Service quality ai-based system

Amr ayman   
 *cs   
Ain shams University*Cairo, Egypt  
[amr2018170267@cis.asu.edu.eg](mailto:amr2018170267@cis.asu.edu.eg)

Ahmed samir  
cs  
Ain shams University Cairo, Egypt  
ahmed2018170534@cis.asu .edu.eg

Youssef mohammed  
cs  
Ain shams UniversityCairo, Egypt  
youssef2018170483@cis.asu.edu.eg

Mostafa gamal  
cs  
Ain shams UniversityCairo, Egypt  
[mustafa2018170391@cis.asu.edu.eg](mailto:mustafa2018170391@cis.asu.edu.eg)

Ali fathy  
cs  
Ain shams UniversityCairo, Egypt  
[ali2018170241@cis.asu.edu.eg](mailto:ali2018170241@cis.asu.edu.eg)

Zyad el sayed  
cs  
Ain shams UniversityCairo, Egypt  
zyad2018170391@cis.asu.edu.eg

***Abstract*—** The concept of smart government has been gaining momentum in recent years, as governments around the world strive to improve service delivery and enhance citizen engagement with technology.

In this context, the Smart Government Service Quality Project represents an innovative and ambitious initiative aimed at improving the quality of government services and enhancing citizen satisfaction.

Our project has applied the target desired from smart government service quality idea as we have deployed many AI NLP-based techniques as topic classification, sentiment analysis, text generation and information retrieval, these techniques allowed us to analyze reviews, complains and feedbacks of citizens in order to get desired outputs automatically without need of human participating, these outputs includes: the ministry responsible for the review, the sentiment of the citizen posting the review, the specific governmental sector responsible for the problem in the review and the recommended solution for the problem in the review.

We have deployed a website to make it easy for every citizen to use it from his house, work, or any place with the internet.

We have applied fine tuning for transformer models like MARBERT, Arabert, Arat5, MT5 on our downstream tasks, we have reached **0.96** f1 score in sentiment analysis task, **0.957** f1 score in topic classification task and **7.8** BLEU score in text generation task.

Introduction— The Smart Government Service Quality Project is an innovative initiative aimed at improving the quality of government services using advanced technologies. The project leverages data analytics, artificial intelligence, and other smart technologies to optimize government service delivery processes, reduce bureaucracy, enhance transparency, and increase citizen satisfaction. The project is based on a customer-centric approach that puts citizen needs and preferences at the center of service delivery. Through this approach, the project seeks to transform the way governments deliver services, making them more efficient, effective, and responsive to the needs of citizens.

Motivation: The purpose of creating an Operating e-services application is to modify and enhance the connection between the government and the people. So, we are motivated to do such an application to help the people to communicate easily with the responsible men working in the government organizations and transform their opinions complains and even their gratefulness to avoid miss understanding and the feelings of being forgotten so we are enthusiastic about achieving this aims to help the government and the people to make a real progress in this field. Improving Mutual benefit between citizens and government organizations by the vastly improved flow of the information from citizens to governments, governments to citizens, and within government itself. Significantly improves communication between citizens and governments. Reduce Bureaucracy during the operations. Instead of the lots of paperwork operations the citizen or the government employee does it can be with a few clicks on the system to reduce a lot of wasted time.

Problem Definition: The Smart Government Service Quality Project aims to address several challenges that currently exist in government service delivery. These challenges include long wait times, inefficient service delivery processes, lack of transparency and accountability, and inadequate citizen feedback mechanisms. These issues can lead to low citizen satisfaction with government services and a lack of trust in government institutions. The project seeks to overcome these challenges by leveraging data analytics, artificial intelligence, and other smart technologies to optimize service delivery processes, enhance transparency and accountability, and improve citizen satisfaction.

Objective: Topics classification: Analyzing the review text and predicting its topic(ministry) to know which ministry to send the review. Sentiment analysis: Classifying the text into negative, positive, and neutral to make it easy to know the good and bad opinions. Generate Recommended Solutions: After the text is classified and we know its target sector, if the problem or opinion is like the known one before it will recommend actions to the problem. Classifying the Governmental Sector: Classify the review text to its specified governmental sector to make it easy to solve the problem in the review in its specified sector.

