}) := \frac{E_{q_i}^T E_{d_j}}{\|E_{q_i}\| \|E_{d_j}\|}$$

(2) *negated squared L2-norm* with an optional normalization of the embedding vectors beforehand:

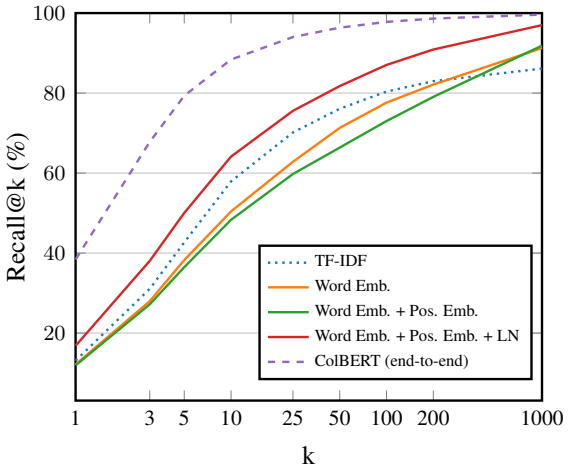
$$sim_{L2}(E_{q_i}, E_{d_j}) := -\|E_{q_i} - E_{d_j}\|^2$$

$$sim_{L2,norm}(E_{q_i}, E_{d_j}) := -\| \frac{E_{q_i}}{\|E_{q_i}\|} - \frac{E_{d_j}}{\|E_{d_j}\|} \|^2$$

E Contribution of Contextual-Aware Embeddings

To investigate the importance of the Transformer encoder, we analyzed the learned embeddings of our ColBERT model, which was trained on MS MARCOv2. Due to the large size of the resulting 768-dimensional word embeddings, we performed the evaluation on the smaller MS MARCOv1 dataset. For our testing we removed the transformer backbone and the final linear mapping and used solely the output of the embedding layer for the end-to-end retrieval process. We tested three experimental settings: (A) utilizing only word embeddings, (B) incorporating word embeddings with

Figure 5: End-to-End Retrieval Performance of Different Embedding Methods on MS MARCOv1



positional embeddings, and (C) combining word and positional embeddings with LayerNorm.

The results, presented in Figure 5, indicated that employing either word embeddings or word and positional embeddings achieved performance comparable to that of TF-IDF. However, when LayerNorm was also applied, the model slightly outperformed TF-IDF. Nevertheless, when compared to the complete ColBERT model, the exclusive use of context-unaware word embeddings resulted in inferior performance. We therefore suspect that the contextual information encoded in the embeddings by the transformer backbone significantly contributes to the overall performance of ColBERT.

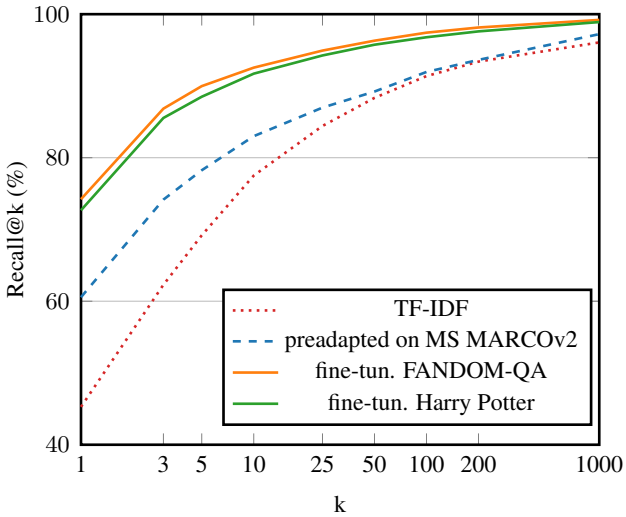
Figure 6: End-to-End Retrieval performance of experimented models with $k = 1000$, $\hat{k} = \frac{k}{\lambda}$ on MS MARCOv2. Unlisted values are identical to those of the base model.

	Backbone	d	sim	norm	Loss	N_q	λ	MRR@10	Rec@1	Rec@10	Rec@1000
base	BERT	128	cosine	yes	cross-entropy	32	2	50.42	35.65	76.97	97.46
		64						49.94	35.19	76.48	96.69
		32						48.66	34.33	74.30	95.34
		24						48.56	34.29	74.23	94.86
		16						43.84	30.97	67.30	91.63
		8						30.66	21.56	47.77	71.54
			L2					47.46	33.38	72.90	96.45
			L2	no				39.04	27.41	60.61	90.88
	RoBERTa							47.50	33.55	72.98	96.51
					MSE			13.20	9.42	20.82	86.98
best						**64**		**51.11**	**36.06**	**78.06**	**97.43**
							1	50.42	35.65	76.96	97.40
							5	50.42	35.65	76.96	97.24
							10	50.41	35.65	76.93	96.29
							20	50.32	35.63	76.63	93.72

Figure 7: Rerank Retrieval performance of experimented models on MS MARCOv2. Unlisted values are identical to those of the base model.

