TechSpotter: YouTube Video Search Engine

DAVID CORDEIRO, Faculdade de Engenharia da Universidade do Porto, Portugal
DIOGO VIANA, Faculdade de Engenharia da Universidade do Porto, Portugal
GONÇALO MARTINS, Faculdade de Engenharia da Universidade do Porto, Portugal
MARIA SOFIA MINNEMANN, Faculdade de Engenharia da Universidade do Porto, Portugal

The *TechSpotter* project introduces a custom search engine designed to retrieve relevant information from YouTube[Steve Chen and Karim 2005] tech videos. A structured dataset was created, containing video metadata, manually written transcripts, tags, and extracted entities. Using Apache Solr[Seeley 2004], a robust indexing and retrieval system was implemented. Field boosts, schema customizations, and advanced querying techniques were applied to enhance precision and recall. The system prioritizes accurate and relevant search results by implementing a hybrid approach that combines lexical and semantic search methods. Additionally, a user interface was integrated with dynamic re-ranking based on user interactions, improving the user's overall experience and ensuring result relevance over time. The final system significantly improves search relevance and precision, as demonstrated through evaluations using precision-recall curves as well as other metrics, showing its ability to identify and rank relevant content within large-scale video datasets.

CCS Concepts: • **Information systems** → Search engine architectures and scalability.

Additional Key Words and Phrases: YouTube Tech Videos, Reviews, Information, Information Retrieval, Metadata Processing, Dataset, Data Extraction, Data Preparation, Data Analysis, Metadata Processing, Pipeline, Conceptual Model, Search Engine

ACM Reference Format:
David Cordeiro, Diogo Viana, Gonçalo Martins, and Maria Sofia Minnemann. . TechSpotter: YouTube Video Search Engine. , (), 12 pages.

1 Introduction

This project focuses on the development of *TechSpotter*, a custom search engine designed to improve the retrieval of information from a processed dataset of YouTube [Steve Chen and Karim 2005] tech videos. The dataset includes the video metadata and transcripts collected from reputable YouTube [Steve Chen and Karim 2005] channels that provide manually written subtitles, ensuring data reliability and quality for analysis.

The report is organized to provide a comprehensive overview of the project. It begins with a description of the dataset's origin and structure, followed by the Pipeline section, which outlines the processes for data collection, cleaning, and preparation. The Final Dataset Description and Conceptual Model sections examine the dataset's fields, relationships, and overall structure. The Data Characterization section further analyses the dataset through statistics and descriptive analyses.

The Possible Search Scenarios section explores practical applications of the system, highlighting its versatility for different user needs. The Indexing Process section describes the decision-making and implementation of the indexing process, while the Retrieval Process section outlines the methods used to retrieve relevant data and improve search accuracy. This section also presents the hybrid search approach implemented in the final system to enhance results.

The Evaluation and Demo section explains the process of identifying relevant documents, describes the three different system configurations, details the defined information needs, and analyzes the average precision metrics for each query across the three systems. It also assesses the effectiveness of the systems using performance metrics, such as precision-recall curves, and draws conclusions from the results.

Following this, the Experiments section discusses various methods tested to improve search performance, which ultimately did not achieve the desired results.

The Search User Interface section provides a brief explanation of the interface and its key feature: click-based signals. This feature dynamically updates the relevance of videos based on user clicks, ensuring a more adaptive and personalized search experience.

Finally, the report concludes with the Conclusion section, summarizing the project results, followed by the Future Work section, which proposes directions for further improvements.

2 Dataset Description, Origin, and Licensing

The dataset consists of a curated set of YouTube [Steve Chen and Karim 2005] tech videos. These videos were chosen based on the popularity and reputation of the associated channels, and whether they provided a manually written transcript. All the channels picked for video extraction, are English-speaking channels. The data itself consists of video metadata (e.g., title, description, upload date, views, likes, etc.) and transcripts. The dataset was constructed using YouTube's publicly available API[Google 2007]. Even though data is being collected for research purposes, the original content (video and transcript) is copyrighted by the video creators, as described by YouTube's API data licensing terms [?], and therefore the group does not claim any ownership over it nor does it intend to redistribute it.

3 Pipeline

The data pipeline begins with the process of carefully finding and selecting IDs of YouTube tech channels, following the criteria described in the previous section. This decision arose from wanting to increase the likelihood that our data came from well-known and trusted sources, and had manually written, and therefore higher

Authors' Contact Information: David Cordeiro, up202108820@up.pt, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal; Diogo Viana, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, up20210803@up.pt; Gonçalo Martins, up202108707@up.pt, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal; Maria Sofia Minnemann, up202007342@up.pt, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal.

quality, transcripts available for their videos. The acquired IDs were fed as input to a Python[Foundation 2001] script, which then processes each channel's videos sequentially, using Python's[Foundation 2001] multi-threading to greatly accelerate its processing. Metadata and transcripts are extracted from each video, through API requests, handling missing values appropriately. Transcripts are cleaned and transformed, also performing keyword and entity extraction on them. The URLs of description links are also obtained. All this data is then compiled in a JSON[Crockford 2002] document, which is then stored in the file system under a directory named after the channel. In the sections below further details are presented on how data is being collected and cleansed.

3.1 Data Collection

The processing of a YouTube[Steve Chen and Karim 2005] channel begins by using its ID in an API request to obtain a maximum of 750 videos with manual transcripts, that it may contain (for the majority of them, the number was less than that). A ThreadPoolExecutor is set up with an appropriate value of max_workers. We submit to each thread the job of creating a JSON[Crockford 2002] document from the provided video ID. The data stored in the final document is collected in 2 major ways: either directly from the video/channel through the YouTube API[Google 2007], or derived from the video's description and transcript, after their acquisition.

3.1.1 Data Obtained Directly. The collection of the metadata and transcript is performed in 2 separate steps. In the collection of the metadata, 2 requests are made to the YouTube developer API[Google 2007]. One is for the video's snippets and statistics (e.g., title, description, upload date, views, likes, etc.), and another is for channel statistics (concretely, subscriber count). The YouTube[Steve Chen and Karim 2005] service object, for these requests, is created inside this function, to ensure thread safety. The transcript is requested and stored afterwards, using YouTube Transcript API[Google 2007] specifically.

3.1.2 Derived Data. Upon appropriately transforming and cleaning the transcript, the YAKE! [TEC 2018] keyword extractor was used to obtain the top 20, dissimilar, short key phrases of a maximum of 3 ngrams, from the text. Then, using a spaCy [ExplosionAI 2016] NLP [IBM [n. d.]b] model, the program extracts and stores its named entities, separating them into their values and types. This model is instantiated globally, since it is only being used for read operations, making it thread-safe, according to spaCy's [ExplosionAI 2016] documentation. The video's description is also used to extract the URLs of any links it may contain.

