Peer-to-Peer Semantic Search using LLMs

Xueyuan Chen, *Delft University of Technology Delft, The Netherlands x.chen-47@student.tudelft.nl*
Johan Pouwelse *Delft University of Technology, Delft, The Netherlands J.A.Pouwelse@tudelft.nl*

***Abstract*—The abstract of this paper**

***Index Terms*—Search engine, Semantic search, Large Language model, Database, Machine learning, Information Retrieval, Distributed Systems.**

IN our rapidly digitizing world, search engines play an important role in finding the specific information we need from the huge amount of content online. Internet users use them to learn new things and for entertainment, such as finding online videos based on personal interests. While popular search engines are helpful, mainstream ones like Google raise significant privacy concerns, as the data they gather is often utilized for purposes beyond improving search results, including personalised advertising. This concern highlights the need for local search capabilities that respect user privacy while efficiently navigating local resources such as videos or documents based on search queries.

Pursuing efficient and privacy-preserving search mechanisms has led to the exploration of fuzzy and semantic search techniques, going beyond simple and strict keyword-result searching. Techniques such as BM25 ranking [1] have significantly improved the performance of search engines like Elasticsearch. Modern search engines also leverage artificial intelligence and natural language processing (NLP) for better performance. The recent popularity of applying large language models (LLMs), like using GPT4 [2] as an alternative to traditional search engines, illustrates a growing trend in search technology. This shift is particularly relevant in the context of edge computing, which brings up an interesting possibility of employing language models as localized search engines in environments constrained by computing power, such as mobile devices.

Motivated by the evolving computational capabilities of mobile devices and the potential development in distributed learning [3], this research aims to explore the feasibility of using LLMs for local semantic search functionalities, taking the example of searching for YouTube videos on devices with limited computing power through natural language input. While LLMs are not designed to replace traditional SQL databases, as they may struggle to achieve the same levels of stability, availability, and data integrity [4], they offer unique advantages in semantic search capabilities. LLMs can support complex queries and understand nuanced relationships between words, as demonstrated by the classic example of "king - man + woman = queen". Although LLMs are computationally expensive, they provide a promising alternative to immature and costly semantic SQL search solutions.

In this study, we position LLMs as a novel type of semantic database, focusing on their ability to enable powerful semantic search functionality. However, the storage characteristics of LLMs remain largely unknown, particularly in terms of stochastic insert and select operations. Our research aims to quantify these unknown properties through several experiments centered around training and evaluation metrics such as recall. We emphasize the usability of LLMs in real-world systems, striving for precision and recall rates exceeding 98%. Additionally, we investigate the storage capacity of these semantic databases using YouTube video search as a research case.

Recent developments, such as Apple's potential deployment of LLMs within each new iPhone [**?**], further highlight the relevance and timeliness of this research. With billions of devices potentially leveraging LLMs for local search capabilities, understanding the performance and limitations of these models becomes increasingly important. Moreover, the self-retrieval architecture proposed by Alibaba [5], which consolidates indexing, generating, and self-assessment using LLMs, provides a foundation for our exploration of LLMs as semantic databases.

This research targets all users of modern search engines, with a particular focus on mobile device users constrained by limited computing resources. Through experiments with state-of-the-art language models like BERT [6] and T5 [7], we seek to evaluate their capacity to store and retrieve key-value type data, such as video IDs corresponding to video information, paving the way for a new type of local or distributed search engines optimized for privacy, efficiency, and accessibility. This article is structured as follows: In section 2, we formulate the main problems to resolve in this study. In sections 3 to 5, we describe the experiments with two language models. We discuss the results in section 6 and conclude our study in the last section.

I. Problem Statement

The problem addressed in this research revolves around the exploration of machine learning (ML) models as an alternative to traditional search engine databases, particularly in distributed computing environments with limited computational resources. The core issue at hand is the exploration of how an ML model, specifically a language model, can effectively function as a search engine database to retrieve video IDs from online videos based on partial information derived from video titles.

Traditional databases and search engines, while overlapping in functionalities such as data storage, retrieval, and modification (CRUD operations: Create, Retrieve, Update, Delete) [4], differ significantly in their output requirements. Databases typically demand strict, exact outputs, while search engines often cater to ambiguous or fuzzy searches to find the

most relevant results. This distinction means the need for a search mechanism that can intelligently navigate through semi-structured or unstructured data, with semantic understanding to deliver precise search outcomes.

Additionally, the challenge extends to the user's awareness of the content within the target dataset. Does the user know what they are looking for, or is the search driven by a vague sense of what the content might contain? Modern search engines go beyond only keyword matching, aiming to semantically interpret the user's query in relation to a knowledge base. This research, therefore, seeks to push the boundaries of conventional search engine databases by applying language models that can understand and respond to user queries with an high level of relevance and accuracy, all within the constraints of limited computing power typical of distributed computing devices.

The model's task revolves around learning and generating video IDs from user queries about video content. Video IDs are traditionally structured in a base64 format, representing a seemingly random combination of letters and digits. This presents a unique challenge for the model: understanding the semantic context of a query well enough to produce a meaningful and precise video ID that corresponds to the stored information. Unlike traditional keyword-based retrieval systems, this requires the model not only to grasp the gist of the query but also to map this understanding to a specific, and semantically unrelated, string of characters.

