NYUCIN at the NTCIR-16 Dataset Search 2 Task

Levy Silva
Universidade Federal de Pernambuco
Recife, Brazil
lss9@cin.ufpe.br

Luciano Barbosa
Universidade Federal de Pernambuco
Recife, Brazil
luciano@cin.ufpe.br

Sonia Castelo
New York University
New York, USA
s.castelo@nyu.edu

Haoxiang Zhang
New York University
New York, USA
haoxiang.zhang@nyu.edu

Aécio Santos
New York University
New York, USA
aecio.santos@nyu.edu

Juliana Freire
New York University
New York, USA
juliana.freire@nyu.edu

ABSTRACT

In this paper, we describe the approach and results of the NYUCIN team in the NTCIR-16 conference. We participated in the Data Search 2 Task, which is a shared task on ad-hoc retrieval for governmental statistical data composed of multiple subtasks. We report our work on the two subtasks we participated in: the English IR Subtask and the UI Subtask. For the IR Subtask, we explored learning-to-rank approaches based on deep learning models. Given the limited training data available for this task, we employed a transfer learning method to train a deep neural network that learns how to match web tables and news articles using data available on the Web. The official evaluation shows that our approach attained the highest score among all submitted runs across all evaluation metrics. In particular, for the nDCG@5 measure, our score of 0.246 represents a 30% improvement compared to the second-best result in NTCIR-16 Data Search 2 Task. For the experimental UI Subtask, we performed two user studies to evaluate the design decisions and the usability of Auctus, a dataset search engine developed by our team.

KEYWORDS

dataset search, learning-to-rank, search user interfaces

TEAM NAME

NYUCIN

SUBTASKS

UI Subtask (English)
IR subtask (English)

1 INTRODUCTION

The large volumes of open data currently available on the Web open up new opportunities for progress in answering many important scientific, societal, and business questions. However, finding relevant data is still a difficult problem. While search engines have addressed this problem for Web documents, there are many new challenges involved in supporting the data retrieval for specific tasks.

The NTCIR-16 Data Search 2 Task is a shared task on ad-hoc retrieval for governmental statistical data, which aims at fostering research on open challenges on dataset search. This edition of the event included multiple sub-tasks and provided standard dataset collections collected from data published by the US government (data.gov) and Japanese government (e-Stat) [13].

This paper reports the work of the NYUCIN team[1] in the context of NTCIR-16. Specifically, we describe the approach and results for the dataset retrieval (IR Subtask) and the user interface (UI Subtask) challenges. We organized this paper into two main parts. In Section 2, we describe our efforts toward the IR Subtask, whereas in Section 3 we describe the work done for the UI Subtask. We conclude in Section 4.

2 IR SUBTASK

In this section, we detail our solution for the dataset search subtask. Given a query and dataset collection, the goal is to generate a ranked list of datasets. Towards this front, we assume a query as a sequence of a few words [12], and a dataset is composed of metadata (title and description), headings, and rows [13]. In what follows, we first overview our solution for this subtask in Section 2.1. Next, Section 2.2 describes the top-k algorithm, which focuses on the retrieval of candidate datasets, and we present the query-dataset matching model in Section 2.3. Lastly, Section 2.4 details the experimental setup, and we conclude this subtask by discussing the main results in Section 2.5.

2.1 Dataset Search Solution (Overview)

The task of dataset search is quite similar to the problem of web table retrieval [6]. Based on that, we assume the techniques designed for this problem could potentially be applied to dataset search. As in the previous work on table retrieval [23, 25], our approach uses an efficient top-k retrieval algorithm to retrieve a pool of initial candidate datasets which are then re-ranked using a more complex learning-to-rank model. In summary, we obtain the best-matching datasets according to the following three steps: (1) we index the corpus of datasets

[1]The NYUCIN team is a joint effort of members from the Centro de Informática (CIn) at Federal University of Pernambuco (UFPE), and the Tandon School of Engineering at New York University (NYU).

#	Classic Method	Dataset Aspect	Recall@50	Recall@100
1	BM25	Title	0.2338	0.2727
2	Cos(TF-IDF)	Title	0.2147	0.2601
3	**BM25**	**Description**	**0.4304**	**0.5014**
4	Cos(TF-IDF)	Description	0.2873	0.3787
5	BM25	Title, Description	0.3917	0.4581
6	Cos(TF-IDF)	Title, Description	0.2876	0.3794

Table 1: Results for the top-k retrieval algorithm on Elasticsearch. We use distinct indexing setups and search methods. The combination of BM25 over the dataset description achieves the best recall (bold values).

on Elasticsearch[2] by utilizing the dataset description; (2) we retrieve the top-100 datasets for each query by employing the BM25 algorithm (we use the default parameter setup and a multi-field retrieval approach); and (3) we re-rank the pool of candidates by using the query-dataset matching model to get the best-matching datasets.