3.2 Data Cleaning

Missing values are handled by providing default placeholder values for the fields. That way, if a particular statistic is not retrieved, the document is still created. We apply Python's [Foundation 2001] int() function on numeric statistics, such as views, likes, and comment count, to ensure they are stored in the final document as integers. Description URLs are also simplified by removing any unnecessary query parameters. Several transformations are applied to the video's transcript. This is done to turn its original format, full of video-specific additional information, into a more readable and naturally flowing text, as well as improve the quality and accuracy of keyword and entity extraction. Regular expressions [Wikipedia [n. d.]] are used to remove any unnecessary whitespace that arises from, for example, introducing line breaks for displaying captions. For keyword extraction specifically, spaCy's [ExplosionAI 2016] NLP [IBM [n. d.]b] model is used to remove filler words (e.g. 'um', 'uh', 'you know', 'basically' etc. and), anything with parentheses or brackets (effectively removing annotations, etc.) and, hyphens are eliminated, and contractions, such as don't, are expanded.

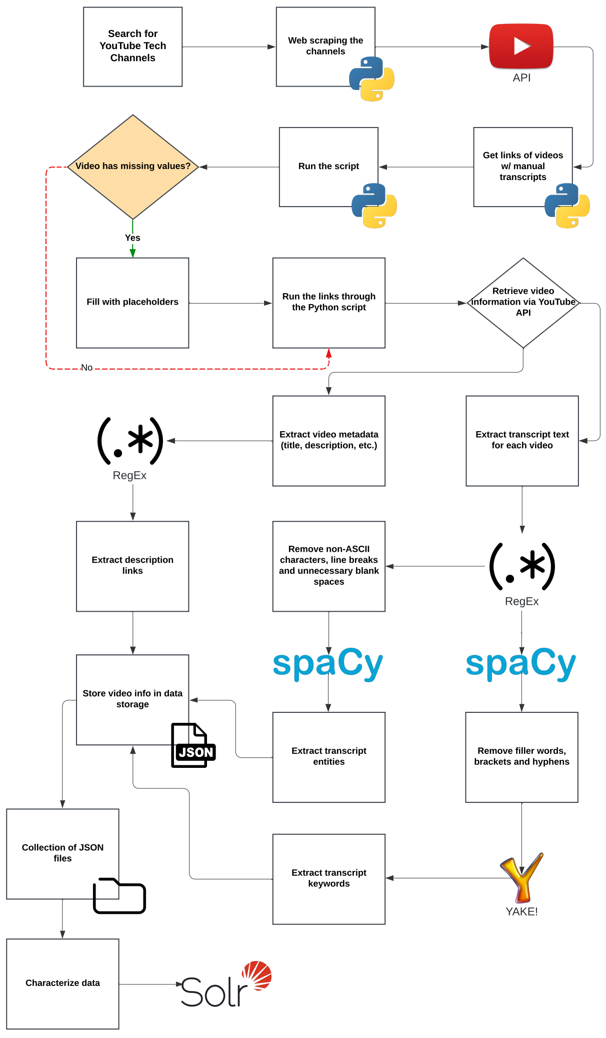

Search for YouTube Tech Channels
Web scraping the channels
API
Get links of videos w/ manual transcripts
Run the script
Video has missing values?
Yes
No
Fill with placeholders
Run the links through the Python script
Retrieve video information via YouTube API
Extract video metadata (title, description, etc.)
Extract transcript text for each video
(.*)
RegEx
Extract description links
Remove non-ASCII characters, line breaks and unnecessary blank spaces
(.*)
RegEx
spaCy
spaCy
Store video info in data storage
JSON
Extract transcript entities
Remove filler words, brackets and hyphens
Collection of JSON files
Extract transcript keywords
YAKE!
Characterize data
Solr

Fig. 1. Pipeline Diagram.

4 Final Dataset Description

The final dataset that was generated is stored in a folder named "video_collection". This folder contains all the respective JSON[Crockford 2002] files generated from the YouTube[Steve Chen and Karim 2005] videos with manual transcripts they had available.

In this system a document is defined as a single video, with each one represented by a JSON[Crockford 2002] file in the dataset.

Each JSON[Crockford 2002] file contains a variety of attributes. The 'video_id' field is a unique ID for each video, obtained from its URL path. The 'title' refers to the video's title, and 'channel' refers to the name of the YouTube[Steve Chen and Karim 2005] channel that published the video. The 'channel_id' field is a unique ID for each channel, obtained from the channel URL path, and 'subscriber_count' is the number of subscribers the channel has. The 'upload_date' field indicates the date when the video was uploaded. The 'video_url' field provides the full URL to the video.

The 'category' field contains the category, obtained from the API, that a specific video falls under. The 'tags' field contains the tags associated with each video. The dataset also registers the 'views' and 'likes' the video had when the metadata was collected, along with the 'comments_count.'

The 'description' field contains the full-text description provided with the video and the "description_links' field contains all URLs present in the video's description. The 'transcript' field contains the full transcript of the video, converting the spoken context into written text. Additionally, the 'transcript_keywords' are the key phrases or words extracted from the transcript using the YAKE![TEC 2018] keyword extraction tool. Finally, the 'transcript_entitiy_types" and "transcript_entity_values" fields contain the extracted entities, separated into their actual values, and the Spacy identified types.

This dataset is organized in a way that simplifies both the analysis of the data and its later retrieval.

Table 1. Field Types.

Field Name	Type
`video_id`	String
`title`	String
`channel`	String
`channel_id`	String
`subscriber_count`	Integer
`upload_date`	Date
`video_url`	String
`category`	String
`tags`	List of strings
`views`	Integer
`likes`	Integer
`comments_count`	Integer
`description`	String
`description_links`	List of strings
`transcript`	String
`transcript_keywords`	List of strings
`transcript_entity_values`	List of tuples of strings
`transcript_entity_types`	List of tuples of strings

5 Conceptual Model

The conceptual model below provides a detailed view of our product review system, which tracks tech product review videos, channels, tags, and associated metadata like keywords and entities. Since the system revolves around YouTube[Steve Chen and Karim 2005] videos, the main table stores video details such as title, URL, date, views, likes, and comment count.

Each table includes a unique identifier to maintain consistency and referential integrity.

Videos can have multiple tags, represented by a many-to-one relationship with the Tag table, which stores tag identifiers and text. A YouTube[Steve Chen and Karim 2005] channel can post numerous videos, shown through a one-to-many relationship with the Channel table, which includes attributes like channel name and subscriber count.

Each video also has a one-to-one relationship with a Caption table, which stores the video's description text. Descriptions often contain important metadata, like links to social media, which are extracted through NLP [IBM [n. d.]b] processes and stored in the Description Links table. Since links may appear in multiple videos, a many-to-many relationship connects the Video and Description Links tables.

To identify the product being reviewed, video transcripts are retrieved and stored in the Transcript table, with a one-to-one relationship with the Video table. The transcript is processed for keyword extraction and named entity recognition [IBM [n. d.]a](NER), with the resulting keywords and entities stored in separate tables, both having many-to-many relationships with the Transcript table.