The primary goal for the model is to accurately generate video IDs based on natural language queries. This involves learning a mapping between the semantic content of user queries and the specific video IDs that correspond to content meeting the search criteria. Given a query q, which reflects what the user seeks, the model should output the video ID i corresponding to the relevant video title. There might also be intermediate output in the form of a video title, which is the title memorized by the LLM or the title the LLM deems most relevant to the query.

II. Experiment - BERT

In this section, we investigate the capability of large language models (LLMs) to function as semantic search engines for video retrieval, where the task is to accurately search for a video URL given a semantic input, such as a natural language query or a title-like description. Since video IDs can directly form a Youtube video URL by adding the prefix "youtube.com/watch?v=", we choose video IDs as the desired outcomes. The success criterion for our model is its ability to grasp these semantic inputs and yield accurate video IDs. This goal is similar to the key-value pair retrieval in a traditional database, where the key is some query and the value is the corresponding video ID.

As an initial setup, we focus on using video titles as queries as the starting point because video titles inherently contain information about the video that can be utilized for search purposes. This simulates a real-world scenario where users search for videos using natural language queries or title-like descriptions. It is important to note that this experiment aims to evaluate the broader potential of LLMs in understanding and relating queries to video content, extending beyond the limitations of exact title matching. To achieve this, we have chosen to fine-tune a state-of-the-art LLM model, BERT [6] (Bidirectional Encoder Representations from Transformers), on a dataset of video titles and their corresponding video IDs. Through fine-tuning we adapt BERT's pre-trained language ability to the specific task of video title matching. This approach does not only reduce training time but also ensures our model can effectively capture the linguistic patterns in the video titles.

BERT has been released in several variations to accommodate different use cases. These include "cased" and "uncased" versions for English text, as well as multilingual models trained on a broader range of languages. For this experiment, we assume that users do not depend on capitalization when creating queries. Therefore, we selected the BERT-uncased model, pre-trained on English text and insensitive to capitalization differences (e.g., "english" vs. "English").

Our task involves fine-tuning BERT to process a text query and predict the integer index corresponding to the most relevant sample in a dataset, which contains both the video title and its unique video ID. We opt to predict the row index instead of the video ID to gain two advantages. The first advantage is the simplified output Space. Video IDs are complex 11-character strings, with most characters randomly selected from a 64-character set, making them challenging to directly predict. By predicting an integer index, we reduce the model's output space to a finite set of numbers, simplifying the prediction task. The second advantage is the extensibility of the output. Row indices provide a more flexible representation than video IDs alone. Predicting the row index allows us to easily retrieve not only the video ID but also other relevant information associated with the video, such as the title or additional metadata, which could be useful in future extensions of this work.

A. Dataset and Preprocessing

The dataset chosen for this experiment is the 'Trending YouTube Video Statistics (daily)' dataset [8] from Kaggle, which contains daily records of top trending YouTube videos across various regions, including the US, Germany, and France. We specifically focused on the US subset due to its English-based content because of the BERT release version we use. This dataset is a forked version of the dataset [9]. Compared to the original version, the forked US videos subset offers a significantly larger volume of data, with 48,471 unique video titles. The dataset provides a one-to-one mapping between video titles and their corresponding 11-character video IDs (consisting of uppercase and lowercase letters, hyphens, and underscores), which simplifies the task of video ID prediction. Henceforth, this dataset is referred to as 'US Videos' III.

We performed a preliminary experiment and uncovered two methods to improve our model. Our setup consisted of separating a subset 4577 samples from the entire US Videos dataset as the training dataset for a faster experiment. Then

we performed pre-processing on this dataset before training to ensure data consistency and compatibility with the model. Firstly, any irrelevant or inconsistent information that may negatively impact the model's learning process is removed from the titles including special characters or HTML tags and leading/trailing whitespaces. As we use the uncased version of BERT, which is not sensitive to letter casing, the original case of each title is not preserved. Secondly, to maintain the focus on English-language content and align with the linguistic scope of the pre-trained BERT model, titles containing non-English characters are filtered out. Each preprocessed title is tokenized using BERT's tokenizer, which converts the text into a sequence of tokens that the model can process. The tokenizer handles tasks such as splitting words, handling punctuation, and converting the tokens into embeddings - their corresponding numerical representations.

B. Metrics

We use several traditional metrics commonly used in Information Retrieval tasks: precision, recall, and F1 score to evaluate the performance of the fine-tuned model. All metrics are calculated based on the row indices and the predicted row indices.

1) Precision: Precision reflects the ability of a model to identify only the relevant instances in classification tasks. It is mathematically the number of true positives divided by the number of true positives plus the number of false positives. In our context, precision represents the fraction of correctly predicted video IDs among all video IDs predicted by the model. A high precision indicates that the model's predictions are highly reliable.