2.2 Top-k Algorithm (Retrieval)

The goal of the top-k algorithm is to efficiently retrieve the highest number of relevant documents to make them available for the matching model. To select the best top-k retrieval algorithm, we ran a set of distinct indexing setups by using different combinations of dataset aspects (e.g., title, description) and ranking models, with the goal of obtaining the highest recall in the candidate set. We evaluate both Cos(TF-IDF) and BM25 algorithms at this stage (two classical search methods available in Elasticsearch). Table 1 shows the main results in terms of recall at cutoffs $k = \{50, 100\}$ for each methodology. In summary, when we use the combination of BM25 and dataset description, we achieve the best performance in terms of Recall@100 (0.5014). The other retrieval combinations attained lower results for Recall@100 even when we combine two dataset aspects. Based on these results, we use this approach (BM25 + Description) as the top-k algorithm to retrieve the top-100 candidate datasets that we use at the ranking step.

2.3 Matching Model (Ranking)

We now explain our model for query-dataset matching. Our approach learns a joint-representation from a $\langle$*query*, *dataset*$\rangle$ tuple and predicts a similarity score to rank the datasets. The model is presented in Figure 1. It produces two types of representations of the $\langle$*query*, *dataset*$\rangle$ input: one based on attention and another based on BERT architecture. For the attention branch, the input is the embedding of the words present in the query and dataset aspects, i.e., title and description. The network then applies a bidirectional recurrent network (bi-context block) on them to produce contextual vectors. These vectors are passed to the attention block that combines the aspects of the query and the dataset and outputs an attention vector for each one of them. In the other network's branch, we utilize BERT architecture to obtain another type of contextual representation based on self-attention from the $\langle$*query*, *dataset*$\rangle$ input. The outputs of two branches are concatenated and passed to an MLP, which computes the matching score on its top layer. Next, we detail each of these blocks as well as the training methodology.

Input Data Representation. We represent each aspect (title, description) as a sequence of words and utilize a word embedding approach to represent words as numerical vectors, similar to previous Neural Information Retrieval studies [11]. Specifically, we use a pre-trained FastText[3] dataset with 1 million word vectors trained on English Wikipedia pages to get the embedding for each word.

Bidirectional Context Block. We apply a Gated Recurrent Unit (GRU) to map each word to a fixed-length vector [8]. Furthermore, we consider a bidirectional GRU to obtain the representation of each word from both directions and use the concatenation of each hidden state as the final word representation. Recurrent Neural Networks (RNN) have been applied both in Information Retrieval tasks as well as in table retrieval approaches [25, 27].

Attention Block. Our model applies attention networks to learn interaction features between dataset aspects and query words, i.e., cross-attention. Specifically, we use the scaled dot-product attention for this network block [26].

Transformer Block. Similar to previous studies that employ BERT for the table retrieval [7] as well as for Dataset Search in NTCIR-15 [17, 18, 24], we apply it to create a contextual vector representation of the inputs. For that, we assume the dataset aspects as a single-field (text) document and use the token $[SEP]$ to separate dataset segments from query words. Lastly, we adopt the final hidden state h of the first token $[CLS]$ as the whole $\langle$*dataset*, *query*$\rangle$ representation. Regarding the implementations, we adopt TFBertModel from Hugging Face.[4]

MLP. In the top of the network, the attention vectors produced by the attention block are concatenated with the BERT vector and fed into a Multi-Layer Perceptron Architecture (MLP) to learn matching features and predict a similarity score. The