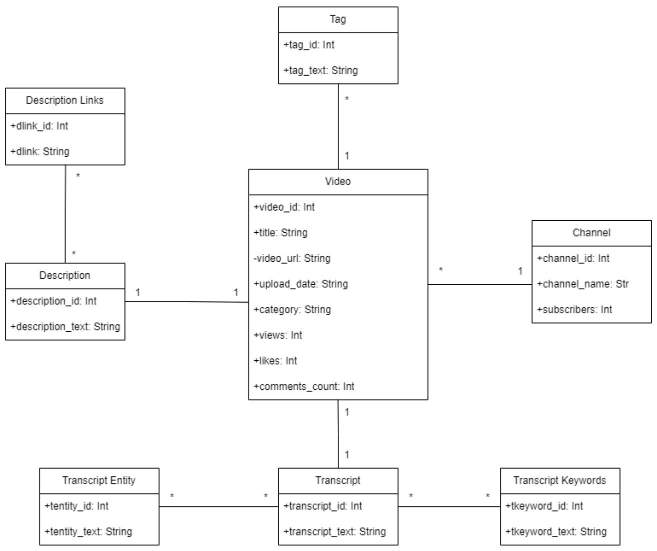


Fig. 2. Conceptual Model.

6 Data characterization

This section provides an overview of our dataset, presenting statistics, channel distributions, and descriptive analyses to understand its data.

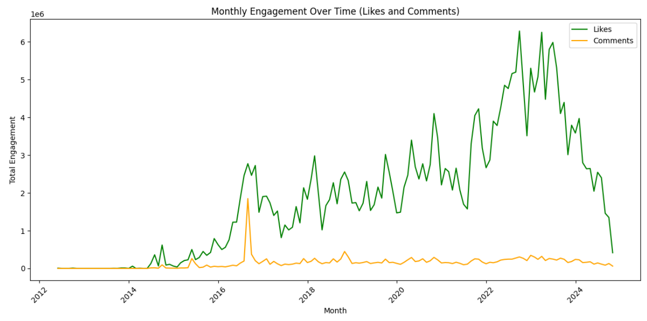


Fig. 3. Monthly Engagement Over Time.

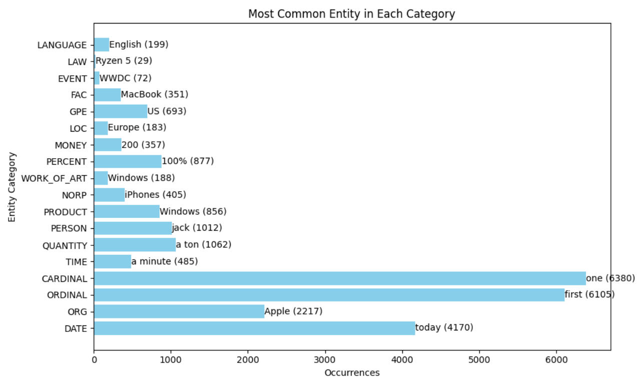


Fig. 4. Most Common Entity Per Category.

6.1 Statistics

In terms of descriptive statistics, the group decided to calculate some general metrics for the entire collection, such as total counts, means, standard deviations, and other relevant summaries.

- Total number of documents: 8346
- Largest document: Video with ID uLPBC2qJgvA
- Smallest document: Video with ID 7V_XEgSjrM0
- Earliest upload date: 2012-05-15
- Latest upload date: 2024-10-15
- Number of unique channels: 24

In the Appendix is the complete summary with tables (Tables 2 and 3) and additional statistics (Figures 5, 6, and 7), including the videos per channel table and descriptive statistics for numeric fields.

6.2 Field or value comparisons

An analysis of the relationships between subscriber metrics such as likes, comments, views, and subscriber counts can be seen above. Figure 3 shows the monthly engagement metrics, showing how interactions have changed over the dataset's time frame.

The remaining charts (Figures 8, 9, and 10), which are included in the Appendix, provide further insights into the relationships between various fields and their values, enhancing our understanding of the dataset's dynamics.

6.3 Text analysis

This section focuses on the analysis of textual data from the dataset, specifically on the most common entities. This analysis, shown in Figure 4, reveals the frequency of various entities present in the video descriptions and transcripts per category. At the end of the Summary Statistics in the Appendix, a textual representation of this chart is also provided.

7 Possible Search Scenarios

For our project aimed at helping users find relevant videos, understanding various search scenarios is crucial. Users normally have specific goals in mind when searching for content. Here are some possible scenarios:

- As a user, I want to be able to find videos by searching for specific keywords, for instance, "battery life".
- As a user, I want to be able to access the transcript of the video in order to quickly inspect specific information about the product without needing to watch the entire video.
- As a company, we want to identify YouTube[Steve Chen and Karim 2005] channels with high engagement metrics, such as views, likes, and comments, for potential sponsors and collaborators.
- As a researcher, I want to analyse trends over time by examining the number of views and likes for specific tech products, for instance, smartphones, laptops, etc.
- As a user, I am interested in finding videos reviewing new technologies, such as product releases.
- As an AI developer, I want to explore ethical concerns regarding Artificial Intelligence.
- As a PC building newbie, I want to find Step-by-Step guides to build a Gaming PC.
- As a user seeking longevity, I want to find durability tests in laptop reviews.

8 Indexing Process

This section describes the entire process and decision-making that went into devising the indexing of documents for our improved evaluation setup (Setup 2). Setup 1 relies on the automated indexing performed by Solr when populating the core without defining a custom schema.

8.1 Indexing Tool

Our collection is indexed using Apache Solr [Seeley 2004]. This decision was made due to the open-source nature of the platform, as well as its robust set of capabilities and popularity. The abundance of documentation and online resources available made it easier to learn how to use, and it accommodated all of our indexing and retrieval needs.

8.2 Indexing setup

A Makefile [Foundation 2024] was designed, with several commands to facilitate the entire indexing, retrieval, and evaluation process. Those relevant for indexing were:

- “make up” - Uses a docker-compose file to set up and start the Docker daemon, running the Solr image.
- “make update_configs” - Copies a custom “stopwords_en.txt” and “synonyms.txt” files to Solr’s configuration directory.
- “make schema” - Uploads the schema field types, and fields through the API, with 2 sequential curl [Stenberg 1998] commands.
- “make populate” - Posts the document collection to Solr, which indexes it based on the schema.
- "make get_embeddings" - Calls a script that will run an AI model throughout our collection to generate vector embeddings of the data for semantic search usage.
- "make populate_semantic" - Posts the document collection to Solr with the vector embeddings obtained from the AI model.
- "make down" - Destroys the created container.

8.3 Indexed Fields

The translation between our documents’ Json fields, to Solr indexed ones, is quite straightforward. All of them are indexable and don’t require any kind of modification to do so, however, the group decided to discard any, that it knew would not be useful for the information needs it is currently working with, resulting in indexing only fields with relevant textual information related with the content of the video itself. Therefore, the list of indexed fields is: “title”, “tags”, “description”, “transcript”, “transcript_entity_values”, “transcript_entity_types” and “transcript_keywords”.

8.4 Schema

With the ability to search over a video’s transcript being the highlight of the search system, and with the characteristics natural to spoken text, that usually makes it harder to search over, a special field type, named “text_transcript” was designed, to target specifically the indexing of the “transcript” field. It utilizes the standard Solr tokenizer, which was deemed more appropriate for the nuances of the transcript when comparing it with the whitespace tokenizer. Four filters were applied:

- solr.LowerCaseFilterFactory - Turns all tokens lowercase.
- solr.StopFilterFactory - Removes tokens that match the stopwords listed in a custom stopwords_en.txt file.
- solr.SynonymFilterFactory - Maps tokens to the synonyms listed in a custom synonyms.txt file. This file provides a comprehensive list of equivalent tech-related words, generated using ChatGPT [OpenAI [n. d.]] .
- solr.SnowballPorterFilterFactory for English - Applies stemming to the tokens, reducing them to their base English form. It is a multilingual extension of the base PorterStem filter, but when defined for the English language, resulted in an observable performance increase in comparison.