2) Recall: Recall reflects the ability of a model to find all the relevant instances in a dataset. It is calculated by number of true positives divided by the number of true positives plus the number of false negatives. For our task, recall represents the fraction of correctly predicted video IDs among all relevant video IDs in the dataset. A high recall indicates that the model can identify a large portion of relevant videos.

3) F1 Score: F1 score is a measure combining both precision and recall by calculating their harmonic mean. It is calculated as follows:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

A high F1 score indicates a good balance between the predictions' reliability and covering as many relevant predictions as possible.

C. Training and Evaluation

The process of fine-tuning and evaluation of the model for video ID retrieval is illustrated in Figure 1. The figure outlines the steps involved, starting from the US videos trending dataset, which contains video IDs, titles, tags, and other relevant information. During the pre-processing phase, cleaning, deduplicating the titles, and adding integer indices as well as tokenization are performed to prepare the data for model training. During the training phase, the BERT

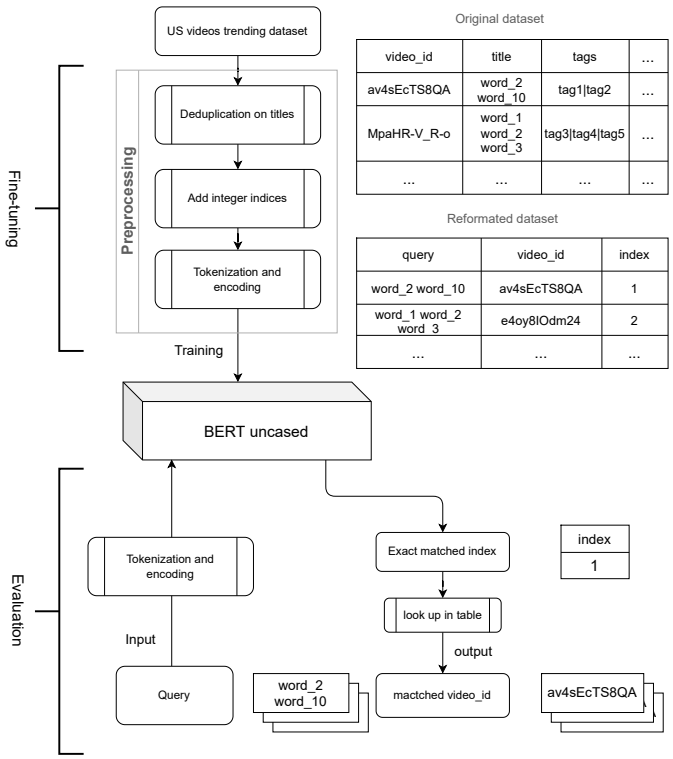

Fig. 1: BERT experiment process

base uncased model is fine-tuned on the processed dataset, effectively incorporating the knowledge of video titles and the corresponding sample indices. In the evaluation phase, the fine-tuned BERT model takes a given title as input and generates the integer index, which is then used to look up the matched video ID in the mapping table. This retrieved video ID can subsequently be used to construct the corresponding video URL, enabling semantic video retrieval based on the input query or title.

We use the sequence classification capability of the BERT model, which is designed to classify an entire input sequence into one of several categories. This is achieved by passing the output of BERT through a sequence classification head, consisting of a linear layer. This linear layer produces a set of scores, called logits, one for each potential class. In our case, each class corresponds to a unique video in the dataset. The model then selects the class with the highest logit score as its prediction for the most relevant video title. During training, a softmax function is applied to these logits to convert them into probabilities, which are then used to calculate the cross-entropy loss:

$$L_{\text{cross-entropy}} = -\sum_{i=1}^{C} y_{o,i} \log(p_{o,i})$$

where C is the number of classes which is the number of unique video titles, y is a binary indicator (0 or 1) of whether predicted row index i is the correct classification for observation o, and p is the predicted probability that observation o is of class i. The cross-entropy loss function quantifies the difference between the predicted probability

distribution and the true label. By minimizing this loss, the model learns to adjust its parameters so that its predictions align more closely with the ground truth.

The model is trained on a Google Colab [10] instance with the following hardware specifications:

- Instance type: n1-highmem-2
- vCPU: 2 @ 2.2GHz
- RAM: 13GB
- GPU: 1 NVIDIA Tesla T4

Preliminary experiments with 2 epochs showed a decrease in loss, and further training for 8 epochs with an initial learning rate of 0.001 and a linear scheduler led to convergence. The training parameters can be found in tableIV in Appendix.

D. Results

We evaluated the model's performance on the training set (4577 samples of US videos) itself, which differs from the conventional approach of using a separate test set. This choice aligns with our objective of enabling the model to memorize the dataset for precise retrieval. In this context, overfitting, which is typically undesirable, is actually beneficial as it allows the model to memorize the input data of video classification information based on the titles as much as possible, which further helps find corresponding video IDs of the queries.

The fine-tuned BERT model achieved a precision of 95.79%, a recall of 99.10%, and an F1 score of 97.41% on the training set. The high precision indicates that when the model predicts a video ID, it is highly likely to be the corresponding one. The exceptional recall value suggests that the model successfully identifies a vast majority of the relevant videos for the given titles in the training set. The high F1 score shows the model's overall effectiveness in accurately matching video titles to their corresponding IDs.