[2]https://www.elastic.co/elasticsearch/

[3]https://fasttext.cc/

[4]https://huggingface.co/transformers/model_doc/bert.html

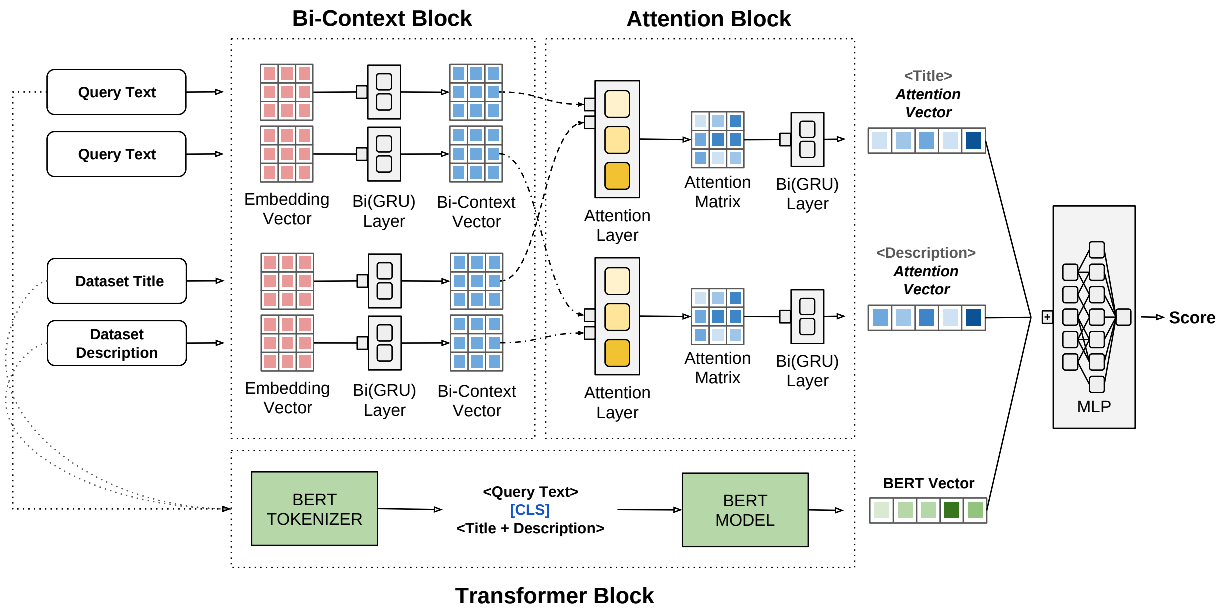


Figure 1: An overview of the matching model. Our model learns a joint-representation for a ⟨*query, dataset*⟩ tuple by applying three network blocks: Bi-Context, Attention and Transformer. Moreover, An MLP architecture captures relevant matches and produces the similarity score on its top layer.

score is generated by a sigmoid function over the last MLP neuron. We use this score to re-rank the datasets retrieved by the top-k retrieval algorithm.

Train Methodology. Since we do not have enough supervised data into the IR sub-task (there are only 141 query-dataset pairs labeled as highly relevant — i.e., have an L2 rating — in the training corpus), we train the proposed model on a task similar to table retrieval. Specifically, we consider the task of matching news articles and web tables [16] and use a corpus of 84k news-table matching pairs created by distant supervision. In short, the task goal is to retrieve web tables that are relevant to augment news stories, i.e., we also aim to match structured data (from the web table side) and unstructured information (from the news side), similar to the dataset search task.

When training the model, we change the model's input from dataset aspects to web tables aspects (i.e., the surrounding text of the web table). Moreover, instead of feeding the query text as input, we use the news aspects (news title and short description). The key idea behind our training methodology is the hypothesis that we can transfer the knowledge obtained in the source task (news-table matching) to the target problem (dataset search): since the core problem in both tasks is to compute the relevance degree between text and structured data, we might be able to reuse the model trained in source task to make predictions in the target task.

2.4 Experimental Setup

In this section, we cover the setup of our dataset ranking study. In what follows, we present the baselines, the experimental dataset, and the evaluation methodology. The source code of our implementation is publicly available on GitHub.[5]

Baselines. We evaluate the effectiveness of both traditional document retrieval methods and novel document/sentence encoders at the ranking phase. For Cos(TF-IDF) and BM25, we consider a multi-field document ranking, while for the other techniques we assume a single-field document approach [3]. We provide a short description of each baseline we used below:

- **Cos(TF-IDF)** [20] is the classic IR method that represents query/documents based on term-frequency and inverse document frequency. It ranks the datasets based on cosine similarity over the TF-IDF vector. We use the implementation of Elasticsearch for this approach.
- **BM25** [19] is the classic IR algorithm based on the probabilistic relevance framework that uses term-frequency weighting and document length for ranking. We also employ Elasticsearch for this ranking function.
- **Doc2Vec** [15] is an unsupervised approach that encodes sentences, paragraphs, or documents using neural networks, similar to Word2Vec. We consider a Gensim pre-trained model for this method,[6] and the cosine similarity is used to score and rank the datasets over the query.
- **Universal Sentence Encoder (USE)** [5] is a transformer-based network that learns sentence embeddings by using attention. The model was pre-trained on similarity-related tasks such as textual entailment and question

[5]https://github.com/levysouza/NTCIR16-Competition
[6]https://radimrehurek.com/gensim

#	Method	Dataset Aspect	NDCG@5	NDCG@10	NDCG@15	NDCG@20
1	Cos(TF-IDF)	Title	0.1077	0.1304	0.1472	0.1599
2	BM25	Title	0.1195	0.1417	0.1610	0.1747
3	DOC2VEC	Title	0.1982	0.2105	0.2237	0.2329
4	USE	Title	0.2041	0.1994	0.2091	0.2239
5	Cos(TF-IDF)	Description	0.1249	0.1511	0.1749	0.1908
6	BM25	Description	0.2017	0.2432	0.2741	0.3027
7	DOC2VEC	Description	0.1267	0.1329	0.1459	0.1565
8	USE	Description	0.1619	0.1680	0.1781	0.1980
9	Cos(TF-IDF)	Title, Description	0.1268	0.1505	0.1740	0.1883
10	BM25	Title, Description	0.1653	0.2166	0.2432	0.2629
11	DOC2VEC	Title, Description	0.1404	0.1482	0.1581	0.1679
12	USE	Title, Description	0.1901	0.1964	0.2059	0.2213
14	**Our Method**	**Title, Description**	**0.2801**	**0.2737**	**0.2834**	**0.2834**

Table 2: Ranking results by considering different datasets aspects and algorithms. We evaluate the methods by measuring *NDCG* at cutoffs $k = \{5, 10, 15, 20\}$. Bold values show the best algorithm.

answering. We import USE from TensorFlow Hub and consider the fourth version of the model.[7] Also, we use the cosine similarity to calculate the dataset score.

Experimental Dataset. We evaluate all methods using the NTCIR-16 Data Search 2 Dataset,[8] which contains a set of $46k$ public datasets and 192 queries. More details about this data collection can be found in [13].