A screenshot of a computer screen

Description automatically generated with low confidenceRelated work— Arabert: The first meaningful representations for words started with the word2vec model developed by (Mikolov et al., 2013). 10 Since then, research started moving towards variations of word2vec like of GloVe (Pennington et al., 2014) and fast Text (Mikolov et al., 2017). While major advances were achieved with these early models, they still lacked contextualized information, which was tackled by ELMO (Peters et al., 2018). The performance over different tasks improved noticeably, leading to larger structures that had superior word and sentence representations. Ever since, more language understanding models have been developed such as ULM Fit (Howard and Ruder, 2018), BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019), ALBERT (Lan et al., 2019), and T5 (Raffel et al., 2019), which offered improved performance by exploring different pretraining methods, modified model architectures and larger training corpora.

Marbert: English and Multilingual LMs. Pre-trained LMs exploiting a self-supervised objective with masking such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b) have revolutionized NLP. Multilingual versions of these models such as mBERT and XLM-RoBERTa (Conneau et al., 2020) were also pre-trained. Other models with different objectives and/or architectures such as ALBERT (Lan et al., 2019), T5 (Raffel et al., 2020) and its multilingual version, mT5 (Xue et al., 2021), and GPT3 (Brown et al., 2020) were also introduced. More information about BERT-inspired LMs can be found in Rogers et al. (2020). Non-English LMs. Several models dedicated to individual languages other than English have been developed. These include AraBERT (Antoun et al., 2020) and Arabic BERT (Safaya et al., 2020) for Arabic, Bertie for Dutch (de Vries et al., 2019), Camembert (Martin et al., 2020) and Flau BERT (Le et al., 2020) for French, Pho BERT for Vietnamese (Nguyen and Tuan Nguyen, 2020), and the models presented by Virtanen et al. (2019) for Finnish, Dadas et al. (2020) for Polish, and Malmsten et al. (2020) for Swedish. Pyysalo et al. (2020) also create monolingual LMs for 42 languages exploiting Wikipedia data. Our models contributed to this growing work of dedicated LMs, and has the advantage of covering a wide range of dialects. Our MARBERT and MARBERT-v2 models are also trained with a massive scale social media dataset, endowing them with a remarkable ability for real-world downstream tasks. NLP Benchmarks. In recent years, several NLP benchmarks were designed for comparative evaluation of pre-trained LMs. For English, McCann et al. (2018) introduced NLP Decathlon (DecaNLP) which combines 10 common NLP datasets/tasks. Wang et al. (2018) proposed GLUE, a popular benchmark for evaluating nine NLP tasks. Wang et al. (2019) also presented SuperGLUE, a more challenging benchmark than GLUE covering seven tasks. In the cross-lingual setting, Hu et al. (2020), provide a Cross-lingual Transfer Evaluation of Multilingual Encoders (XTREME) benchmark for the evaluation of cross-lingual transfer learning covering nine tasks for 40 languages (12 language families). ARLUE complements these benchmarking efforts and is focused on Arabic and

its dialects. ARLUE is also diverse (involves 42 datasets) and challenging (our best ARLUE score is 77.40).