Our approach has limitations in terms of generalization on partial or modified titles. When presented with queries that were not part of the training data, the model's performance is expected to decline. This is because the model has been optimized to memorize the exact titles rather than learning to generalize to unseen variations. We performed qualitative analysis using only part of the titles as queries. This confirmed our expectation that the model's performance would drop significantly. While this lack of generalization may be seen as a drawback in other contexts, it aligns with our specific goal of retrieving video IDs based on exact title matches.

It is important to acknowledge that the model's effectiveness relies on its ability to memorize the training samples. By overfitting towards the training data, the model can achieve a high recall, ensuring that it captures the most relevant video IDs in the dataset. Although this approach may not be suitable for scenarios requiring generalization on unseen data, it is suited well for our task of accurate video retrieval given queries based on exact titles.

III. Experiment with T5

The previous approach using BERT as a classifier for video retrieval faced a significant limitation: the inability to directly generate video IDs. BERT, being an encoder-only model, requires an external mapping between the predicted indices and the corresponding video IDs. This indirect approach introduces an additional layer of complexity and storage requirements, as the mapping needs to be maintained separately. To address this issue, we explored the use of the "Text-to-Text Transfer Transformer" (T5) [7] model, which has an encoder-decoder architecture capable of directly generating video IDs.

Our motivation for choosing the T5 model comes from its ability to learn and generate sequences, making it well-suited for the task of mapping video titles (a sequence of words) to video IDs (a sequence of characters) without the need for an intermediary mapping step. The T5 model has demonstrated success in similar tasks, such as generating document IDs from queries [11]. By leveraging the sequence-to-sequence (seq2seq) nature of T5, we aim to create a direct mapping between the input video titles and the generated video IDs, eliminating the need for external storage of mappings.

For this experiment, we chose the flan-T5-small variant [12] of the T5 model. The T5-small model is a smaller version of T5 with 60 million parameters, making it more suitable for environments with limited computational resources. The "flan" version of T5 is an updated release that has been fine-tuned on more than 1,000 additional tasks, covering a wider range of language processing tasks such as question answering and chain-of-thoughts [13] [14] compared to the original T5 model.

To encode the video IDs, we applied the Naively Structured String Identifiers strategy [11]. In this approach, we used T5's original tokenizer to encode the video ID token-by-token, where each token can be any substring of the video ID. For example, the ID 'J78aPJ3VyNs' is encoded by 'J78', 'aP', 'J3', 'V', 'yNs' tokens which are already present in the T5 tokenizer's vocabulary. This strategy allows the model to learn the structure and composition of the video IDs.

A. Data Preparation and Preprocessing

For this experiment, we continued to use video titles as queries, similar to our approach in the BERT experiment. However, we performed additional data augmentation and data preprocessing steps to generalize the capability of the model to handle more queries based on the titles.

The T5 model is sensitive to the capitalization of user input. The T5 tokenizer has uppercase and lowercase letters in its vocabulary, so it distinguishes between them during training and inference. Changing the case of words in the input can lead to different model outputs. In the original dataset, many of the samples are capitalized while some are not. To make the model able to handle all lowercase queries, we created an additional lowercase copy for each sample containing uppercase.

To further improve the model's ability to learn query-video ID associations, we extracted key nouns and named entities from the video titles using the `spaCy` library. Each extracted keyword from the original video title is added as a new query with the same video ID of that title. For example, for the video title "When your cat is a real couch potato", key nouns or named entities "cat", "couch", and "potato" are extracted

and added as queries. This augmentation enables the model to focus on the most relevant information within the titles, potentially enhancing its retrieval performance.

After augmenting the dataset, we performed deduplication to ensure that each unique query maps to only one video ID. In cases where duplicate queries are mapped to different IDs, we kept only the first query-ID pair. This deduplication step was intended to enforce a one-to-one mapping between queries and video IDs. However, in retrospect, this may be a potential limitation to our work. Further experimentation without this deduplication step is planned for future work.

B. Metrics

To evaluate the performance of the T5 model, we utilized the same metrics as in the BERT experiment, namely precision, recall, and F1 score. As the T5 model might output invalid video IDs, we introduced a new metric specific to this task: the validity rate. The validity rate measures the proportion of generated video IDs that meet the format of a YouTube video ID (containing lowercase letters [a-z], uppercase letters [A-Z], hyphens [-], and underscores [_]) out of all the predicted outputs. This metric provides insights into the model's ability to generate well-formed video IDs. The validity rate is calculated as follows:

$$\text{Validity Rate} = \frac{\text{Number of valid video IDs}}{\text{Total number of generated IDs}}$$

A low validity rate could limit the model's practical utility since it suggests that the model is having difficulty understanding the structure and format of the video IDs. We may evaluate the model's effectiveness in producing precise and well-formed video IDs by taking the validity rate into account in addition to the other metrics.