Methodology. Following previous work of table retrieval [23, 25], we assume there is a pool of the top-100 candidate datasets for re-ranking. Based on that, we evaluate Doc2Vec, USE, and the proposed model over the ranking step. Moreover, we consider Cos(TF-IDF) and BM25 over the whole corpus as baselines for comparison. Lastly, we employ $NDCG$ at cutoffs $k = \{5, 10, 15, 20\}$ to measure the quality of the ranking for each method.

2.5 Results and Discussions

We now discuss the results for the ranking step. Table 2 shows the *NDCG* scores at cutoffs $k = \{5, 10, 15, 20\}$ for each methodology (note we group the results by dataset aspects). In general, our model outperforms all baselines for several *NDCG* metrics. For example, our approach is at least 37% most effective for ranking datasets in terms of *NDCG@5* (0.2801) compared to the best baselines, i.e., USE over the title aspect and BM25 over description field, in which the *NDCG@5* scores are 0.2041 and 0.2017 respectively. Also, by comparing our ranking method against classical document retrieval algorithms as Cos(TF-IDF) (over dataset title and description), our approach is more than twice effective for *NDCG@5*. Last, by examining *NDCG@10*, our method is 12% better effective (0.2737) than the BM25 algorithm by applying the dataset description (0.2432). Based on that, we argue that our approach ranks the best matching datasets at the first ranking positions in comparison to the other dataset search approaches.

By analyzing the other dataset search techniques (when we apply both dataset title and description over the search), the BM25 algorithm and USE are the best baselines for most *NDCG* scores, and the method Cos(TF-IDF) gets the worst results in the ranking. Furthermore, by comparing the document/sentence encoders (DOC2VEC and USE), Universal Sentence Encoder surpasses DOC2VEC for all *NDCG* metrics. Last, the BM25 algorithm also outperforms both DOC2VEC and USE as well as the Cos(TF-IDF) method for NDCG@k = {10, 15, 20} (only for *NDCG@5* USE is better than BM25).

Finally, the official evaluation [14] shows that our approach attained the highest score among all submitted runs across all evaluation metrics. In particular, for the nDCG@5 measure, our score of 0.246 represents a 30% improvement compared to the second-best result in NTCIR-16 Data Search 2 Task [14].

3 UI SUBTASK

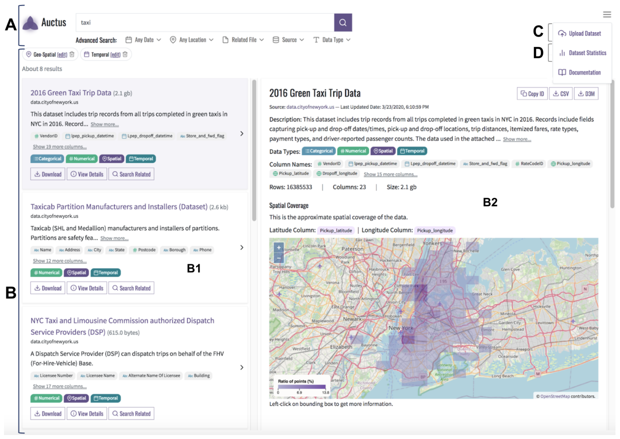
In this section, we describe the work that we performed for the UI Subtask. Our goal for this subtask was to evaluate the design and usability of user interfaces for dataset search tasks. More specifically, we conducted two user studies to evaluate the search engine result page (SERP) of Auctus [4], a dataset search engine designed to support data discovery.[9]

Auctus' goal is to provide a user-friendly interface that allows users to pose not only simple textual queries that match metadata but also a wide range of structured data queries [10, 21, 22, 28] that match data in the tables (e.g., finding datasets that join with a given input query dataset). For this study, however, we restrict the study scope and focus on evaluating the effectiveness of the visual components from Auctus' SERP.

In the remainder of this section, we describe Auctus focusing only on data discovery queries and results presentation (Section 3.1). In addition, we present two user studies that

[7]https://tfhub.dev/google/universal-sentence-encoder/4
[8]https://ntcir.datasearch.jp/subtasks/ir/

[9]Auctus is available at https://auctus.vida-nyu.org/.



About 8 results

2016 Green Taxi Trip Data (2.1 gb)
data.cityofnewyork.us
This dataset includes trip records from all trips completed in green taxis in NYC in 2016. Record... Show more...
VendorID | lpep_pickup_datetime | Lpep_dropoff_datetime | Store_and_fwd_flag
Show 19 more columns...

Taxicab Partition Manufacturers and Installers (Dataset) (2.6 kb)
data.cityofnewyork.us
Taxicab (SHL and Medallion) manufacturers and installers of partitions. Partitions are safety fea... Show more...
Name | Address | City | State | Postcode | Borough | Phone
Show 12 more columns...

NYC Taxi and Limousine Commission authorized Dispatch Service Providers (DSP) (615.0 bytes)
data.cityofnewyork.us
A Dispatch Service Provider (DSP) can dispatch trips on behalf of the FHV (For-Hire-Vehicle) Base.
Licensee Number | Licensee Name | Alternate Name Of Licensee | Building
Show 17 more columns...