The set of filters was obtained through experimentation, verifying which ones improved the overall performance of the search system. Two more field types were used in our schema: “text_en”, and “string”, both provided by Solr. “text_en” also utilizes the Standard Tokenizer, and applies some filters that work well with the English language, and it was used to index the fields: “title”, “tags” and “description”. “string” keeps a piece of text as is, and was therefore applied to fields where it made sense searching for exact matches, without text transformations, those being: “transcript_entity_values”, “transcript_entity_types” and “transcript_keywords”. The fields “tags”, “transcript_entity_types” and “transcript|_keywords”, were also marked as *multiValued*, since they stored lists of elements.

9 Retrieval Process

For the baseline system, the queries were run with no boosters, just the query string and the respective fields that needed to be searched. As for the second system, stopword and synonym files were copied to the Solr [Seeley 2004] Docker[Hykes 2013] container, a custom schema was uploaded to the Solr[Seeley 2004] server, and boosters were used for the queries. The group felt that the most important boosters to use were field boosts because the database has some fields, like *transcript* and *title*, that are of more importance to the end user than, for example, *description*, where the majority of the information is social media links and rarely have actual product content on them. Therefore, a bigger weight was given to *title*, followed by *transcript*, and left *description*, *transcript_keywords*, *transcript_entities* with no weight, because the keywords and entities are already part of the transcript.

For the final system, the main idea explored in order to improve our search engine was semantic search.

When running the semantic search tutorial provided with the *TechSpotter* dataset, the response included documents that were considered relevant after reading their content by the group, that were not included in our *qrels* text file. This led to an hypothesis. What if the semantic search could be implemented alongside the lexical search and provide documents the lexical could not?

After some research about the topic of merging the lexical and semantic search, three possible solutions were found:

- Filter Query
- Query Re-ranking
- Hybrid Search

Filter Query and Query Re-ranking led to better results than system 2, but Hybrid Search was not only the option that made more sense in its definition, it was also the one that had the better metrics out of the three.

Filtering the queries meant obtaining the results of the lexical search and then filtering the response with the semantic search model, while re-ranking the queries also meant obtaining the documents from the lexical search and ordering the response by scores provided by the semantic search model.

As for the Hybrid Search, both searches were executed and then the retrieval responses are merged into the same document, organizing it by the scores provided by each. The main problems with this method are the time it takes to do two searches instead of one, and choosing which search has more impact on the metrics. In our case, the lexical search yielded scores that ranged from 0 to 50, while the semantic scores only varied between 0 and 1. This means that a normalization of the scoring process must be done before attempting to merge the documents. By dividing the scores of all documents retrieved in both searches by the maximum score value found in the responses, the results have both been normalized to the same range, and they can now be put together. Another important thing in the hybrid search method is attributing weights to each search score. At first, when the same document was found in both retrieval responses, a normal mean would be applied so that both searches had the same weight. Still, the group decided to play around with the weights to find the best performance it could metrics-wise, giving the semantic search a slightly bigger impact on the results.

10 Evaluation and Demo

In this section a description of how the group found and evaluated relevant documents is provided, as well as the difference between system setups, the defined information needs and the analysis of their results.

10.1 Document Relevance Judgement

To identify our collection of relevant documents, in a systematic and manageable way, 4 different Solr configurations were created:

10.1.1 Configuration A.

- Query Parser: edismax
- Fields to Search (with boosts): transcriptˆ3 transcript_entitiesˆ2 transcript_keywordsˆ2 titleˆ2 descriptionˆ1 tagsˆ1
- Additional Parameters:
  - Phrase Fields (pf): transcriptˆ2
  - Phrase Slop (ps): 2

10.1.2 Configuration B.

- Query Parser: edismax
- Fields to Search (with boosts): titleˆ3 tagsˆ2 transcriptˆ2 descriptionˆ1
- Additional Parameters:
  - Minimum Match (mm): 75
  - Synonyms enabled for the transcript

10.1.3 Configuration C.

- Query Parser: standard
- Fields to Search: Default (relies on the df parameter)
- Default Field (df): transcript
- Additional Parameters:
  - Explicit use of the boolean operators AND/OR

10.1.4 Configuration D.

Same as A, but using boolean operators.

Utilizing Solr's[Seeley 2004] web interface, each query was ran with the four configurations, storing the top 30 results at each point. We then eliminated any duplicate documents, and manually evaluated the remaining ones, either by reading the transcript that was retrieved, or watching the corresponding video, retaining only those where the content for that particular query was considered relevant. Relevant document ids are stored per row beside the id (1, 2, or 3) of the query it is associated with (e.g. "1 DRe2TCx-fao"), in a file named "qrels.txt", which is then used in our evaluation pipeline, when running the TREC[NIST 1992] evaluation tool, to compare it with the documents retrieved by Solr[Seeley 2004], during the search.

Semantic search was later used to find more relevant documents for all queries, and then analyzing the corresponding video transcripts and, in some cases, watching the videos themselves. Following this meticulous review, the *qrels* text file was expanded.

10.2 System Setups

Three distinct Solr[Seeley 2004] setups were used for evaluation. Their characteristics were devised, to make the performance increase observed with the implementation of the schema, and the use of field boosts for querying, more noticeable.

- Setup 1
  - Does not define a custom schema, meaning it relies only on Solr's[Seeley 2004] automated indexing process.
  - Queries the fields: "transcript", "transcript_keywords", "transcript_entity_values", "title" and "description".
- Setup 2
  - Utilizes the custom schema described in the previous section.
  - Queries the same fields as in setup 1, boosting the "transcript" field by 2 and "title" by 3 (boosts that yielded the best overall performance increase).
- Setup 3
  - Utilizes the custom schema used in Setup 2.
  - Queries the same fields as in setup 1 and 2, boosting the "transcript" and "title" field by 1.5
  - Utilizes Hybrid Search instead of Lexical-only.

10.3 Information Needs

Choosing quality information needs, that accurately represented and differentiated our search system, proved more difficult than initially intended. It was easy to fall into the trap of defining information needs that closely overlapped with a simple YouTube[Steve Chen and Karim 2005] search, but the group wanted something more targeted at the capabilities our search system provides. The biggest differentiating factor is the ability to utilize the video transcripts, and therefore information needs were formulated based on interesting technological information that a user would want to find over the actual discourse of the videos. Below are the 3 information needs created, alongside their corresponding query strings, which were used for Solr[Seeley 2004] retrieval.

10.3.1 AI Ethical Concerns.

- Information Need: As an AI Developer, I want to explore the ethical concerns regarding Artificial Intelligence.
- Query:
  - *q:* "ethics artificial intelligence debate discussion concerns news"
  - *fields:* "id, score"
  - *qf:* "transcript, transcript_entities, transcript_keywords, title, description"
  - *rows:* 50

Table 2. Average Precision comparison for Query 1 between System 1, 2 and 3.