C. Training and Evaluation

The model was trained in the same environment as the BERT experiment, utilizing 1 NVIDIA T4 GPU for 8 epochs. We discovered through experimental investigation that we obtained the best result with an initial learning rate of 0.001. Also, we evaluated learning rates of 0.002 and 0.0005 but observed no significant differences in performance. With a learning rate of 0.0005, the training process was notably slower. As such, we opted for the default value of 0.001 as the initial learning rate. From this starting value, the scheduler decreases the learning rate over time during training. The T5 model also use cross-entropy as its default loss function, which is commonly used in sequence-to-sequence tasks.

The training phase involved 1 experiment with 50 samples and 11 consecutive experiments across multiple increasing sample sizes, ranging from 100 to 1100 samples in increments of 100 (i.e., 100, 200, 300, ..., 1000, 1100). The AdamW optimizer [15], with an initial learning rate of 0.001, and the default linear scheduler were utilized alongside the cross-entropy loss function. Data tokenization was performed using the T5 tokenizer. It is important to note that no separate test data was used in these experiments, and the evaluation was conducted on the augmented training dataset itself. This approach aligns with the reasoning behind the BERT experiment, where the focus was on the model's ability to memorize and retrieve exact video IDs from the training set.

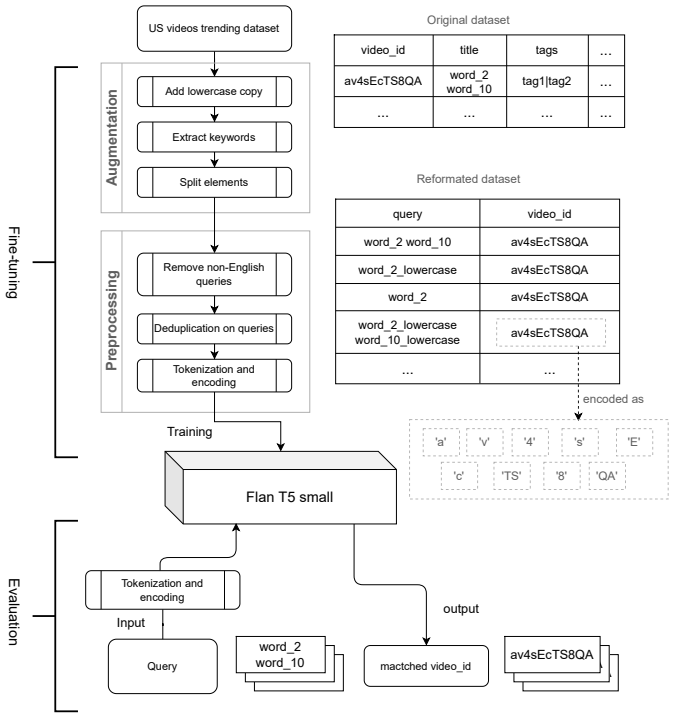

Fig. 2: Experiment process with T5 using video titles

D. Results

Training with the dataset of 50 samples achieved the most favorable loss reduction, with the loss dropping below 0.0001 after 1000 epochs, which achieved 100% precision and recall. Therefore, for small data sets, training the T5 model from the ground up is unnecessary, and fine-tuning the model is sufficient.

We now present the results for the larger sample sizes. The resulting precision of the varying sample sizes is given in Figure3. These results reveal a clear trend: the precision of the model decreases as the size of the sub-training dataset increases. This observation suggests that the T5 model faces challenges in maintaining high precision when trained on larger datasets, given the fixed number of epochs used in each experiment.

One possible explanation for the loss in precision as the data set size increases is that larger datasets introduce more complexity and diversity in the training samples, making it harder for the model to converge to a low training loss within the allocated number of epochs. For instance, a larger dataset may contain a wider variety of video titles, ranging from simple and straightforward titles like "Funny cat video" to more complex and descriptive titles such as "The Kissing Booth Cast Kisses A Hairless Cat & Other Weird Stuff — Kiss & Tell — Netflix". The diversity can be reflected in the difference in title length, structure, and vocabulary. Also, a larger dataset is likely to include a broader range of topics, genres, and styles, requiring the model to learn and memorize associations across a more heterogeneous set

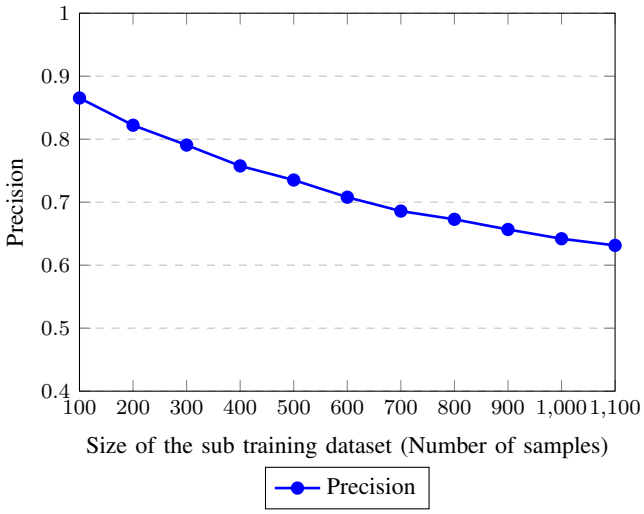

Fig. 3: Precision rate across different dataset sizes

of samples. For example, a smaller dataset might primarily consist of cat videos, while a larger dataset could encompass a mix of cat videos, cooking tutorials, music performances, and travel vlogs. Furthermore, the likelihood of encountering ambiguous or overlapping titles may increase with the dataset size growing. For instance, two videos with similar titles like "Amazing Dance Moves" and "Beautiful Dance Moves" might have different video IDs, requiring the model to learn fine-grained distinctions.