A screenshot of a graph

Description automatically generated with low confidence

**Mt5:** The model architecture and training procedure that we use for mT5 closely follows that of T5. Specifically, we base mT5 on the “T5.1.1” recipe,5 which improves upon T5 by using GeGLU nonlinearities (Shazeer, 2020), scaling both dmodel and dff instead of just dff in the larger models, and pre-training on unlabeled data only with no dropout. We refer to Raffel et al. (2020) for further details on T5. A major factor in pre-training multilingual models is how to sample data from each language. Ultimately, this choice is a zero-sum game: If low-resource languages are sampled too often, the model may overfit; if high-resource languages are not trained on enough, the model will underfit. We therefore take the approach used in (Devlin, 2018; Conneau et al., 2020; Arivazhagan et al., 2019) and boost lower-resource languages by sampling examples according to the probability p(L) ∝ |L| α, where p(L) is the probability of sampling text from a given language during pre-training and |L| is the number of examples in the language. The hyperparameter α (typically with α < 1) allows us to control how much to “boost” the probability of training on low-resource languages. Values used by prior work include α = 0.7 for mBERT (Devlin, 2018), α = 0.3 for XLM-R (Conneau et al., 2020), and α = 0.2 for MMNMT (Arivazhagan et al., 2019). We tried all three of these values (ablation results in section 4.2) and found α = 0.3 to give a reasonable compromise between performance on high- and low-resource languages. The fact that our model covers over 100 languages necessitates a larger vocabulary. Following XLM-R (Conneau et al., 2018), we increase the vocabulary size to 250,000 word pieces. As in T5, we use Sentence Piece (Kudo and Richardson, 2018; Kudo, 2018) models trained with the language sampling rates used during pre-training. To accommodate languages with large character sets like Chinese, we use a character coverage of 0.99999 and enable Sentence Piece’s “byte-fallback” feature to ensure that any string can be uniquely encoded.

A picture containing text, screenshot, number, parallel

Description automatically generated

**Ara gpt2:** English and Non-Arabic Language modeling GPT-1 (Radford et al., 2018) showed that Causal Language Modeling4 is an effective pre-training technique that improves a model’s generalization capabilities. GPT-2 then showed that using a larger model trained on a larger dataset surpasses the state-of-the-art of many tasks in a zero-shot setting, where a model solves a task without receiving any training on that task. Taking the scaling approach to the extreme led to the creation of GPT-3 (Brown et al., 2020), with 175 billion parameter model, also trained with CLM using terabytes of internet text. GPT-3 explored the idea of few-shot learning, where a model is given examples from a new task as a text prompt, which unlocks new capabilities at test time. It was later shown that a carefully designed GPT-3 prompt allows the model to generate website designs, scramble/unscramble words... 4 This is the regular Language Modeling objective where the model learns the probability of a word given the previous context. The CLM acronym is used to distinguish from masked language modeling (MLM). The advantage of scaling model sizes and training datasets comes with drawbacks, particularly the high computational cost.

**Methodology—** We have created this project with many scripts, we used Python programming language as it was the simplest and most used language in the field of AI because it is open source, easy language and have enormous number of packages and libraries that are very useful in our project field. We used Jupiter notebooks and visual studio code for our scripts; however, we used 3 environments to run those scripts due to the high processing needs some of these scripts required. We used Google drive and GitHub for version control and for merging work together.

### **Environments:**

1. **Localhost**

* While developing our scripts (preprocessing, training, and testing), we tried running them locally, but the processing needs for these scripts were high, so localhost wasn’t sufficient, it Takes too much time and lots of memory.

1. **Google Colab**

* Secondly, we tried Google colab, it worked for most of our scripts, however we faced lots of issues with it as the memory was very limited.

1. **Google Colab Pro**

* We tried Google colab pro as it was the best solution for the processing power needed as it provided V100 Tesla GPU with 40 Gb memory, 80 Gb ram and 150 Gb disk space.

1. **Kaggle Notebooks**

* Same as Google colab with additional TPU.

### **Packages & Libraries:**

1. **Pandas**: a powerful, fast, flexible, and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.
2. **Matplotlib**: a collection of command style functions that make matplotlib work like MATLAB, each pyplot function makes some change to figure.
3. **OS**: provides a way to interact with the operating system that Python is running on. This library offers a set of functions that allow you to perform various operations related to the file system, process management, environment variables, and other system-related tasks.
4. **NumPy:** Python library for performing numerical computations with

multidimensional arrays and matrices.

