Automatic Query Response System for RIT Housing

Anonymous

ABSTRACT

Individuals looking to live on-campus at any university have many housing options to choose from. Students wish to explore all the available options and select the most suitable apartment. In this process, students will have numerous queries regarding different aspects of the housing such as house rent, co-occupant selection, contract details, and others. A few students who are already staying on campus might also have a few doubts and confusions. Universities usually have details of housing operations available online for the benefit of students. However, not all the queries of the students are addressed by this, since many students do not go through all the details of housing contracts or frequently asked questions (FAQs). The objective of this project is to help the Housing operations as it would reduce the number of people who have to respond to student emails. It would also be easier for the students and other users to get their queries clarified by an automated answer system as the response time is less.

1. INTRODUCTION

One of the basic needs of an individual is housing. This is particularly more important for students studying away from home. This project is concerned with Housing Operations at RIT. There is a wide range of housing options to choose from. With each additional option, there are more questions and queries. Information about the housing options at RIT is available online and is publicly accessible [8], but it contains a lot of information that needs the reader to visit multiple pages and put things together by themselves. It would always be better if there is a way to find the right answer to the question we have in mind, instead of having to read over to multiple pages to know the answer. Live chat is an option where a student can get all their doubts clarified talking to a human representative, but live chats require many human employees as the number of querying students are very high. Also, humans work for a limited number hours in a day.

To help improve this we have developed a query response system using IBM Watson on Bluemix. We have used Retrieve and Rank feature of Watson for the task. Through this, we have trained the machine with most of the questions that a student might ask - which ranges from a simple question like '*what is the rent?*' to '*where do I leave my keys at the end of the year when I move out?*.'

1.1 Data Collection

To train Watson, we have acquired text documents from RIT Housing Operations that contain information about the different RIT Housing options that are available. We got four sets of documents, namely, Housing Selection, Semester Break, Arrival Guides and End of Year. Each set had about four to five individual documents [8, 9, 11, 10]. These documents contain housing information and even the terms and conditions. If a student were to proceed with the rental process or would like to just look at them, these are the documents he/she would look out for. RIT housing operations provided us with Frequently Asked Questions along with the answers. These were the questions that students asked in the form of emails.

2. DATA CLEANING AND PREPARATION

The source data were provided to us in PDF files as text. We converted these files to DOC files. We also had to change the format from multiple columns text to single column text because when we tried converting these document with Watson's Document Conversion, we kept getting texts that did not have a proper structure since it was trying to convert the documents line by line. We used different types of font for the headers and the description. This was done to get Watson's Document Conversion service to differentiate between the title of the topic and the content. This was done so that it would be easier for the Document Conversion service of Watson to generate the textual information in JSON format for processing. Also, some of the headers were very long, which we had to trim. Additionally, there were some topics that were present in multiple documents and also repeated within the same document. We studied all the files and removed this redundancy making the data more distinct. The Frequently Asked Questions provided by the team at RIT Housing Operations were lengthy and had to be split into multiple parts. This would help in the better training of the system.

3. IMPLEMENTATION

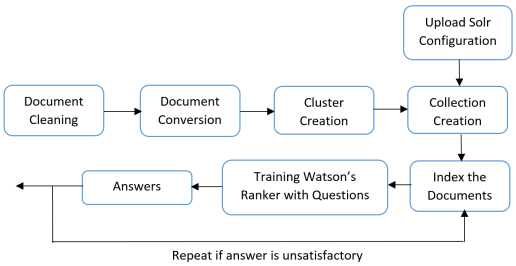

Figure 1: Project Architecture

3.1 Document conversion

After preparing the document in the header and the normal text form, where the document header contains topic related to the text in the H2 (header) format and the normal text is in ordinary text format, it is fed to the document conversion service of IBM Watson [2]. The Document Conversion service's configuration is customized to identify the text in the H2 header format as topics and the normal text following the H2 header as the answer related to that header [2]. The output of Document Conversion is a JSON text that assigns unique document IDs (*doc_id*) to each document, which in our case is the combination of header and the corresponding answer. These *doc_ids* will later be useful in training the ranker module of Watson [5].

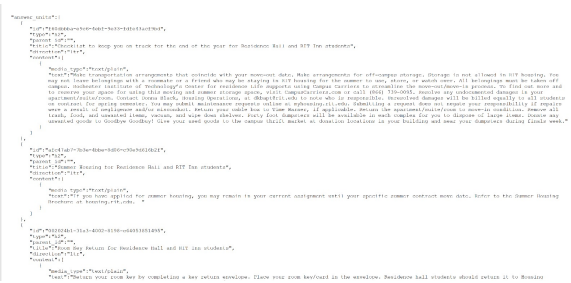

Figure 2: JSON file

3.2 Cluster and Collection Creation

The cluster is the most important component for Watson's Retrieve and Rank service [7]. It serves as the canvas on which all other components reside and communicate, with Apache Solr as the communication medium [1]. The cluster also manages all the collections which will be unique for a project. A collection maintains all the documents that were added to the retrieve and rank service [6].