2016 Green Taxi Trip Data
Source: data.cityofnewyork.us — Last Updated Date: 3/23/2020, 6:10:59 PM
Description: This dataset includes trip records from all trips completed in green taxis in NYC in 2016. Records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data used in the attached ... Show more...
Data Types: Categorical | Numerical | Spatial | Temporal
Column Names: VendorID | lpep_pickup_datetime | Lpep_dropoff_datetime | Store_and_fwd_flag | RateCodeID | Pickup_longitude | Pickup_latitude | Dropoff_longitude | Show 15 more columns...
Rows: 16385533 | Columns: 23 | Size: 2.1 gb
Spatial Coverage
This is the approximate spatial coverage of the data.
Latitude Column: Pickup_latitude | Longitude Column: Pickup_longitude
Left-click on bounding box to get more information.

Figure 2: Components of Auctus' user interface: keyword and filter-based search box (A); search results (B); dataset-snippets (B1); dataset summary (B2); dataset upload (C); dataset collection statistics (D).

we conducted to evaluate Auctus' UI (Section 3.2). First, we provide a detailed analysis of our first round of think-aloud interviews with seven people from which we collected user feedback and insights (Section 3.2.1); and finally, we discuss the second round of interviews conducted with ten people to assess Auctus' usability and design improvements made based on feedback from the first round of interviews (Section 3.2.2).

3.1 The Auctus System

Auctus is an open-source dataset search engine that was designed to support data discovery and augmentation [4]. It supports a rich set of discovery queries: in addition to keyword-based search, users can specify spatial and temporal queries, data integration queries, and they can also pose complex queries that combine multiple constraints. These queries are enabled in part by a data profiler [9] that we developed to automatically extract metadata from the actual datasets. This information is then used to construct indices that support efficient query evaluation. Users can search large dataset collections through an easy-to-use interface that guides them in the process of specifying complex queries. To help users identify relevant datasets, Auctus displays snippets that summarize the contents of datasets.

3.1.1 Auctus' User Interface. Auctus provides an easy-to-use interface that allows users to query for datasets, explore search results, explore ingested datasets, and upload new datasets. Below, we briefly describe the visual components included in the SERP that allows users can issue data discovery queries and explore the search results. For a more complete description of the system we refer the reader to [4].

Data Discovery Queries. Users can query indices by specifying keywords and constraints using various filters (see Figure 2(A)).

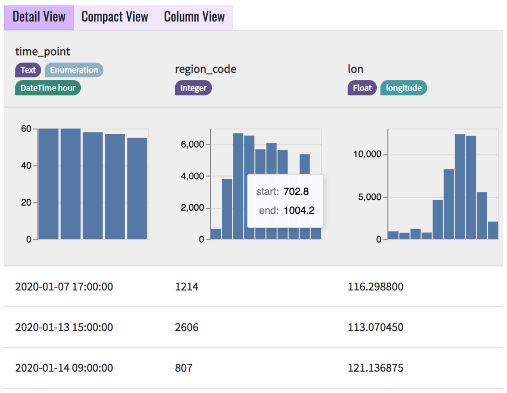
Temporal and Spatial Search. Users search for datasets by specifying a date range – datasets containing a temporal column that overlaps with that range will be retrieved. They can further refine the search by specifying a desired temporal resolution (e.g., year). To perform a spatial search, users can either draw a bounding box around a geographical area on the map, or specify an administrative area, which Auctus translates into a polygon. Datasets containing spatial attributes that overlap with the search polygon are returned.
Source Filter. The source filter allows users to restrict the data sources of interest, and only results from the selected sources are retrieved.
Data Type Filter. The data type filter allows users to search for datasets based on the types of their attributes, e.g., categorical, numerical, spatial, and temporal, which were inferred by the profiler.
Data Integration Queries. The related file filter allows users to find datasets that can augment a given input dataset. The user can upload the input dataset or select a dataset from a set of search results.

Result Presentation and Exploration. Unlike Web documents which can be summarized using short text snippets, datasets have many different facets that the user must consider to determine their relevance. Thus, an important challenge for Auctus is how to present search results. Figure 2 shows the results for the query "taxi". On the left, there is a list of search results displayed as snippets (see Figure 2(B1)). Users can select a dataset to inspect its details (see Figure 2(B2)), which include description, source, attribute names and types, as well as a summary of the dataset's contents. For spatial datasets, a visualization showing the geographical extent covered by the dataset is displayed, as shown in Figure 2(B2). The summary also includes a sample of the dataset records and statistics about its columns (Figure 3). The tabs above the dataset sample allow for the visualization of these statistics in different levels of detail. For example, *Detail View* shows plots of the columns' value distributions, *Compact View* shows minimal information next to the column names and above a sample of the data, and *Column View* shows detailed statistical information about each column.

3.2 User Study

We conducted two-round interviews for this UI Subtask. The first round of interviews were conducted to validate the Auctus design decisions, and the second round was run to evaluate Auctus usability and design improvements.