1. **Py torch**: often referred to simply as Torch, is an open-source machine learning library for Python that provides a fast and flexible way to build and train neural networks.
2. **Transformers:** state-of-the-art natural language processing library in Python that provides a simple and efficient way to train and deploy pre-trained models for a wide range of language tasks, including text classification, question answering, and language translation.
3. **Sentence Piece:** language-agnostic sub word tokenization library in C++ and Python that uses unsupervised machine learning to segment sentences into sub word units, which can be used to improve the performance of natural language processing tasks such as machine translation and text classification.
4. **Tqdm:** Python library that provides a fast and extensible way to add progress bars to loops and other iterable objects, making it easier to track the progress of long-running tasks and providing a more user-friendly experience for users.
5. **Json:** provides a simple way to encode and decode data in the JSON (JavaScript Object Notation) format, which is widely used for data exchange between web applications and services.
6. **TensorFlow:** open-source machine learning library in Python that provides a flexible and efficient way to build and train a wide range of machine learning and deep learning models, using both CPUs and GPUs, and is widely used in industry and academia.
7. **Scikit-learn:** often abbreviated as sklearn, is a popular machine learning library in Python that provides a wide range of tools for data preprocessing, feature engineering, model selection, and performance evaluation, making it easier to build and deploy machine learning models in real-world applications.
8. **NLTK:** Python library that provides a suite of tools and resources for natural language processing (NLP), including tokenization, stemming, tagging, parsing, semantic reasoning, and corpus analysis, making it easier to process and analyze human language data.
9. **Re**: provides a set of functions for working with regular expressions, which are powerful and flexible patterns that can be used to match and manipulate text data, making it easier to process and extract relevant information from large amounts of unstructured text data.
10. **Joblib:** provides tools for easy and efficient parallel computing, including fast and efficient memory-mapped file storage, task scheduling, and caching, making it easier to speed up and optimize computationally intensive tasks in machine learning and other data-driven applications.
11. **Selenium:** widely used open-source library that provides a suite of tools and APIs for automating web browsers, making it easier for developers to test and interact with web applications
12. **Beautiful Soup:** popular Python library used for web scraping purposes, providing a convenient way to parse HTML and XML documents and extract useful information from them.

## **Data Preprocessing:**

### **Glare Dataset Preprocessing:**

* We split the dataset into several small csv files to be able to work with it as it was exceeding 20 Gb, every small file was 5000000 rows.
* There was a separate file containing the category of each review linked with the dataset with unique id for each row, we applied a function to get the specific category for each review in the dataset.
* Then we worked on the rating column (sentiment analysis) to restrict its values from 5 to 3 categories only.
* Then we filtered the data set to only reviews which have reply content text (text generation).
* Then we got only the Arabic reviews and reply content text and remove any other rows containing any other language.
* We end up with dataset labeled with category, rating, and recommended solution text.

### **Merged Topics Data preprocessing:**

* We have searched for many datasets for topic classification task, they were in different domains, so we merged many datasets together to form one dataset that has 140k + rows to use in topic classification task.
* We have balanced the dataset categories to be in the same range.

### **Web Scrapping Module:**

* Due to the lack of datasets in the task of recommending solution (text generation) in our domain we have decided to apply web scrapping techniques to get new data.
* We have scrapped open website (دليل الخدمات العامة) to get data in form of citizen problem (or service), solution & steps to get this service and governmental sector for each service.
* We used selenium and beautiful soup libraries to get these data.
* We ended up with 433 rows of data and augmented our data by using ChatGPT to produce the same form of data in our domain, lastly, we got 930+ rows of data.

### **Input Text Preprocessing Module:**

* Text preprocessing pipeline contains the following functions.

**Handling Punctuation Function**: this function removes all punctuation.

**Stop words Removal Function**: this function removes all Arabic stop words.

**Lemmatization Function**: this function applies Arabic lemmatization to text.

## **Topic classification Module:**

**Models Used:**

1. MARBERT: is a large-scale pre-trained masked language model focused on both Dialectal Arabic (DA) and MSA provided by UBC\NLP.
2. Arabic-MARBERT-news-article-classification is a news article classification model that was built by fine-tuning the MARBERT model provided by Ammar-Alhaj-ali.

**Data Preparation:**

* **Corpus Collection:** The first step in data preparation for text classification is to collect a comprehensive and representative dataset. The dataset should cover a wide range of instances that are relevant to the classification task at hand.
* **Text Preprocessing:** Involves transforming raw text into a more structured and suitable format for classification.
* Key preprocessing techniques include:

1. **Text Cleaning:** Removing noise, such as HTML tags, special characters, punctuation, and irrelevant content from the text data.