	Backbone	d	Sim	norm	Loss Fun	N_q	MRR@10	Rec@1	Rec@10	Rec@1000
base	BERT	128	cosine	yes	cross-entropy	32	50.49	35.70	76.74	90.57
		64					50.10	35.28	76.48	
		32					49.47	34.86	75.57	
		24					49.48	34.83	75.62	
		16					47.21	33.07	72.97	
		8					41.65	28.59	66.67	
			L2				48.50	34.06	74.33	
			L2	no			42.67	29.62	67.11	
	RoBERTa						49.86	35.15	76.26	
					MSE		24.42	15.89	43.20	
best						**64**	**50.91**	**35.94**	**77.33**	**90.57**

Figure 8: End-to-End Retrieval on The Witcher ($d = 24$, $sim =$ cosine)



Query: Who won the football championship in 2006?

Kernel Density Estimation (top-2):

[CLS] [D] the football championship in the year 2006 was a great sports event that was won by italy . [SEP]

Absolute count (top-2):

[CLS] [D] the football championship in the year 2006 was a great sports event that was won by italy . [SEP]

Accumulated Similarities (top-2):

[CLS] [D] the football championship in the year 2006 was a great sports event that was won by italy . [SEP]

Figure 9: KDE, Abs. Count, Acc. Similarities - ColBERTv2
Custom Query and Passage

Query: Who won the football championship in 2006?

Similarity to the query token 'who':

[CLS] [D] the football championship in the year 2006 was a great sports event that was won by italy and took place in germany . [SEP]

Query: When did Italy win a football championship?

Similarity to the query token 'when':

[CLS] [D] the football championship in the year 2006 was a great sports event that was won by italy and took place in germany . [SEP]

Query: Where was the 2006 football championship?

Similarity to the query token 'where':

[CLS] [D] the football championship in the year 2006 was a great sports event that was won by italy and took place in germany . [SEP]

Figure 10: Single Token Similarity - ColBERTv2
Custom Query and Passage

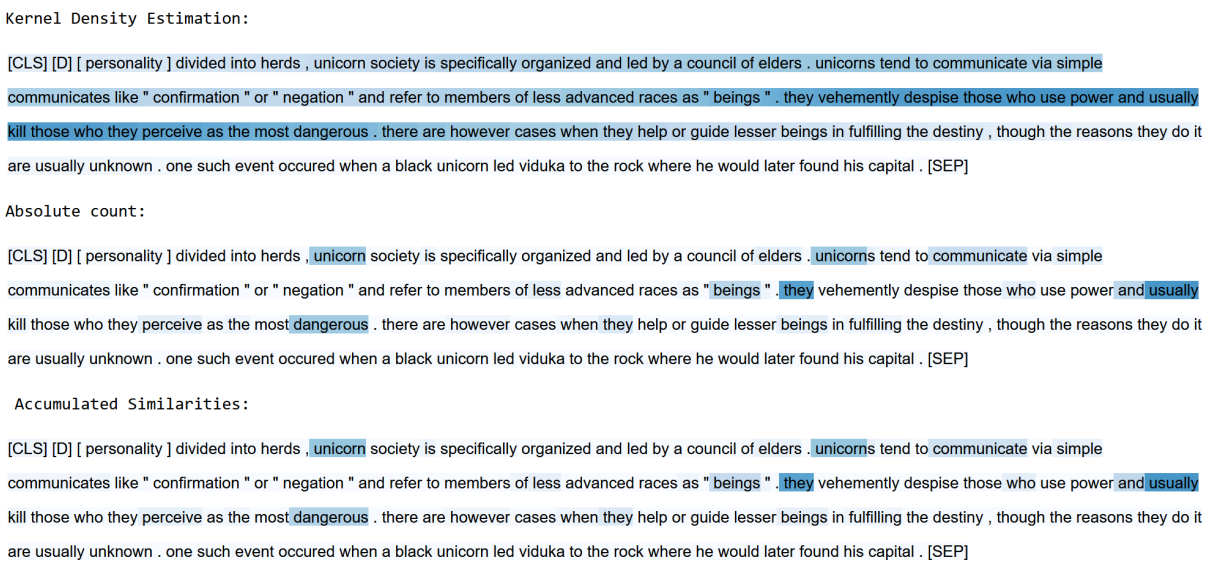


Figure 11: End-to-end Retrieval - ColBERTv2 - 24 dimensional embedding

Custom Query: Are unicorns peaceful or do they harm others?, Top 1 Passage retrieved from Witcher Fandom