Metric	System 1	System 2	System 3
AvP	0.3112	0.7971	0.8015

The average precision metric allows us to determine whether a system is compiling the relevant documents at the top of the search's response.

The average precision for System 2 shows a substantial improvement over System 1. This increase highlights the strength of field boosting and synonym-based query expansion in retrieving relevant documents.

Artificial Intelligence is a highly prevalent topic, but ethical concerns often appear as secondary points within broader discussions about AI. In such cases, simple lexical matching struggles to retrieve documents where "ethical concerns" are not the primary focus. By boosting the transcript field, System 2 ensures that content covering AI ethics, even if embedded within longer conversations, ranks higher in the results.

Furthermore, the availability of a robust synonym file for AI-related terms significantly contributes to the improved precision by matching related phrases that may not directly appear in the query. This highlights the importance of both context analysis and query expansion in handling nuanced and secondary topics effectively.

Surprisingly, the hybrid search in System 3 did not provide a significant gain in precision for this query. The group had expected that the context analysis of 'ethical concerns' through semantic search would help retrieve additional relevant documents, but this was not observed.

10.3.2 *Building a gaming PC with guides.*

- Information Need: As a PC building newbie, I want to find Step-by-Step guides to build a gaming PC
- Query:
  - *q:* "how build a gaming pc tutorial"
  - *fields:* "id, score"
  - *qf:* "transcript, transcript_entities, transcript_keywords, title, description"
  - *rows:* 38

Table 3. Average Precision comparison for Query 2 between System 1, 2 and 3.

Metric	System 1	System 2	System 3
AvP	0.5682	0.6184	0.7834

Although System 1 already had a strong performance, mainly due to the popular and easily searchable topic of gaming PC builds, introducing field-specific boosts and a custom schema enhances the relevance of results, especially when specific fields are closely aligned with user intent.

Hybrid search significantly outperforms keyword-only methods by capturing the semantic meaning of the query, which is crucial for long-tail or natural-language queries. While field boosts and schema adjustments are valuable, they alone cannot match the sophistication of a hybrid approach.

10.3.3 *Durable laptops.*

- Information Need: As a user seeking longevity, I want to find durability tests in laptop reviews.
- Query:
  - *q:* "durability test laptops laptop"
  - *fields:* "id, score"
  - *qf:* "transcript, transcript_entities, transcript_keywords, title, description"
  - `rows:` 18

Table 4. Average Precision comparison for Query 3 between System 1, 2 and 3

Metric	System 1	System 2	System 3
AvP	0.3373	0.3056	0.6597

The final search scenario was not very precise, either for the baseline or the complex system. The core issue appears to stem from the generic nature of the term "laptop" and its widespread usage across the dataset. Since "laptop" and its synonyms occur frequently in numerous documents, many irrelevant results are retrieved, reducing precision.

In this case, a decrease in AvP is observed from System 1 to System 2. This decline is likely due to the boosted fields in System 2 not aligning well with the key fields containing relevant information for this query. The query context, centered around 'durability tests' and 'laptops', plays a critical role here. System 3 shows a significant improvement because it effectively incorporates semantic search, enabling it to identify content that discusses durability tests even when the exact query terms are not explicitly present. This highlights the importance of query context analysis and semantic understanding for improving retrieval performance in cases where lexical matching alone falls short.

10.4 Results Analysis

The group ran all queries through TREC [NIST 1992] evaluation and compiled the metrics of all queries for each system.

The progression from the first precision-recall curve to the second and third curves shows a clear improvement in both the Area Under the Curve (AUC) and Mean Average Precision (MAP). Each system demonstrates a marked enhancement in ranking relevant documents higher and irrelevant ones lower.

System 1's lower AUC score reflects its struggle to effectively rank relevant documents at the top and irrelevant ones at the bottom. As a result, its performance in precision and recall is limited, especially across varying thresholds.

System 2 demonstrates an improvement, with the AUC increasing. This improvement indicates better ranking capabilities compared to System 1. Additionally, the MAP score rises, highlighting System 2's ability to retrieve a greater proportion of relevant documents in higher-ranked positions on average.

System 3, utilizing Hybrid Search, further enhances these metrics, achieving an higher AUC and MAP. This substantial improvement

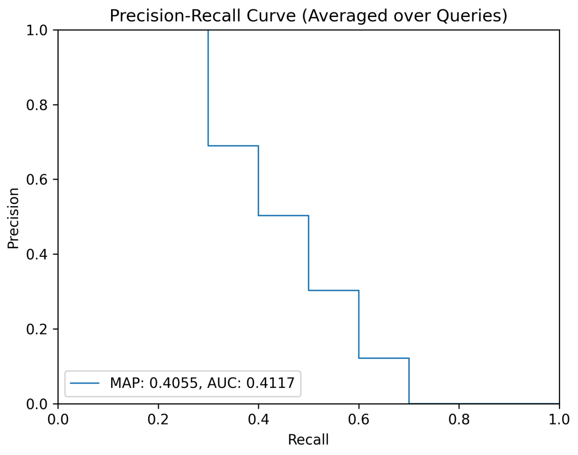


Fig. 5. Precision-Recall Curve M3 - System 1.

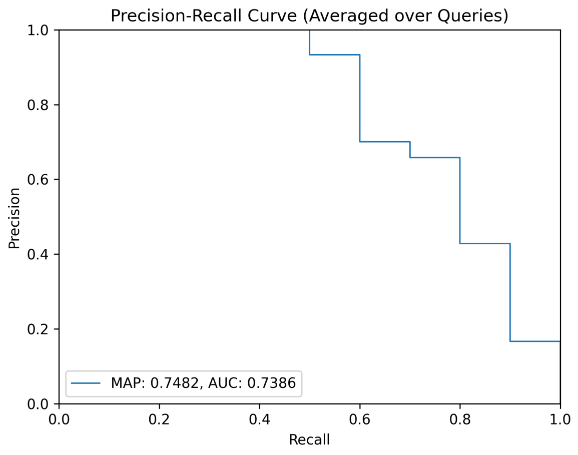


Fig. 7. Precision-Recall Curve - System 3.

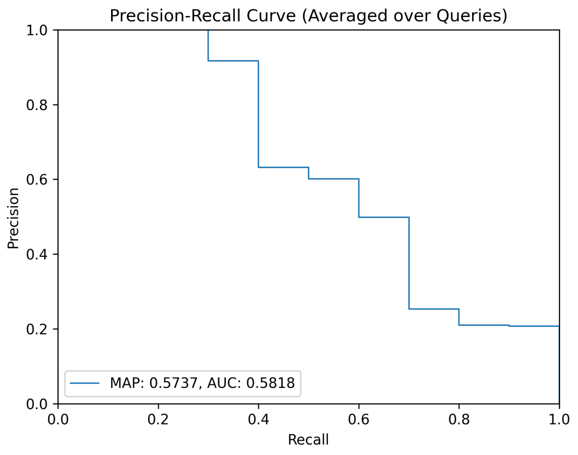


Fig. 6. Precision-Recall Curve M3 - System 2.

reflects System 3's advanced semantic search capabilities, enabling it to better analyze context and retrieve highly relevant documents. The increased MAP indicates superior ranking precision, meeting expectations for a system that integrates semantic search techniques.