2. **Tokenization:** Breaking down the text into smaller units called tokens, such as words, sub words, or characters. Tokenization allows for better analysis and feature extraction.

3.**StopWordRemoval:** Eliminating common and insignificant words, known as stop words (e.g., "the," "and" "is"), which do not carry much semantic meaning.

4.**LemmatizationandStemming**: Reducing words to their base or root form (lemmatization) or stripping them down to their core (stemming). This helps in reducing vocabulary size and capturing the essence of words.

* **Feature Extraction:** involves transforming the preprocessed text data into numerical representations using the selected model’s embeddings.
* **Label Encoding:** Assigning unique integer values to each label, converting “Categories” column into numerical representations.

**Training Procedure:**

* **Splitting the Dataset:** Divide the processed dataset into training, validation, and test sets.
* **Model Selection:** We choose for training procedure MARBERT model.
* **Model Training:** Train the MARBERT model on the preprocessed data using the Trainer library.
* **Hyperparameter Tuning:** We fine-tune the model's hyperparameters to achieve better performance. define the “AdamW” optimizer and learning rate scheduler starting with “LR=5e-5”, batch size “train\_batch\_size=32”, and the number of epochs “Num epochs = 5”. This process involves experimentation and validation using the validation set to find the optimal combination of hyperparameters that results in the best performance.

**Evaluation Procedure**:

* **Evaluation Metrics**: We use metrics such as Macro F1-score, accuracy, precision, and recall can be used to assess how well the model generalizes to unseen data. We evaluate the trained model's performance using the validation set.

## **Sentiment Analysis Module**

## **Models Used**:

1.Marbert model was used provided by UBC\NLP

2.Arabert model provided by AUB\mind lab

Both of which employ a multi-layered Transformer consisting of multiple encoder layers, each layer contains self-attention mechanisms and feed-forward neural networks.

And after extensive experiments with both models the best results was

provided with Arabert model.

Arabert, like other language models, requires preprocessed data for training and fine-tuning. The preprocessing steps for Arabert involve tokenization, sub word segmentation, and the addition of special tokens. Here's an overview of how Arabert preprocesses data:

* **Tokenization:** The first step in preprocessing Arabic data for Arabert is tokenization. Tokenization involves splitting the text into individual tokens, such as words, sub words, or characters. Tokenization is necessary to create input sequences that can be processed by the model. Arabert typically uses a tokenizer specifically designed for Arabic, which considers the unique characteristics of the language.
* **Sub word Segmentation:** Arabic words often exhibit complex morphology, with different forms and variations. To handle this, Arabert models often employ sub word segmentation. Sub word segmentation breaks down words into smaller meaningful units, such as root letters, prefixes, suffixes, and diacritics. This helps the model handle morphological variations and improves its ability to capture the meaning of Arabic words.
* **Special Tokens:** Arabert introduces special tokens to mark the boundaries of the input sequence and provide additional information to the model. These special tokens include:
  + **CLS:** The classification token, which is added at the beginning of the input sequence. It helps the model understand that the input is meant for a classification task.
  + **SEP**: The separator token, which is inserted between two consecutive sentences or segments of the input.
  + **PAD**: The padding token, used to pad the input sequences to a fixed length for efficient batch processing.
  + **MASK**: The mask token, which is used during pre-training to represent masked or missing tokens that the model tries to predict.
* **Input Formatting:** Arabert models expect a specific input format. The preprocessed input consists of tokenized and sub word segmented sequences, with the special tokens added. The input is often represented as token IDs or embedding indices that correspond to the vocabulary of the model.

These preprocessing steps help prepare the Arabic data for training or fine-tuning Arabert models. The preprocessed data is then fed into the model during the training or inference phase to learn contextualized representations or perform NLP tasks in Arabic. The specific implementation details may vary depending on the Arabert variant and the preprocessing tools used.