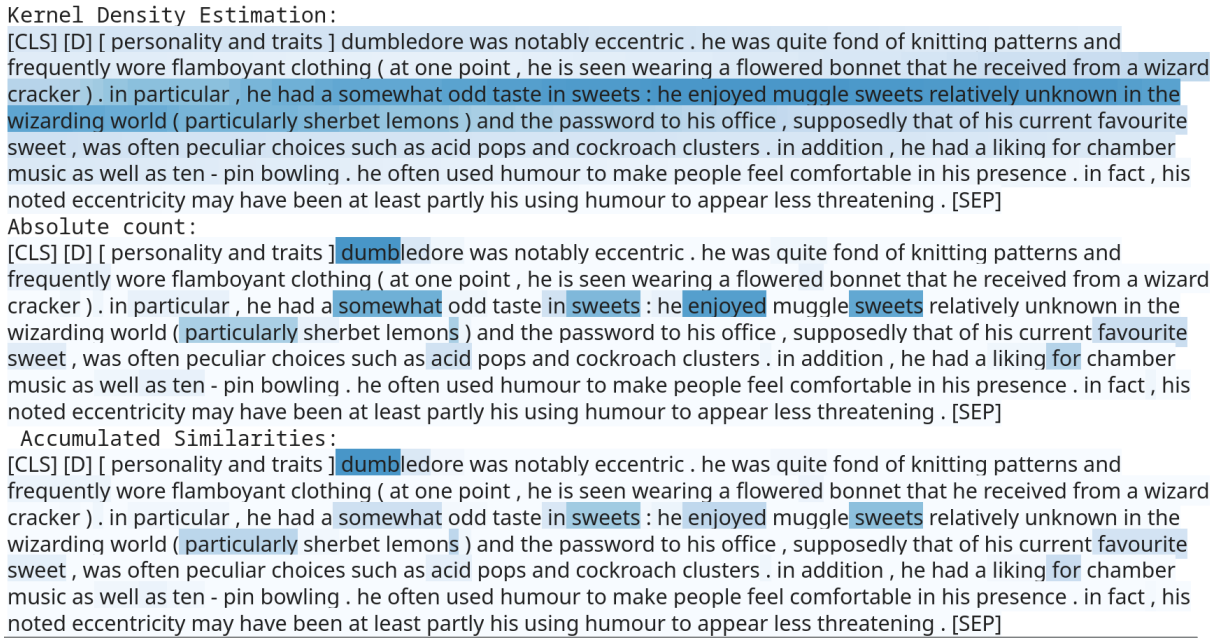


Figure 12: End-to-end Retrieval - ColBERT preadapted MS MARCOv2, fine-tuned on Harry Potter - 24 dimensional embedding

Custom Query: What is Dumbledore's favourite candy?, Top 1 Passage retrieved from Harry Potter Fandom

Figure 13: Random Samples out of the FANDOM-QA Dataset. Each passage has two positive and two negative queries.

Passage	Positive Queries	Negative Queries
Sincantar is an Altmer mystic and a member of the Mages Guild residing in Marbruk, Greenshade. He is found at the local branch, selling his wares.	1. Who is Sincantar and where does he live? 2. What is Sincantar's occupation and where can he be found?	1. What type of mystical abilities does Sincantar possess? 2. How long has Sincantar been a member of the Mages Guild?
[Lions in the wizarding world] It was a possible corporeal form of the Patronus Charm. Patricia Rakepick's Patronus took the form of a lioness.	1. What is one possible corporeal form of the Patronus Charm in the wizarding world? 2. Whose Patronus took the form of a lioness according to the given paragraph?	1. Are there any other characters besides Patricia Rakepick whose Patronus takes the form of a lioness? 2. Why do some witches and wizards choose a lion as their Patronus?
[Chest] Usually the largest storage containers found and they normally contain items of higher value than other containers. Chests can vary in appearance depending on their location. As chests are the "best" containers available, the term "chest" is often used for any storage container found within a dungeon. Chests may be locked or trapped, but never both. The one exception is the locked and trapped chest inside Pinewatch, in the room with the Unusual Gem.	1. What is the difference between chests and other storage containers in a dungeon? 2. Can a chest be both locked and trapped?	1. What is the highest value item found in a chest? 2. What are the different appearances of chests in different locations?

Figure 14: Dataset sizes and median count of words in a Passage or Query

Dataset	# Passages	# Queries	Median word count	
			Passage	Query
MS MARCO v1	707,946	92,373	72	6
MS MARCO v2	8,132,834	909,824	50	6
FANDOM-QA	705,136	576,526	53	11