3.3 Index/Searching

After creating a cluster, it takes quite some time to get ready for taking requests. Once it is ready, we upload solr configuration to configure the cluster. The *schema.xml* and *solrconfig.xml* files define how the fields in the source documents are defined and how these fields need to be indexed for easier searching.

3.4 Training

We used question-answer pairs to train the model. This was done using the doc_ids. Each question could have multiple answers as well. We had a relevance factor for each answer to a question, irrespective of whether there was only one answer or multiple answers. The relevance factors ranged from zero to five. Zero being the least relevance answer and five being the most relevance answer for that question. We used about 500 questions for the training [3, 4]. We started to train Watson's Retrieve and Rank service using training questions. Same questions were also asked in different ways to check if it is able to identify the answer to the questions. This was done multiple times to improve the results.

3.5 Results (Answers)

After querying Watson's trained question answering model, the service returns a list of ranked answers that match closely with the question, with the answer at the top of the list being the closest match to the question. If the user is not satisfied with the results, they can re-train the model by adding the question and the answer's *doc_id* to the ground truth file. This process is repeated over and over again for all kinds of questions; but, no matter how much we train, a system can never be perfect and if a system is indeed perfect, then the data should have overfitted the model.

The figure 3 shows a query "do i have to pay a termination fee if i drop out of college midway". The system correctly answers with the termination fee related information as to when a student is mandated to pay the termination fee and when he/she is not.

The figure 4 shows a query "is there housekeeping at rit inn". The system shows the correct result as to when the housekeeping staff would visit and what activities would done by them.

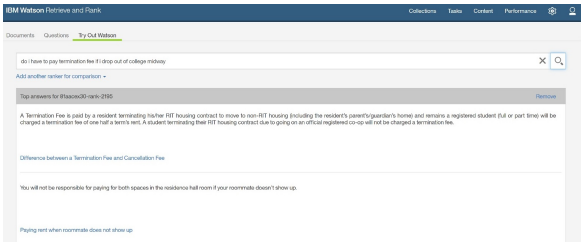

Figure 3: Result for query 1

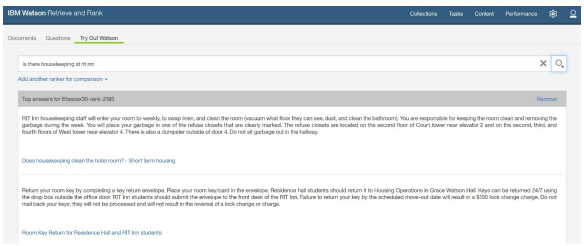

Figure 4: Result for query 2

4. ETHICAL ISSUES IN THE PROJECT

Any project, or even a task for that matter, related to data mining and analytics will have a lot of decisions made on ethical grounds like the accessibility of the data - publicly accessible or not, and other such issues. This section addresses the ethical issues we faced while developing the project. The frequently asked questions had the names of the students who had asked the questions, and thus, we had to anonymize the names and contact information so as to protect the privacy of the students in case this project were to go live in the future.

The second issue was about taking care not to provide incorrect information regarding housing contracts as this could in the future lead to legal issues. For this purpose, a question related to insurance and contract is answered in a paragraph and not a phrase as the wrong answer in phrases could take it totally out of context.

5. FINAL REMARKS

Natural language processing and cognitive computing has evolved in the past decade to become one of the most trending topic in computer science and many other fields. Enhancing traditional computing systems to have cognitive power is a complex process and involves a lot of dedicated resources. IBM Watson is one of the most powerful and used cognitive computing service platform, which offers various service for applications. Applications using these cognitive computing services has become very powerful and achieve tremendous success, which a normal application would never reach in most of the cases.

In our application, IBM Watson is the backbone which provides support for almost all the aspect of the services. Starting from document conversion to the formation of solar cluster, indexing, training the data model and other aspects of the application heavily relies on various IBM Watson services. Even with the monetary limitations, IBM Watson will be very helpful to many applications.

6. FUTURE WORK

Automatic query response system for RIT housing can be further enhanced to include a live chat box using the conversation service from the IBM Watson. Live chat would answer all the queries of incoming and existing students related to housing at RIT. Live chat box will be very helpful for the students to clear all the doubts related to RIT housing. Even RIT housing management will find this application helpful in utilizing the human resource.

A well-trained application will almost always be accurate, though there might be a few fail cases. More the training more accurate will be the results. So, in future IBM Watson model needs to be trained with a plethora of questions related to all the aspects of the RIT housing. Yet we need to be sure that we are not overfitting the data model.

Another feature we wish to add is a well-defined user interface that would let the users choose already existing questions or search for a question that is not available on the list.

7. ACKNOWLEDGEMENTS

We would like to thank the Executive Director of RIT Housing Operations and her team for providing the source documents and frequently asked questions. We would also like to thank the course instructor for her guidance.