Overall, the data confirms a consistent trend of improvement from System 1 to System 3, with each iteration achieving better precision, recall, and ranking efficiency.

The P@K metrics provide valuable insight into how well each system ranks the most relevant documents within the top K results, supplementing other performance indicators like MAP and AUC. The comparison of P@10 and P@20 reveals distinct trends across the three systems, illustrating the improvements introduced by the more complex configurations.

At P@10, the baseline system retrieves relevant documents with limited effectiveness within the top 10 results. In contrast, System 2 shows improvement, demonstrating that field boosts and schema customizations help prioritize relevant documents more effectively at higher rankings. System 3 further advances precision, highlighting the strength of combining hybrid search techniques with booster tweaks. These enhancements ensure that the system not only captures relevance but also leverages semantic understanding to better match the user's intent.

At P@20, System 1 struggles further, reflecting its inability to maintain relevance as the cut-off expands. System 2 performs much better at this level, which indicates that schema adjustments and field boosts significantly improve the system's ability to sustain relevance as more results are considered. System 3 outperforms both, confirming its superior ranking consistency. The hybrid approach used in System 3 proves effective at retrieving not only the most relevant results but also contextually appropriate ones, even at a larger cut-off.

Overall, the data underscores a clear progression of performance from System 1 to System 3. While System 2 successfully addresses the milestone objective of improving over the baseline by maintaining a steady precision across both cut-offs, System 3 demonstrates the most robust performance. The consistent improvement in precision, particularly at P@20, highlights the value of hybrid search and booster tweaks in delivering a highly effective ranking system. System 3 not only excels at identifying the most relevant documents but also ensures that relevance is sustained as the scope of results expands, offering a more comprehensive and refined solution for complex search queries.

Table 5. Performance comparison between System 1, 2 and 3 for P@10 and P@20.

Metric	**System 1**	**System 2**	**System 3**
P@10	0.3	0.7	0.8333
P@20	0.2	0.5333	0.6333

11 Experiments

Throughout this last milestone, there have been countless attempts to improve the metrics of the search engine. Unfortunately, not all of those tries contributed in a positive manner towards the desired goals.

Among those failed attempts are the tests with the following parameters:

- **Tie-breaker** (tie)
  This parameter determines the degree of flexibility allowed in word proximity within phrase queries. A higher "slop" value permits greater variation in the order and spacing of terms while still considering the phrase a match. This can enhance the adaptability of the query, allowing it to account for minor differences in word arrangement or spacing within the text.
  It was tested multiple times across a range of values from 0.1 to 0.5.
- **Minimum Match** (mm)
  This parameter specifies the minimum number of query clauses (terms or phrases) that must match for a document to be considered a match. It allows for partial matches in multi-term queries while controlling the strictness of the matching criteria.
  The following values were tested:
  - **50%**: At least half of the terms in the query must match.
  - **75%**: At least three-quarters of the terms in the query must match.
  - **2<75%**: If the query contains more than two terms, at least three-quarters of the terms must match.
  - **3<50%**: If the query contains more than three terms, at least half of the terms must match.
- **Phrase Slop** (ps)
  This parameter determines the allowable degree of word proximity flexibility within phrase queries. It controls how far apart the terms in a phrase can appear while still being considered a match. A higher slop value provides more flexibility in the order and spacing of terms, enabling looser phrase matching.
  The following values were tested:
  - 6: Minimal flexibility, requiring terms to be relatively close to each other in the text.
  - 8: Moderate flexibility, allowing for slightly larger gaps between terms.
  - 10: A balance between strict and loose matching, accommodating moderate variation in spacing.
  - 12: High flexibility, suitable for cases where terms may appear in varying orders or with significant gaps.
  - 15: Maximum flexibility tested, allowing for substantial variation in both term order and spacing.

  Lower values (e.g., 6) ensure stricter phrase matching, favoring precise results where terms closely match the specified order and spacing and higher values (e.g., 12 or 15) increase tolerance for term reordering and spacing differences, retrieving a broader range of results but potentially at the expense of precision.
- **Phrase Fields** (pf)
  This parameter increases the relevance of documents where the query terms appear as an exact phrase within specified fields. By assigning a boost value to each field, this parameter prioritizes matches in those fields, making them more influential in the ranking of search results.
  It was tested with the following field boosts: transcript$\hat{3}$ title$\hat{2}$.

Since none of the parameters mentioned above were able to improve the metrics of the search engine, the group decided not to include them in the final version of the project.

The group also conducted experiments with the Solr[Seeley 2004] feature named **More Like This** (MLT). This is a feature designed to find documents similar to a given reference document. It analyzes the content of the input document to identify key terms and uses those terms to retrieve other documents with similar content. This approach can be highly effective in use cases where a representative document encapsulates the characteristics of a desired result set.

However, during the experiments, the MLT query was not suitable for this specific use case, since the search queries query fields that contain large text blocks, which vary significantly from video to video (e.g. transcript). This variability makes it challenging to identify a single representative document that can reliably retrieve relevant results for all queries. As a result, the MLT feature failed to consistently improve the relevance of the search results and was excluded from the final implementation.

12 Search User Interface

The search user interface was designed to provide a seamless and intuitive experience for users interacting with the *TechSpotter* search engine. It uses the Flask[Ronacher 2010] API to handle search queries. Results are displayed as interactive video cards that, when clicked, open a detailed page containing key information about the video. The key features are:

- **Dynamic Search**: Users can input queries, and the system fetches results using the backend Flask[Ronacher 2010] API.
- **User Interaction**: Clicking on a video card leads to its detailed page while tracking user behavior to improve future search rankings.

It is possible to visualize the Demo of the project here.

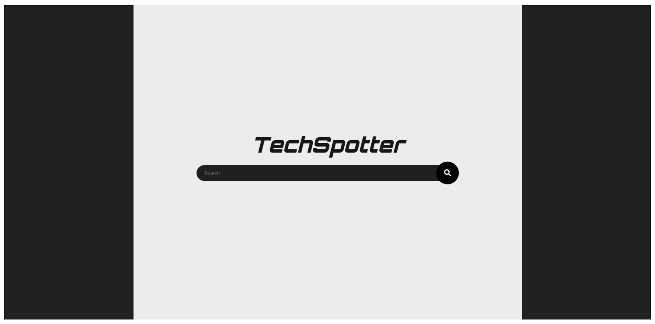


Fig. 8. Search User Interface.