* When fine-tuning Arabert for sentiment analysis, several functions are commonly used to optimize the model's performance and adapt it to the specific task. Here is a description of the functions typically employed during the fine-tuning process for sentiment analysis with Arabert:
  + **Data Preparation:** This function involves preparing the sentiment analysis dataset for training. It includes tasks such as data cleaning, preprocessing, and tokenization. The text data is transformed into a format suitable for input to the model, with labels indicating the sentiment category (positive, negative, neutral, etc.) associated with each example.
  + **Model Architecture:** Arabert utilizes a transformer-based architecture, such as BERT, as its foundation. The model architecture function initializes the Arabert model, configuring its layers, attention mechanisms, and parameters. It sets up the basic structure and configuration of the model for sentiment analysis.
  + **Loss Function:** The loss function measures the discrepancy between the predicted sentiment and the true sentiment labels in the training data. we have used loss functions for sentiment analysis like categorical cross-entropy or binary cross-entropy. The loss function guides the model's learning process by providing feedback on its performance.
  + **Optimization Algorithm:** we have used stochastic gradient descent (SGD) algorithm and its variants, such as Adam or AdaGrad. The optimization algorithm adjusts the model's parameters based on the gradients calculated from the loss function, aiming to minimize the loss and improve the model's performance.
  + **Training Loop:** The training loop function iterates over the training data, feeding it to the model and optimizing the model's parameters. It consists of steps such as forward propagation, loss calculation, backward propagation (gradient computation), parameter updates, and repetition of these steps for a specified number of epochs. The training loop ensures that the model learns from the data and gradually improves its sentiment analysis capabilities.
  + **Evaluation Metrics:** we have used evaluation metrics for sentiment analysis such as accuracy, precision, recall, F1 score.

**Hyperparameter Tuning**: we have experimented with various Hyperparameters such as learning rate, batch size, number of layers, dropout rate, and others. These values significantly impact the model’s training dynamics and overall performance.

## **Recommended Solution generation Module:**

After the input review text has been through the text preprocessing pipeline, it is ready to be moved through this module to produce the governmental sector and recommended solution for the problem issued by the user.In this module we have been through two NLP techniques which are:

### **Text Generation:**

**Models Used:**

1. **Google\Mt5:** a multilingual sequence-to-sequence transformer that is pretrained on very big multilingual data from which is Arabic.
2. **UBC\NLP Arat5:** a t5 based transformer that is pretrained only on Arabic language using very large Arabic datasets.
3. **Aubmind\lab Aragpt2:** a gpt based transformer for sequence-to-sequence generation but pretrained on Arabic language.

**Data Preparation**

* The preprocessed input text is then prepared for training by getting tokenized using the model’s tokenizer, then we extract the feature using the transformer model representation (using attention mask) to be able to be passed to the architecture of the model.

**Training Procedure**

* **Fine Tuning:** We have fine-tuned the sequence-to-sequence transformer models on our dataset to apply transfer learning from the pretrained model on our task and domain.
* **Training loop:** We used the transformer Trainer library and trained the model using PyTorch, we can find snippet of training function code in Figure 33
* **Loss Function:** Cross-entropy loss was used in our model as it is the most famous loss function in sequence-to-sequence generation as it proved the best results.
* **Optimization Algorithm:** we have used stochastic gradient descent (SGD) algorithm and its variants, such as Adam or AdaGrad. The optimization algorithm adjusts the model's parameters based on the gradients calculated from the loss function, aiming to minimize the loss and improve the model's performance.
* **Evaluation Metrics:** we used BLEU score for evaluation as it is a metric that measures the similarity between a machine-generated text and one or more human-generated reference texts, based on the number of overlapping n-grams (sequences of adjacent words) between the machine-generated text and the reference texts.

### **Information Retrieval :**

After trying the text generation techniques, we found that it didn’t get the desired result and it was outside the domain of our project, we just wanted to prove the concept, then we studied the Information Retrieval techniques, and it gets the best results.