3.2.1 Expert Interviews. To validate our design decisions, we conducted a first round of interviews with seven people (U1-U7). Each interview took 40 minutes and proceeded as follows. We first conducted a tutorial session to present Auctus to the participant and clarified any questions they had (12 min). This tutorial consists of three parts: (1) a video demonstrating our system, (2) a presentation emphasizing Auctus' functionalities that will be mostly used during the user study, and (3) an



time_point	region_code	lon
2020-01-07 17:00:00	1214	116.298800
2020-01-13 15:00:00	2606	113.070450
2020-01-14 09:00:00	807	121.136875

Figure 3: Data summary views.

interactive exploration with an example query to familiarize the participant with Auctus. Then, we let them answer several dataset search questions using Auctus (23 min). Finally, we asked the participant to fill out a system usability questionnaire and left any comments on the system (5 min). The participants were free to explore any features in Auctus that may help them answer these dataset search questions. They were asked to think out loud while using Auctus. When the participants performed the task, we took notes related to the actions they performed.

Dataset Search Questions. To assess and validate our design decisions, we evaluate the effectiveness of the different features of Auctus using a variety of dataset search questions:

- *Effectiveness of Dataset Summary.* Useful information is highlighted in the Auctus' interface to summarize the contents of each dataset: column names and data types displayed in the snippets; additional information like number of rows and dataset size in the dataset summary, which is located on the right side of the interface. We formulate two questions to evaluate the effectiveness of these features: The first asked users to identify datasets that contains certain data types, and the second asked users to ascertain the dataset relevancy in terms of size (e.g., according to their number of features).
- *Effectiveness of Spatial Coverage.* One prominent feature of Auctus is the spatial coverage information, which can be shown through a *Spatial* data type tag or a *Spatial Coverage* map. This can help users quickly find datasets that contain geographical information. We designed a question that asks the participant to explore the search results and check if any of the datasets contains spatial information.
- *Informative Column Statistics.* The data summary views (Figure 3) include statistics about each column and visualizations in different levels of detail. To examine this informative feature, users were asked to answer a statistical question about a specific column in a dataset.
- *Effectiveness of the Entire System.* Finally, we designed a question that is not focused on a specific feature of Auctus. Instead, the participant can use any features of Auctus to answer the question and we can understand which are the *most-frequently used* features. These kinds of questions are formulated as follows: *Which datasets are more relevant to study topic T?*

User Insights and User Feedback. We received positive feedback from the participants. They expressed interests in using Auctus and suggested new features to improve the system. Followings are some of their feedback:

- U1 mentioned that it is useful, for comparison purposes, to have the dataset snippets (left side of the interface) and the dataset summary (right side of the interface) side by side, and also to have repeated information on both sides (e.g. *data types*). He says that he can have the dataset of interest in the right side and then scroll down the left side to check the snippet information of other datasets and make comparisons.
- Most of the participants (U1-U7) appreciate the *Dataset Summary* component (see Figure 2(B2)). They used this side of the interface to actually evaluate whether the datasets was useful for their task or not. However, some of them (U1, U3) suggested to better organize the information presented in the top section of this component: grouping related information may help users to identify relevant data more quickly.
- U1-U5 expressed interests in the *Data Summary Views* (Figure 3), which summarize data samples in three different ways. Most of them found the *Detail View* the most useful tab because it shows summaries and sample data in the same view. Based on this feedback, we rearranged the tabs order to make the *Detail View* the default tab.
- U2 expressed interests in using Auctus for her study on evaluating different machine learning algorithms (e.g. find datasets of different size). For suggestions, she mentioned that, besides number of rows, Auctus should also display the number of columns. This could help her to see how much data contains a dataset in terms of columns. For the histogram plot in *Detail View*, it would be great to sort the bars in a consistent way (e.g. either by frequency or by alphabetical order).
- U5 suggested that, similar to the spatial coverage map, it would be great if Auctus also had a temporal coverage plot to help users better identify the datasets that are in the temporal range they desire.

3.2.2 Usability Evaluation. After the first round of interviews, we implemented improvements in Auctus based on users' insights and feedback. To evaluate Auctus usability and design improvements, we conducted a second round of interviews with ten people. Two types of usability evaluation techniques were employed: The performance-based evaluation and the questionnaire-based evaluation.

The goals for this second user study are:

(1) Explore and analyze the datasets made available for the NTCIR Dataset Search 2 competition.
(2) Assess Auctus' ability to help users to complete search tasks. To carry out this performance-based usability assessment, we measured and analyzed observed question success rates, question completion times, features usage ratings, and users' feedback.
(3) Evaluate the perceived usability by performing a standardized questionnaire-based evaluation (SUS). This evaluation was performed in both rounds of interviews, and therefore, it allows us to compare Auctus UI's improvements implemented for this second round.
(4) Identify system weaknesses by according to the results obtained in the performance-based evaluation. This helps us to recognize which features are hard to understand or not intuitive.

Each interview took 50 minutes and proceeded as follows. We first conducted a tutorial session, as we did for the first round (12 min). Then, we let participants perform the actual user study using `Auctus` (30 min). The study consists of 3 search queries (each of them containing 5 questions). The queries cover three topics: 1) crime rate, 2) hospital, and 3) causes of death. Each query comes with a background description and a direct link to the searched results in `Auctus`. Note that we recorded response time for each question separately for later analysis. Finally, we asked the participant to fill out a system usability questionnaire (SUS) and left any comments on the system (5 min). The participants were free to explore any features in `Auctus` that may help them answer these dataset search questions.