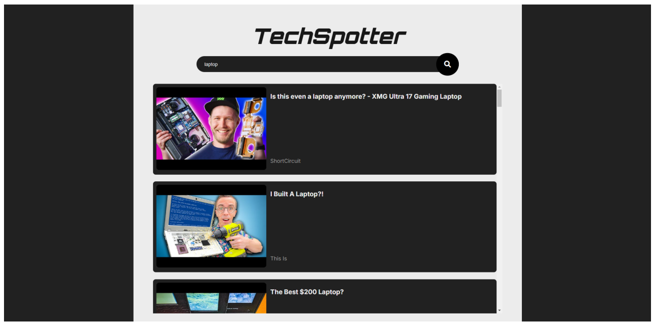
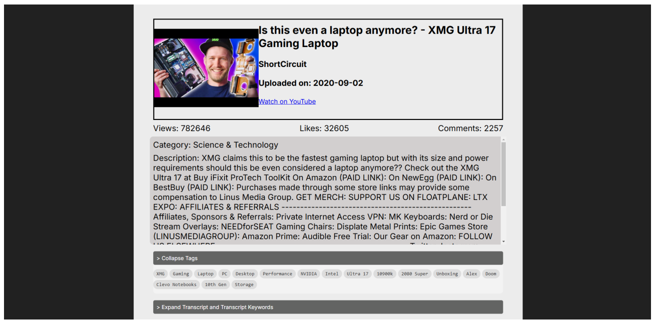


Fig. 9. Search User Interface - Search Results.



Is this even a laptop anymore? - XMG Ultra 17 Gaming Laptop

ShortCircuit

Uploaded on: 2020-09-02

Watch on YouTube

Views: 782646 Likes: 32605 Comments: 2257

Category: Science & Technology

Description: XMG claims this to be the fastest gaming laptop but with its size and power requirements should this be even considered a laptop anymore?? Check out the XMG Ultra 17 at Buy iFixit ProTech ToolKit On Amazon (PAID LINK): On NewEgg (PAID LINK): On BestBuy (PAID LINK): Purchases made through some store links may provide some compensation to Linus Media Group. GET MERCH: SUPPORT US ON FLOATPLANE: LTX EXPO: AFFILIATES & REFERRALS -- Affiliates, Sponsors & Referrals: Private Internet Access VPN: MK Keyboards: Nerd or Die Stream Overlays: NEEDforSEAT Gaming Chairs: Displate Metal Prints: Epic Games Store (LINUSMEDIAGROUP): Amazon Prime: Audible Free Trial: Our Gear on Amazon: FOLLOW

Fig. 10. Search User Interface - Video Detailed.

12.1 New Signals

For the User Interface, it is common for search engines to update the results based on the user's behavior. The easiest change to spot is by clicking a link in a Google[Larry Page 1998] search, where the link is re-ranked higher after being clicked multiple times. This was implemented on the User Interface and it can be seen in the video already linked above.

When a user clicks on an already displayed video card, the system, before opening the specific document webpage, will store in a JSON[Crockford 2002] file the number of clicks that document has had so far. When the user inputs a query string where that document is shown, a recalculation of the score will be executed based on the number of clicks, and the new score might be high enough to show a different order of documents compared to the previous query.

13 Conclusion

In conclusion, the *TechSpotter* project successfully developed a custom search engine for YouTube[Steve Chen and Karim 2005] tech videos, delivering improved performance through systematic enhancements and iterative testing.

During the Data Preparation phase, the team successfully collected and cleaned a large amount of data to create a complete dataset with all essential metadata attributes and transcription information. This dataset was structured to support effective analysis and retrieval of relevant video content. The creation of keyword fields was particularly important for enhancing search precision and relevance.

In the Information Retrieval phase, the team implemented indexing and retrieval processes using Apache Solr[Seeley 2004], incorporating a custom schema and field boosts to improve query performance. This resulted in System 2, which prioritized relevant fields, such as video titles and transcripts. The team manually evaluated the results and kept a record of the relevant documents to ensure accuracy and reliability.

Now, in Improvement phase, further enhancements were made to the search engine, resulting in System 3. A hybrid search approach was introduced, combining the strengths of both lexical and semantic search methods. This approach improved the retrieval of contextually relevant documents, addressing the limitations of purely lexical matching. Additionally, user behavior signals were integrated into the user interface, enabling dynamic re-ranking of results based on click interactions.

The evaluation of the final system demonstrated significant improvements in precision-recall metrics, showcasing its ability to effectively identify and rank relevant content within large-scale video datasets.

14 Future Work

However, there is still room for improvement. Future work could include further optimization of the hybrid search approach, and exploration of other ranking methods such as machine learning-based re-ranking or query expansion techniques, and enhancements to semantic search capabilities. Expanding the user interface to include advanced filtering options and improving its performance for larger datasets could also significantly enhance the user experience.

With its robust design, *TechSpotter* sets the groundwork for scalable and adaptive search engines tailored to domain-specific video collections, providing an effective tool for exploring vast amounts of video content.

References

Douglas Crockford. 2002. JavaScript Object Notation. https://www.json.org/json-en.html.
ExplosionAI. 2016. spaCy. https://spacy.io.
Free Software Foundation. 2024. Makefile. https://www.gnu.org/software/make/.
Python Software Foundation. 2001. Python Programming Language. https://www.python.org/.
Google. 2007. Youtube Data API. https://developers.google.com/youtube/v3?hl=pt-br.
Solomon Hykes. 2013. Docker. https://www.docker.com.
IBM. [n. d.]a. Named Entity Recognition. https://www.ibm.com/topics/named-entity-recognition.
IBM. [n. d.]b. Natural Language Processing. https://www.ibm.com/topics/natural-language-processing.
Sergey Brin Larry Page. 1998. Google. https://www.google.com.
NIST. 1992. TREC evaluation. https://github.com/usnistgov/trec_eval.
OpenAI. [n. d.]. ChatGPT. https://openai.com/chatgpt.
Armin Ronacher. 2010. Flask. https://flask.palletsprojects.com/en/stable/.
Yonik Seeley. 2004. Solr. https://solr.apache.org.
Daniel Stenberg. 1998. curl. https://curl.se/.
Chad Hurley Steve Chen and Jawed Karim. 2005. YouTube. https://www.youtube.com.
INESC TEC. 2018. Yet Another Keyword Extractor. https://repositorio.inesctec.pt/server/api/core/bitstreams/ef121a01-a0a6-4be8-945d-3324a58fc944/content.
Wikipedia. [n. d.]. Regular Expression. https://en.wikipedia.org/wiki/Regular_expression.

A Data Characterization Charts

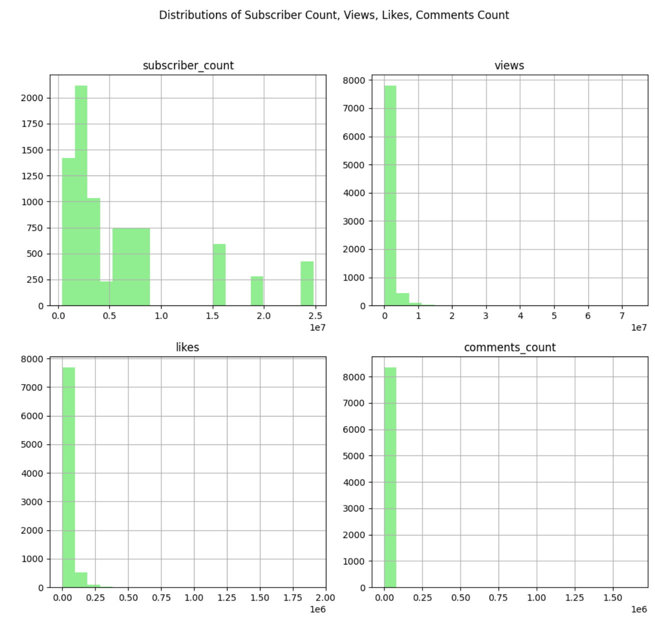


Fig. 11. Numeric Distributions.