As the dataset size increases, the model requires more training iterations to effectively learn and memorize the associations between video titles and their corresponding IDs. Consequently, with a fixed number of epochs, the model ends up with a higher training loss when trained on larger datasets compared to smaller ones. It highlights the trade-off between dataset size and the model's ability to memorize and retrieve exact video IDs. While larger datasets provide more diverse and representative samples, they also pose challenges to the model's convergence and precision.

Our result shows the difficulty for the T5 model to memorize more video IDs. Training time, which positively correlates with the number of training samples, is a significant consideration in this context. This is a scalability concern for environments with constrained computing resources because it is expected to take tremendous time to let the model memorize 800 million videos which is an estimation of the number of Youtube videos as of 2023 [16], not taking into account the limit of the capability model to memorize.

E. Discussion

While the experiment illustrates that the T5-small variant faces challenges with larger datasets in terms of memorization capacity, it also brings forth an intriguing question. Why does the T5-small's ability to memorize video IDs decline with an increase in IDs, despite the apparent trend of precision drop with fewer data points? This observation may not be directly explainable by the aforementioned trends and suggests an area

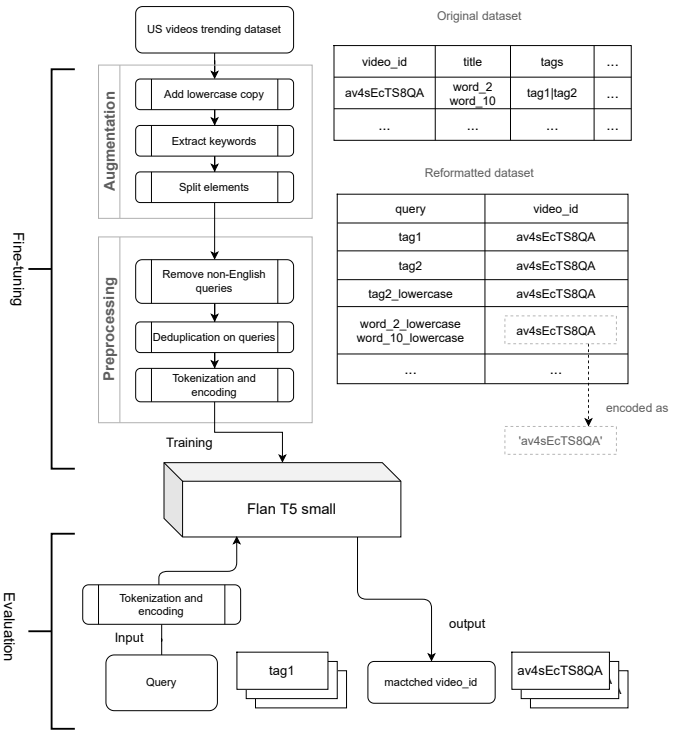

Fig. 4: Experiment process with T5 using video tags

for further investigation. It could imply a nuanced complexity in how sequence-to-sequence models like T5 deal with information density and the memorization-retrieval balance, especially when scaled down to smaller variants like the T5-small.

In summary, the T5 model demonstrates a promising capacity to memorize and generate video IDs from title inputs, though with limitations influenced by dataset size and computing constraints. This experiment not only showed the potential of utilizing language models like T5 in search engine applications but also highlighted the critical balance required between computing resources and model precision in distributed environments.

IV. Experiment with T5: use tags as queries

The previous experiment using video titles as queries provided valuable insights into the T5 model's performance. However, we recognized that video titles may not accurately reflect typical user search behavior. Users often perform fuzzy searches using keywords, rather than full titles [17]. To better simulate real-world scenarios, we explored the use of video tags as queries, as tags tend to be shorter and more keyword-oriented.

We treated this task as an information retrieval (IR) problem, where the goal was to retrieve the correct video ID given a tag (query). The overall process is illustrated in Figure 4. We used pairs of $(query, video\_id)$ as training samples, where $video_id$ was the expected output. In the US Videos dataset, each video had multiple tags stored as a string separated by the "|" character in the "tags" column. We extracted these tags and created a $(query, video\_id)$ pair for each tag.

A. Data Preprocessing

Similar to the preprocessing steps in the previous T5 experiment, we performed data augmentation using techniques including extracting keywords, splitting elements and adding lowercased queries. However, in this experiment, we applied the Unstructured Atomic Identifiers strategy [11], where each whole video ID was added as a single token to the vocabulary of the tokenizer, and the model embedding dimension was resized accordingly. This approach treats the video IDs as atomic units, enabling the model to generate them as complete entities.

