"A cloud solution for analysing patterns in NGO projects"

Team Number: Team 36

Names: Rachel Mattoo, Mark Anson, Yansong Liu

Date: 27/03/2020

Overview of progress

BERT algorithm complete:

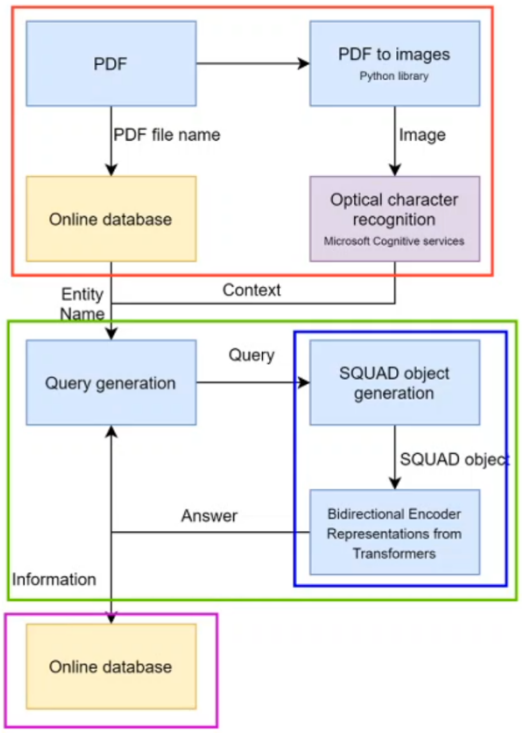
The remaining challenge was to implement the custom list of 30 questions I (Rachel) created with the ALBERT question and answering model. We were also required to run this algorithm across the whole pdf document and not just across one page, as we had previously programmed. We knew that this would reduce the accuracy of the ALBERT question and answering system, but it was the most time-effective solution.

Given more time, we would have fine-tuned the model on NGO specific data by creating a huge list of reading comprehension answers based the annual NGO reports and training the model on these particular questions. This process would have improved the accuracy slightly.

Here is how our algorithm works (also on our report website):

The functionality of BERT can be analogized by children doing reading comprehension task. Given a paragraph of reading material and several questions regarding that material, a child would be required to answer all the questions concisely and accurately. The normal steps for a human doing reading comprehension task is to read the material, read the questions and answer the question based on the material. However, because machines cannot really understand natural language, BERT uses numbers to represent words. Every single word can be represented by a particular number. BERT models then use all these numbers to create two high-dimension vectors. One number for the context and another for the question. The mathematical approach used is to measure the distance between the dimension of the context vector and the dimension of question vector. The shorter the distance between the two vectors the closer the meaning between them. Then BERT will return the vector with shortest dimension, as well as the text surrounding it. Then those numbers will be translated to the source language again as the final answer.

In this section we will briefly explain how all the features have been implemented. The code can be divided into 4 parts, as shown by the BERT algorithm diagram. Each part is implemented in a single script. The first part (the red box) imports all the pdfs which need to be processed from the input folder, and these pdfs are converted to in an image format using the pdf2image python library. pdf2image is a python wrapper for pdftoppm and pdftocairo to convert PDF into a PIL image object. This is because some pdfs in the input folder are already in an image format and so each pdf is converted into an image to ensure uniformity. The algorithm then uses Microsoft Azure's Cognitive Services Computer Vision OCR API to use optical character recognition to iterate through each page and convert these images into a JSON format. This JSON format is then converted into a string format.



For each pdf, the first thing to do is get the name of NGO associated with the annual report, from the online database (Database 2 – UN "knowledgebase") hosted on Azure. The names have already been uploaded to the database using the name extraction algorithm. The name extraction algorithm essentially allows the user to upload their pdfs to Database 2, under the pdfs table, and to choose whether to manually input the NGO name associated with the pdf or automate this process by using our name extraction algorithm. If the user chooses to use the name extraction algorithm, they must authenticate the output is correct before it can be uploaded to the database. The name extraction algorithm is a separate entity from the main BERT algorithm and has an accuracy of around 50%. We had to manually automate name extraction because BERT cannot recognise named entities such as NGO names because these words are unfamiliar to the corpus which BERT has been pre-trained on.

Now that we have the entity (NGO) name and the content of the report, this is passed on the Query generation part of the algorithm (green box). The Query generation part is entirely done in the Questions.py, where the answer for the previous question is used to generate the next question. For example, we want our algorithm to return a list of PROJECTS for each NGO. Then for each PROJECT, we want to ask the ALBERT* model further questions about this particular project such as project mission, start date, end date, location etc. So essentially the list of project names returned by ALBERT inform the questions asked about each project, using the project name. After the questions have been generated, they are sent to the ALBERT model (in docQuery_utils.py). The text and the questions are packaged into a single SQuAD object (blue box), and then passed to ALBERT. In order to simplify query generation with the ALBERT question and answering system, we used the Hugging Face Transformer python wrapper. The model generates a set of 20 predictions for each question, based on answers with the highest probability of being correct. We changed docQuery_utils.py so that the all the answers are returned as text, and not just the best answer with the highest probability,

which the model was previously configured to do. For example, for NGO address, we want the "best" answer (bestPredictionText) whereas for Project names we want all the possible names (nBestAnswerPredictionsText) so we choose the top 10 predictions returned by the ALBERT model.