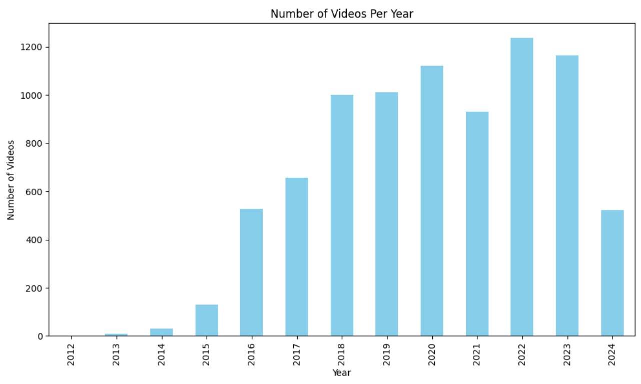


Fig. 13. Videos per Year.

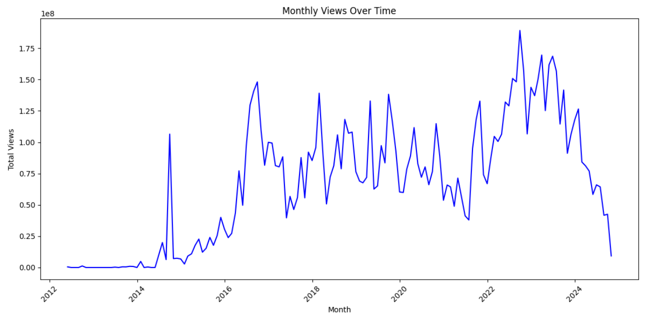


Fig. 14. Monthly Views Over Time.

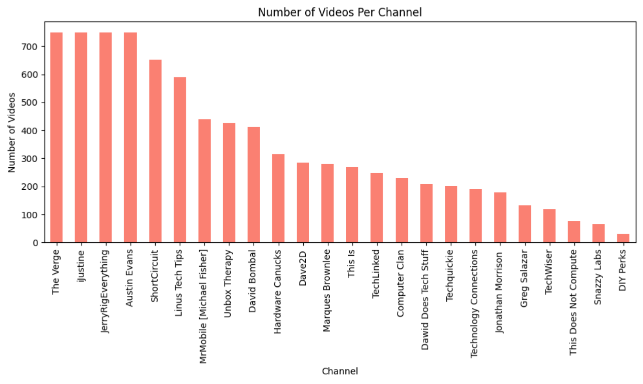


Fig. 12. Videos per Channel.

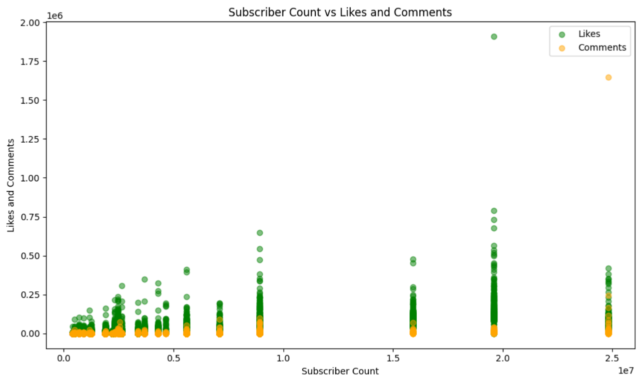


Fig. 15. Subscribers vs Likes and Comments.

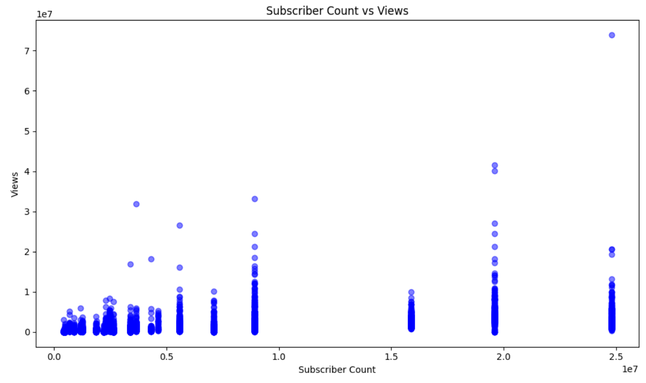


Fig. 16. Subscribers vs Views.

B Statistics Summary

Total number of documents: 8346
Largest document: https://www.youtube.com/watch?v=uLPBC2qJgvA
Smallest document: https://www.youtube.com/watch?v=7V_XEgSjrM0
Earliest upload date: 2012-05-15
Latest upload date: 2024-10-15
Number of unique channels: 24

Videos per channel:

Table 6. Videos per Channel.

Channel	Videos
The Verge	750
iJustine	750
JerryRigEverything	750
Austin Evans	749
ShortCircuit	653
Linus Tech Tips	589
MrMobile [Michael Fisher]	440
Unbox Therapy	425
David Bombal	411
Hardware Canucks	314
Dave2D	284
Marques Brownlee	280
This Is	269
TechLinked	248
Computer Clan	229
Dawid Does Tech Stuff	208
Techquickie	202
Technology Connections	190
Jonathan Morrison	179
Greg Salazar	133
TechWiser	119
This Does Not Compute	78
Snazzy Labs	65
DIY Perks	31

Descriptive statistics for numeric fields:

Table 7. Descriptive Statistics for Numeric Fields.

Field	Subscriber Count	Views	Likes	Comments Count
Count	8,346	8,346	8,346	8,346
Mean	6,252,080	1,219,511	34,754.74	2,638.16
Std Dev	6,397,027	2,115,895	53,595.82	18,868.69
Min	414,000	0	0	0
25%	2,200,000	245,514.5	7,351.25	525
50%	3,390,000	569,388	18,084.50	1,151
75%	7,100,000	1,447,309	42,131.25	2,638
95%	24,800,000	4,328,769	117,591.80	7,830.25
Max	24,800,000	73,893,950	1,907,538	1,647,222

Most Common Entity in Each Category:

- DATE: today (4170 occurrences)
- ORG: Apple (2217 occurrences)
- ORDINAL: first (6105 occurrences)
- CARDINAL: one (6380 occurrences)
- TIME: a minute (485 occurrences)
- QUANTITY: a ton (1062 occurrences)
- PERSON: jack (1012 occurrences)
- PRODUCT: Windows (856 occurrences)
- NORP: iPhones (405 occurrences)
- WORK OF ART: Windows (188 occurrences)
- PERCENT: 100% (877 occurrences)
- MONEY: 200 (357 occurrences)
- LOC: Europe (183 occurrences)
- GPE: US (693 occurrences)
- FAC: MacBook (351 occurrences)
- EVENT: WWDC (72 occurrences)
- LAW: Ryzen 5 (29 occurrences)
- LANGUAGE: English (199 occurrences)