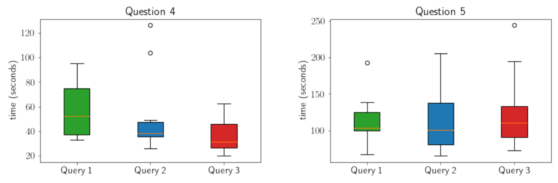
Performance-based Evaluation. Performance evaluation was done based on certain metrics which are measured while performing the questions. These metrics are described below:

- *Question Completion Time.* We designed five questions for each query. Each question asks same type of question across different queries. For example, question 1 in all 3 queries ask about data types of relevant datasets. As a result, for the same type of question, we expect that users will spend less time to find the answer as they become more familiar with both question type and the Auctus UI. For example, less time should be spent on question 1 in query 3 than that in query 1. We record the user response time of each question.
- *Question success rates.* We analyze the accuracy of each question to understand if Auctus can help users find the correct answer to dataset query questions. Accuracy of questions to all three queries are shown in Table 3.
- *Feature Usage.* For each question, we also conduct a feature analysis question to help us better understand which features are most helpful to answer each question.

Based on the analysis of *question completion times, question success rates, post-question Auctus' features usage ratings for each question*, we obtained the following interesting observations:

	Query 1	Query 2	Query 3
Question 1	1.0	1.0	1.0
Question 2	1.0	1.0	1.0
Question 3	1.0	0.9	0.8
Question 4	1.0	1.0	1.0
Question 5	0.8	0.8	0.6

Table 3: Question success rates (accuracy).



(a) Question 4 (b) Question 5

Figure 4: Time spent by users to complete Question 4 and Question 5 for different queries.

- *Completion Time.* For straightforward questions that are easy to solve, a decreasing trend in time is observed Figure 4(a). For questions that are difficult to solve or required a deeper analysis, no obvious decreasing trend in time is observed Figure 4(b).
- *Success Rates and Feature Usage.* Table 3 demonstrates that Auctus can help users easily find correct answers to most questions for all queries. However, some questions were found to be consistently hard across all queries. For example, question 5 has accuracy of 0.8, 0.8 and 0.6 on three different queries. Even tough users spent significant amount time on this question (see Figure 4(b)), some users were not able to find the correct answer. Moreover, question 5 in query 3 has the worst accuracy because more efforts are needed to carefully investigate dataset views (i.e. dataset samples and statistic summaries) compared to that in query 1 and query 2, as shown in Figure 5.
- *Feature Usage.* For questions that require to compare multiple datasets, users prefer to use Dataset Snippets. Users mentioned that having summarized information about the dataset in the snippets let them quickly identify datasets with certain characteristics, like data types, as shown in Figure 6. In cases where the users need to inspect column-specific information, they seem to prefer Auctus features that belong to the *Dataset Details* view. For questions that involve analysis of geographical information, most of the users (96.6%) make use of the spatial coverage map.

Questionnaire-based Evaluation. We also evaluated the usability of `Auctus` using the well-known System Usability Score (SUS) [2]. We conducted a survey at the end of each interview for both user studies: we asked participants to fill

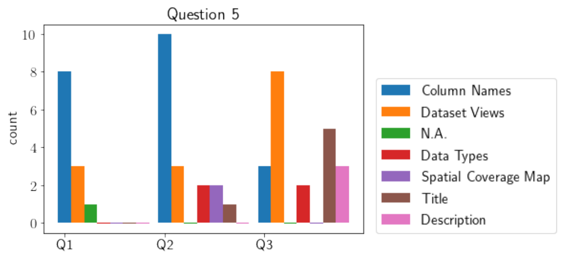


Figure 5: Distribution of different features used on Question 5.

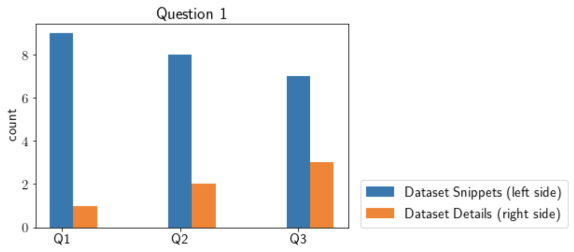


Figure 6: Distribution of different features used on Question 1.

out the standard SUS survey, grading each of the 10 statements on a scale from 1 (strongly disagree) to 5 (strongly agree). The SUS grades systems on a scale between 1 and 100 to assess the quality of system interfaces [1]. In the first round of interviews, `Auctus` obtained an average score of 88.33±15.51. According to Bangor et al. [1], a mean score above 80 is in the fourth quartile and is acceptable. The average score achieved in the second round of interviews (after improvements) was 88.80 ± 15.45. Even though this is not a significant increase, the score from the second user study suggests that the changes implemented based on user feedback improved the system usability.

4 CONCLUSIONS

In this paper, we describe the approach and results of Team NYUCIN for Data Search Task in NTCIR-16. To overcome the lack of training data, we employed a transfer-learning approach that leverages a model trained for a different but closely-related task. Our experimental results suggest that using transfer learning and distant supervision approaches might lead to significant improvements in the ranking quality of dataset search engines. Finally, we also presented Auctus, our effort in developing an effective dataset search engine, and two user studies to evaluate the effectiveness of its user interface.