The answers are returned to Questions.py which packages them and in turn returns the package to 'docQuery long context.py'. This data is then passed to submitDataToDB.py, which uploads the data to the corresponding table and field of the database, depending on the content of the answer. The reason we can easily determine table and field name is because Questions.py returns the answers to the questions as a list of nested dictionaries of table and field information for each table in the database. This allows us to easily write python insert SQL queries, so that this information can be directly uploaded to the database.

Deployment:

We have managed to successfully deploy the algorithm by linking the BERT algorithm with Database 2 using SQL database connections. There were a few minor SQL errors during integration testing, but we managed to deploy everything on time.

List of tasks completed and whether project is running on time

The project has run on time as we have successfully met our deadline (deadline extended due to coronavirus situation). Here is a list of tasks completed:

- Completed MoSCoW requirements
    - MUST and SHOULD completed
- Completed BERT algorithm
- Successfully deployed our project
    - Connected BERT algorithm with Database 2
        - Using SQL database connections

Plan for coming 2 weeks

There is no longer a plan because we have completed our project but here is an final evaluation of our project as we believe this is useful for future development. We want our project to have a good impact on society.

Final Evaluation

Functionality

Our final deliverable carries out its intended functionality, as per our user-requirements. We will address each of the MUST and SHOULD requirements, discussing the extent to which each requirement was successfully met and identifying areas for improvement.

1) **Database 1** – ANCSSC database for web app

This Database was created to host information for the master's team, for their front-end web app, and Team 37, to include geographical location details. Creating the database required multiple iterations as the master's team constantly updated their user requirements or improved on their current prototype designs. The backbone of the database was built on guidelines provided by Matt, a member of the ANCSSC and Team 37's client. After multiple meetings with both the Master's team and Matt, we managed to successfully create a database which satisfied both their requirements. We also provided the master's team a copy of the SQL script we created to build the database and some sample queries, so that they can make alterations in the future if need be.

2) **PDF Extraction tool**

- **Extract relevant information from pdf NGO reports**

The PDF extraction tools mostly manages to extract relevant information from the NGO reports. This includes information such as project information, staff details, NGO details. There are a few samples of garbage data scattered in the database, due to the BERT model misinterpreting the pdf text in the context of the question asked. This was something we expected when we chose our model, since it has not been finetuned in the context of NGO report data. The reason our model is not finetuned is because it is incredibly resource intensive and time consuming, as it involved creating a large custom list of questions in the context of our data, and powerful GPUs to train our model on those questions.

- **Using Azure Cognitive Services**

The PDF extraction tool successfully uses the Microsoft Azure Cognitive Services Computer Vision OCR API. This uses optical character recognition to convert pdf reports, in image format, into JSON text format. Cognitive Services is much more powerful compared to open source alternatives and the price justifies the improvement in performance. It is also within the ANCSSC monthly budget, as highlighted by the cost matrix.

- **ALBERT Question and Answering system to extract data from natural language**

The ALBERT model, an upgrade on BERT and developed by researchers at Google AI, was chosen because it has the highest accuracy according to the SQuAD rankings. ALBERT works well at extracting relevant information from natural language. However, even though it can work on any length of text, it is more optimized for a smaller length of text and for returning the best answer based on probabilities. This would not be the case if we had fine-tuned our model.

3) **Database 2** – UN" Knowledge-base"

This database contains fields to store information extracted from the annual NGO reports using PDF Extraction tool. The design for the ER diagram had undergone multiple revisions as involved analyzing multiple reports to determine how to structure the information. Through testing, each field except the primary key fields was changed to DEFAULT NULL and varchar data-type. This is because sometimes the model may not find an answer and the answer can only be returned as a string data type.

4) **Quality documentation** for PDF extraction tool

We have provided quality documentation for PDF extraction tool. We have written an extensive deployment manual and the report website, as well the bi-weekly reports, highlight the large amount of research involved to produce our final deliverable.

Project Management

We have worked well as team to successfully meet our requirements. The process has been challenging since we have had to communicate with multiple teams and liaise with multiple people, including Dean for technical support and our NLP expert, Dr. Pontus Stenetorp. Documenting these meetings and updating each other through platforms such as messenger has enabled us to meet our deadlines. Everybody in the team was very supportive of each other and we compensate for each other's weaknesses by allocating tasks based on our strengths.

Due to the coronavirus situation, we switched to Microsoft Teams, as messenger is not easily available in China. This allowed us to efficiently communicate and share resources, so that our project deliverables were of the highest quality possible.