B. Training and Evaluation

For the training phase, we utilized the DAS6 [18] High-Performance Computing (HPC) resources, which provided access to NVIDIA A4000 GPUs. Similar to the previous T5 experiment, we chose the flan-T5-small model variant to align with our computing resource constraints. The AdamW optimizer [15] with an initial learning rate of 0.001, the default scheduler, and cross-entropy loss were used during training. We still only use the training dataset and tokenized the data using the T5 tokenizer. Training the model for 70 epochs on the augmented full dataset of 48,266 samples required approximately 68 hours of computation time on the A4000 GPU, highlighting the significant computational demands of training language models on larger datasets.

C. Results

We evaluate the model's performance, initially focusing on the same metrics used in the previous T5 experiment: precision, recall, and F1 score. The results are summarized in Table I.

Dataset size	Augmented data-size	Epochs	Avg Recall
100	3220	70	0.957
100	3220	150	0.957
100	3220	70	0.955
1000	19382	70	0.836
10000	104833	70	0.938
20000	175364	100	0.287
48266	340884	70	0.001

TABLE I: Results of T5 experiment with tags as queries: Average recall on augmented dataset in T5 experiment

The results show that the model achieves decent recall rates on small dataset sizes but performs poorly when the dataset size is larger, based on the initial metrics. However, upon manual analysis of the incorrectly generated video IDs, we discovered a significant number of false negatives. In many cases, the video linked to the predicted video ID either contained the input tag or had relevant information in its title. For example, for the 'trailer' query, the expected video ID was present in the dataset, but the model predicted a different video ID that also contained the 'trailer' tag. For another example, when given the query "First Take," the model was expected to output a video with the tag 'first take' (jLX-tcoI7q4), but it instead outputs a video (TkWSOtqJf6I) that did not have the 'first take' tag but had 'First Take' in its title.

Dataset size	Augmented data-size	Epochs	New Metric Recall
100	3220	70	1.000
100	3220	150	1.000
100	3220	70	1.000
1000	19382	70	0.999
10000	104833	70	1.000
20000	175364	100	NA
48266	340884	70	NA

TABLE II: New metric recall on augmented dataset in T5 experiment

We recognized that this might be a drawback of removing all duplicate mappings when one tag maps to many video IDs and only outputs one video ID for each query.

Based on the manual analysis, we updated the evaluation metric to address the false negatives. We now consider a video ID prediction correct if the input tag is present in either the tags or the title of the predicted video. Using this updated metric, the model achieves near-perfect recall rates on dataset sizes of 1,000 and 10,000, with a recall of 0.999 on the 10,000 datasetsII. Furthermore, by manually examining the incorrectly predicted results, we found that all 'wrongly' generated video IDs turn out to be false negatives because they all link to relevant videos but the casing of the tag is mismatched either in the tag or the title. Upon closer inspection, the model achieved a 100% recall rate, successfully retrieving relevant video IDs even when the exact tag-to-ID mapping is not present in the training data.

D. Encoding video IDs with word list

The training target, video IDs, do not contain any semantic information; they looks like hash strings. In our previous experiments, we explored two strategies for encoding the video IDs: the Naively Structured String Identifiers strategy and the Unstructured Atomic Identifiers strategy. Although both strategies gave relatively good performance, we wondered if combining them could further improve the results. On a closer look of the Naively Structured String Identifiers strategy, we noticed that many tokens in the vocabulary of the T5 tokenizer used to encode the video id did not have inherent meanings and appear to be random substrings. However, language models may perform better when dealing with semantic information. This observation led us to hypothesize that replacing each part of the video ID with meaningful words could potentially enhance the model's performance.

We selected the intersection of the BIP39 word list [19] and the T5 tokenizer vocabulary as the vocabulary for encoding the video IDs. The words in this list are more distinct and well-separated [19], which we believed might aid the model in better understanding the semantic information. We decided to randomly select 64 words from this intersection vocabulary and performed a small-scale experiment with 10 samples to assess the effectiveness of this approach. Unfortunately, the initial results were not promising, with the model achieving a

precision rate of only 0.2568. This low performance indicated that simply replacing parts of the video ID with meaningful words from the BIP39 list did not yield the desired improvement.

E. Discussion

The T5 experiment using video tags as queries provided valuable insights into the model's performance in a more realistic search scenario. By updating the evaluation metric to consider the presence of the input tag in either the tags or title of the predicted video, we observed significant improvements in recall rates, especially on smaller dataset sizes. This optimization highlighted the importance of considering the practical aspects of search engine applications when designing experiments. In real-world scenarios, a query may be relevant to multiple videos, and the model should be able to associate a single tag with multiple video IDs. While we did not remove the deduplication step during preprocessing in this experiment, it is an important consideration for future work to better reflect real-world search scenarios.

While the word encoding experiment using the BIP39 word list did not yield promising results, it gives insight for further exploration in incorporating semantic information into the training process. Future work could investigate alternative word encoding strategies or the use of different semantic-rich vocabularies to potentially enhance the model's performance.