8. REFERENCES

[1] Apache Solr Quickstart.
[2] Document conversion - api reference| IBM Watson Developer Cloud.
[3] How to use the "Retrieve and Rank" service in IBM Bluemix.
[4] How to use the "Retrieve and Rank" service in IBM Bluemix 2.
[5] IBM Watson Developer Cloud Github.
[6] Natural Language QA system with IBM Watson-Fartash.
[7] Retrieve rank - API Reference| IBM Watson Developer Cloud.
[8] RIT Housing. `https://www.rit.edu/fa/housing/`.
[9] RIT Housing - Arrival Guides. `https://www.rit.edu/fa/housing/content/fall-move-2016#arrival%20guides/`.
[10] RIT Housing - End of Year Closing. `https://www.rit.edu/fa/housing/content/end-year-closing`.
[11] RIT Housing Selection.

APPENDIX

A. USER INSTRUCTIONS

This appendix includes instructions to interact with the project.

1. Install the necessary packages needed using the commands
   npm install watson-developer-cloud –save
   npm install solr –save
   npm install jsonfile –save These commands will add the dependencies to the *package.json* file.
2. The first step is to create a cluster online on Bluemix and get the cluster ID. A cluster is a base for all operations concerned with Retrieve and Rank service of Watson. This is to be done by logging into the Bluemix console, launching the Retrieve and Rank tool and creating a cluster. It will take a few minutes before the cluster is created and is ready to use. The user then needs to note the cluster id.
3. Once the documents have been prepared in a way that can be understood by Watson's Document Conversion service, the user can execute *docconversion.js* file. To run *docconversion.js*, the user needs to set the input parameters *files_directory* to the path of the folder that contains all the source documents in *.doc* format and *json_directory* to the path of the folder where the user wishes to get the JSON files after conversion. For example,
   files_directory = './resources/'
   json_directory = './JSONFiles/'
   The command to run document conversion in command prompt is
   node docconversion
4. After invoking Document Conversion service and receiving the JSON files, the next step is to upload Solr

configuration. The Solr configuration is required to index the files and search answers to any query. The provided *solrconfig.zip* has *schema.xml* and *solrconfig.xml* that define how to index and search files, the fields in text, request handlers etc. Before running *uploadsolrconfiguration.js*, the user has to set the parameters in *inputconfigurations.js* as
cluster_id = 'scd2968656_0ff2_4fcd_948a_56c37e87ef1f'
config_name = 'HousingSolrConfig'
config_zip_path = 'solrconfig.zip'
where *config_name* is a user-defined name that is given to the Solr configuration that was uploaded, and *config_zip_path* is the path to the solr configuration zip file. After successful execution, the status will be 200.

5. The above step is followed by indexing the documents using Solr that will help with searching for answers. This requires setting the parameter *collection_name* to a user defined collection name. For example,
collection_name = 'HousingCollection'
A collection is where the JSON answer units from Document Conversion will be stored along with the training questions and answers. The file *indexdocuments.js* needs to be executed with the following command
node indexdocuments
It will take a few minutes to create a collection and when checked on the tool online, it might show a warning asking the user to update the schema. In that case, the user needs to click on the *'Update Schema'* button to update the configurations.

6. Now that all the components for using Retrieve and Rank are ready, the user needs to provide training questions for Watson, which is called the ground truth. The ground truth is a comma separated values file in the format *question, answer1, relevancefactor1, answer2, relevancefactor2....* The *question* covers all possible questions that a user could ask Watson, *answer1* refers to a matching answer in the document, and the *relevancefactor* denotes how relevant the answer is to the question - the higher the number, more relevant is the answer. Watson requires the user to write a minimum of 250 questions. This ground truth needs to be uploaded to Watson to train the ranker module. This is done using the *train.py* file provided by IBM. The command to train the ranker is
python train.py -u retrieve_and_rank_username:retrieve_and_rank_password -i ground_truth_file -c cluster_id -x collection_name -n "user_defined_ranker_name"

7. It takes a few minutes before the ranker is ready to accept questions. Once it is ready, the user has to note the *ranker_id*. This value is to be entered in the *ranker_id* of *inputconfigurations.js*. The user then has to enter their query that needs to be searched in the documents in the *question* parameter of *inputconfigurations.js* file. The user can then run *askquestions.js* file which will give the most relevant and ranked answers.

B. INDIVIDUAL CONTRIBUTIONS

Student Initials: KJ

- Document Cleaning involving structuring and formatting the text in Watson-readable format.
- Partial coding of Retrieve and Rank service.
- Wrote questions for ground truth file.
- Wrote Appendix section and README file.
- Contributed to writing report.

Student Initials: GN

- Document Cleaning involving structuring and formatting the text in Watson-readable format.
- Partial coding of Retrieve and Rank service.
- Wrote questions for ground truth file.
- Helped in creating the presentation.
- Contributed to writing report.

Student Initials: SCA

- Document Cleaning involving structuring and formatting the text in Watson-readable format.
- Coding of Document Conversion.
- Wrote questions for ground truth file.
- Helped in creating the presentation.
- Contributed to writing report.