**Techniques used:**

1. Tf-idf representation
2. Sentence cosine similarity

**Algorithm:**

**Feature Extraction:** we get the preprocessed text and apply feature extraction technique like tf-idf, bag of words

transformer models embeddings and others but the best was tf-idf due to the lack of data that prevent us from using deep learning, and the power of tf-idf in representing the frequency of important words in text, this procedure is done on the documents in database and the query of user

* **Similarity check:** here we check the similarity between the query and each document in database, we used cosine similarity from sklearn library as it is the most used metric in this task and proved the best results, this process code is shown in Figure 35

**Ranking: then** we rank the documents retrieved by the query by their scores and get only the top one score document.

### **Experiments & Results:**

### **Topic classification:**

**A table with numbers and letters

Description automatically generated**

After large number of experiments using different models, applying different text preprocessing, and applying hyperparameter tuning we concluded that experiment 7 using marbert model and preprocessed Merged topics dataset gives the best f1 score as shown in Table 4.

A black background with white text

Description automatically generatedA table with numbers and letters

Description automatically generated**sentiment analysis:**

After large number of experiments using different models, applying different text preprocessing, and applying. hyperparameter, tuning we concluded that experiment 5. Using arabert model and preprocessed combination between100k reviews dataset and glare dataset, gives the best f1 score.

##### **Text Generation:**

##### A screenshot of a computer Description automatically generated

After a large number of experiments using different models, applying different text preprocessing and applying hyperparameter tuning we concluded that experiment 3.

using arat5 model and preprocessed glare dataset, gives the best bleu score as shown in Table 6.

### **Information Retrieval:**

### A screenshot of a computer Description automatically generatedHere are some outputs from our algorithm shown in

##### Acknowledgment

We would like to use this opportunity to express our gratitude to everyone who supported this project.

We are thankful for their indispensable guidance, invaluably constructive criticism, friendly advice, and the most obliged provision of their genuine and illuminating views.

We would like to offer our special thanks to our supervisor Dr. Ahmed Salah for his help and support.

We would also like to extend our thanks to T.A. Moataz Mohamed for guiding and helping us throughout the whole course of the project; his help was invaluable to us.

##### References

##### [1]Abdul-Mageed, M., Elmadany, A. and Nagoudi, E.M.B., 2020. ARBERT & MARBERT: deep bidirectional transformers for Arabic. arXiv preprint arXiv:2101.01785.

##### [2]Antoun, W., Baly, F. and Hajj, H., 2020. Arabert: Transformer-based model for arabic language understanding. arXiv preprint arXiv:2003.00104.

##### [3]Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A. and Raffel, C., 2020. mT5: A massively multilingual pre-trained text-to-text transformer. arXiv preprint arXiv:2010.11934.

##### [4]Nagoudi, E.M.B., Elmadany, A. and Abdul-Mageed, M., 2021. AraT5: Textto-text transformers for Arabic language generation. arXiv preprint arXiv:2109.12068.

##### [5]Antoun, W., Baly, F. and Hajj, H., 2020. AraGPT2: Pre-trained transformer for Arabic language generation. arXiv preprint arXiv:2012.15520.

##### [6]AlGhamdi, F., Mohammed, R. and Alowisheq13, H.A.K.A., GLARE: Google Apps Arabic Reviews Dataset.

##### [7]Omari, M.A., 2022. OCLAR: logistic regression optimization for Arabic customers' reviews. International Journal of Business Intelligence and Data Mining, 20(3), pp.251-273.

##### [8]Kobayashi, M. and Takeda, K., 2000. Information retrieval on the web. ACM computing surveys (CSUR), 32(2), pp.144-173.

##### [9]Al-Ayyoub, M., Khamaiseh, A.A., Jararweh, Y. and Al-Kabi, M.N., 2019. A comprehensive survey of arabic sentiment analysis. Information processing & management, 56(2), pp.320-342.

##### [10] Alruily, M., 2021. Classification of arabic tweets: A review. Electronics, 10(10), p.1143. 73 | P a g e

##### [11] Zakraoui, J., AlJa'am, J.M. and Salah, I., 2022, December. DomainSpecific Text Generation for Arabic Text Summarization. In 2022 International Conference on Computer and Applications (ICCA) (pp. 1-4). IEEE.

##### [12] Ameen, A., Alfalasi, K., Gazem, N.A. and Isaac, O., 2019, December. Impact of system quality, information quality, and service quality on actual usage of smart government. In 2019 first international conference of intelligent computing and engineering (ICOICE) (pp. 1-6). IEEE.

##### [13] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pretraining of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

##### [14] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692