Overall, the T5 experiments using both video titles and tags as queries demonstrated the potential of using language models for video retrieval tasks. The results underscored the importance of selecting appropriate query types, preprocessing techniques, and evaluation metrics to align with real-world search scenarios. Furthermore, the experiments highlighted the trade-offs between model complexity, dataset size, and computational resources, emphasizing the need for careful consideration when deploying such models in resource-constrained environments. Future work should explore the removal of the deduplication step during preprocessing to allow the model to associate a single query with multiple video IDs, better reflecting real-world search scenarios. Additionally, investigating alternative approaches to incorporate semantic information into the training process could potentially improve the model's performance and generalization capabilities.

V. Conclusion

The conclusion goes here.

Appendix A
Experiments

A. Dataset

B. Training parameters

Acknowledgment

The authors would like to thank ChatGPT. ChatGPT is only used for polishing sentences and not for generating the content of this paper.

References

[1] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," *Foundations and Trends in Information Retrieval*, vol. 3, pp. 333–389, Jan. 2009.

[2] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[3] R. Ahmed and R. Boutaba, "A survey of distributed search techniques in large scale distributed systems," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 150–167, 2010.

[4] P. Dean and B. Sundgren, "Quality aspects of a modern database service," in *Proceedings of 8th International Conference on Scientific and Statistical Data Base Management*. IEEE, 1996, pp. 156–161.

[5] Q. Tang, J. Chen, B. Yu, Y. Lu, C. Fu, H. Yu, H. Lin, F. Huang, B. He, X. Han, L. Sun, and Y. Li, "Self-retrieval: Building an information retrieval system with one large language model," 2024.

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

[7] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *CoRR*, vol. abs/1910.10683, 2019. [Online]. Available: http://arxiv.org/abs/1910.10683

[8] R. SHARMA, "Youtube trending video dataset (updated daily)," https://www.kaggle.com/datasets/rsrishav/youtube-trending-video-dataset, [Online; accessed 1-March-2024].

[9] M. J, "Trending youtube video statistics," https://www.kaggle.com/datasets/datasnaek/youtube-new/data, 2018, [Online; accessed 10-December-2023].

[10] E. Bisong, *Google Colaboratory*. Berkeley, CA: Apress, 2019, pp. 59–64. [Online]. Available: https://doi.org/10.1007/978-1-4842-4470-8_7

[11] S. Zhuang, H. Ren, L. Shou, J. Pei, M. Gong, G. Zuccon, and D. Jiang, "Bridging the gap between indexing and retrieval for differentiable search index with query generation," 2023.

[12] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," in *International Conference on Learning Representations*.

[13] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, and J. Wei, "Challenging big-bench tasks and whether chain-of-thought can solve them," 2022.

[14] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=d7KBjmI3GmQ

[15] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," Jan. 2019, arXiv:1711.05101 [cs, math]. [Online]. Available: http://arxiv.org/abs/1711.05101

[16] S. Allen, "How many videos are on youtube? 33+ interesting stats," https://www.nichepursuits.com/how-many-videos-are-on-youtube/, [Online; accessed 12-April-2024].

[17] M. J. Halvey and M. T. Keane, "Analysis of online video search and sharing," in *Proceedings of the Eighteenth Conference on Hypertext and Hypermedia*, ser. HT '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 217–226. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/1286240.1286301

[18] H. Bal, D. Epema, C. de Laat, R. van Nieuwpoort, J. Romein, F. Seinstra, C. Snoek, and H. Wijshoff, "A medium-scale distributed system for computer science research: Infrastructure for the long term," *Computer*, vol. 49, no. 05, pp. 54–63, may 2016.

[19] "Mnemonic code for generating deterministic keys," https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki/, [Online; accessed 12-April-2024].

Field Name	Description
video_id	Unique identifier for each video. Useful for indexing and referencing specific videos in the dataset.
trending_date	The date when the video was trending. This can help in analyzing trends over time.
title	The title of the video. This is a crucial text field for IR, as it often contains keywords and topics that are highly relevant to the content of the video.
channel_title	The name of the channel that posted the video. This can be used for channel-based recommendations or analysis.
category_id	The category of the video (e.g., Entertainment, News, etc.). Useful for categorizing content and making category-based recommendations.
publish_time	When the video was published. This can be used to study the impact of publication time on trending status or viewership.
tags	Keywords associated with the video. Tags are extremely valuable for IR as they directly represent the content and context of the video.
views, likes, dislikes, comment_count	Engagement metrics. These can be used to gauge the popularity and reception of a video.
thumbnail_link	Link to the video's thumbnail. While not directly useful for IR, it can be used for visual analyses or to enhance the presentation of search results.
comments_disabled, ratings_disabled, video_error_or_removed	Boolean fields indicating certain statuses of the video. These can be used for filtering out certain videos from the analysis.
description	The description text of the video. Like the title, this is a rich text field that can be mined for keywords and topics.

TABLE III: Fields in the 'Trending YouTube Video Statistics' Dataset

Parameter	Value
global_step	32760
training_loss	3.232750225882245
train_runtime	3834.9877
train_samples_per_second	68.337
train_steps_per_second	8.542
total_flos	7106770821765600.0
train_loss	3.232750225882245
epoch	8.0

TABLE IV: Training parameters and outputs for BERT training