REFERENCES

[1] Aaron Bangor, Philip T Kortum, and James T Miller. 2008. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction* 24, 6 (2008), 574–594.

[2] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[3] Michael J. Cafarella, Alon Y. Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. 2018. Ten Years of WebTables. *VLDB* 11, 12 (2018), 2140–2149.

[4] Sonia Castelo, Rémi Rampin, Aécio Santos, Aline Bessa, Fernando Chirigati, and Juliana Freire. 2021. Auctus: A Dataset Search Engine for Data Discovery and Augmentation. *Proc. VLDB Endow.* 14, 12 (jul 2021), 2791–2794.

[5] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *CoRR* abs/1803.11175 (2018).

[6] Zhiyu Chen, Haiyan Jia, Jeff Heflin, and Brian D. Davison. 2020. Leveraging Schema Labels to Enhance Dataset Search. In *ECIR (Lecture Notes in Computer Science, Vol. 12035)*. Springer, Lisbon, Portugal, 267–280.

[7] Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Yinan Xu, and Brian D. Davison. 2020. Table Search Using a Deep Contextualized Language Model. In *SIGIR*. ACM, Virtual Event, China, 589–598.

[8] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014).

[9] Datamart Profiler Library. 2021. https://pypi.org/project/datamart-profiler/.

[10] Raul Castro Fernandez, Jisoo Min, Demitri Nava, and Samuel Madden. 2019. Lazo: A cardinality-based method for coupled estimation of jaccard similarity and containment. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1190–1201.

[11] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS*. Montreal, Quebec, Canada, 2042–2050.

[12] Emilia Kacprzak, Laura M. Koesten, Luis-Daniel Ibáñez, Elena Simperl, and Jeni Tennison. 2017. A Query Log Analysis of Dataset Search. In *ICWE (Lecture Notes in Computer Science, Vol. 10360)*. Springer, Rome, Italy, 429–436.

[13] Makoto P. Kato, Hiroaki Ohshima, Ying-Hsang Liu, and Hsin-Liang Chen. 2021. A Test Collection for Ad-hoc Dataset Retrieval. In *SIGIR*. ACM, Virtual Event, Canada, 2450–2456.

[14] Makoto P. Kato, Hiroaki Ohshima, Ying-Hsang Liu, and Hsin-Liang Chen. 2022. Overview of the NTCIR-16 Data Search 2 Task. In *Proceedings of the 16th NTCIR Conference on Evaluation of Information Access Technologies,*.

[15] Quoc V. Le and Tomás Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*, Vol. 32. Beijing, China, 1188–1196.

[16] Alyssa Lees, Luciano Barbosa, Flip Korn, Levy de Souza Silva, You Wu, and Cong Yu. 2021. Collocating News Articles with Structured Web Tables. In *WWW*. ACM / IW3C2, Virtual Event, Ljubljana, Slovenia,, 393–401.

[17] Phuc Nguyen, Kazutoshi Shinoda, Taku Sakamoto, Diana Andreea Petrescu, Hung Nghiep Tran, Atsuhiro Takasu, Akiko Aizawa, and Hideaki Takeda. 2020. NII Table Linker at the NTCIR-15 Data Search Task: Re-ranking with Pre-trained Contextualized Embeddings, Data Content, Entity-centric, and Cluster-based Approaches. In *NTCIR-15*. Tokyo, Japan, 286–292.

[18] Taku Okamoto and Hisashi Miyamori. 2020. KSU Systems at the NTCIR-15 Data Search Task. In *NTCIR-15*. Tokyo, Japan, 274–281.

[19] Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[20] Gerard Salton and Chung-Shu Yang. 1973. On the specification of term values in automatic indexing. *Journal of Documentation* (1973).

[21] Aécio Santos, Aline Bessa, Fernando Chirigati, Christopher Musco, and Juliana Freire. 2021. Correlation sketches for approximate join-correlation queries. In *Proceedings of the 2021 International Conference on Management of Data*. 1531–1544.

[22] Aécio Santos, Aline Bessa, Christopher Musco, and Juliana Freire. 2022. A Sketch-based Index for Correlated Dataset Search. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*.

[23] Roee Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Canim. 2020. Web Table Retrieval using Multimodal Deep Learning. In *SIGIR*. ACM, Virtual Event, China, 1399–1408.

[24] Lya Hulliyyatus Suadaa, Lutfi Rahmatuti Maghfiroh, Isfan Nur Fauzi, and Siti Mariyah. 2020. STIS at the NTCIR-15 Data Search Task: Document Retrieval Re-ranking. In *NTCIR-15*. Tokyo, Japan, 282–285.

[25] Yibo Sun, Zhao Yan, Duyu Tang, Nan Duan, and Bing Qin. 2019. Content-based table retrieval for web queries. *Neurocomputing* 349 (2019), 183–189.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. Long Beach, USA, 5998–6008.

[27] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations. In *AAAI*. Phoenix, USA, 2835–2841.

[28] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J Miller. 2019. Josie: Overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the 2019 International Conference on Management of Data*. 847–864.