Efficient Arabic Query Auto-Completion for Question Answering at a University

Momin Abdelhafez, Ghaydaa Khateeb and Adnan Yahya
Computer Engineering Department

Birzeit University
Birzeit, Palestine
yahya.birzeit.edu

***Abstract*— In this paper we describe an implementation of an Arabic query auto-completion system for student question-answering at a University. University students make many inquiries concerning academic life: about majors, concentrations, dates, instructors, courses, rooms, exams and more. Auto-completion (AC) has recently been part of many user interfaces, such as search bars on web pages, social media sites and mobile applications. We investigate multiple approaches to completion candidate generation and ranking and the role Arabic NLP may play in that. After experimenting with other options, we collected the data used in our system directly from students at the University. This data can be expanded to account for more types of queries and ways to express information needs. We describe our dataset, give an evaluation of individual system components and of the system results in general. Our goal is to improve the answer search experience by reducing the time of entering the query and biasing the completed query towards unambiguous and easily answerable questions. We divided the implementation into different stages and came up with the results for each stage separately. The results we got were very good compared with other QAC systems and it can improve the QAC used for the Arabic language in general. Finally, we give an informal evaluation of overall results and the improvement resulting from using QAC for our QA system.**

I. INTRODUCTION

Auto-completion (AC) has recently been part of many modern user interfaces [1], such as search bars on web pages, social networking sites, numerous mobile applications and many text editors and IDEs (e.g. VS code, sublime) [2]. Users are more likely to utilize search engines that grasp their goal and save time and effort by anticipating their information needs and helping formulate the queries that truly reflect these needs. Users prefer search tools with such characteristics because they eliminate the need to struggle with crafting the query conveying the information required and avoid typing difficulties especially on small devices. The query completion benefit is not limited to recognizing the user's need, but may also assist in better answering his/her questions by making it more specific to the issuing user or time of issuance and biasing the query formulation so that it is unambiguous and can be answered efficiently [3].

A. Query Auto-Completion (QAC)

Auto-completion works by taking what the user is typing and anticipating his/her need and automatically completing the task of issuing the query/question expressing the information need. QAC is common in modern search engines, as it can enhance the search experience by reducing the time it takes to enter a query by significantly reducing the number of characters typed by the user. It also reduces the number of typing errors by simply offering the user to choose one of the suggestions displayed [4]. Auto-replace is a concept related to auto-completion in which specified abbreviations are typed and the system automatically replaces them with full-blown strings. The QAC feature is useful in many domains, including web and eCommerce searches where using this feature, users can select full searches based on a prefix of a few characters. There are two steps in QAC: starting with a prefix entered by the user, a number of completions is generated to form the suggestion list [5]. In the second step, the suggestions are ranked based on the system assessment of their relevance to the user information need and the top ones are displayed to the user.

B. Problem Description

University students make many inquiries concerning academic life: about majors, concentrations, dates, instructors, courses, rooms, exams and more. A system that completes student questions by mapping their partial queries to standard ones that can be translated to executable structured query language (such as SQL or SPARQL) can be of great help.

Students ask their questions on social media sites like Facebook in a variety of ways.

QA is not straightforward especially if the user can ask unrestricted questions in natural language. AC may be utilized to limit user question variations by suggesting specific completions for a given prefix. The adopted completion can then be translated into a structured query language to get the answer easily. Ideally, we need to take all possible ways a user may issue a query and map them to the standard one: only stanard queries are given as a suggestions.

A system that gives students direct answers to their questions can be very useful and can be integrated into the University portal. There are some important points that should be present in such a system:

1) The queries in the suggestion list should be ranked so that the intended question should be among the

top n (say 3) in the list.
2) The system should be adaptive, by learning from correct (user adopted) and incorrect (user rejected) suggestions.
3) Two stages are included in the QAC [6]: the generation stage, where we identify potential completions for the partially entered query. Then the ranking of the generated potential completions, based on criteria like relevance to entered prefix, User Profile, popularity or using learning-based approaches such as Neural Language Modeling. We will talk about the details in the Methodology section.

II. BACKGROUND AND RELATED WORK

In this section, we discuss aspects of QAC and QA related to the system we are developing. First we discuss two approaches to QAC: traditional/heuristic then learning/language-based approaches. We also talk about QA and finally the datasets for QAC.

A. Heuristic Approaches to Query Auto-Completion (QAC)

1) Probabilistic QAC Approaches: These approaches try to compute the probability $P(q_c|p,t,u)$ of generating a query completion q_c when the user types prefix p, taking into account the time t and the user u [4].

2) General QAC Models: An ideal QAC system generates candidates ordered according to their chance of matching the user's information need. The Maximum Likelihood Estimation (MLE) is determined based on the past popularity (i.e., frequency as computed from logs) of queries. [7] refer to this type of ranking as the most popular completion (MPC):

$$MPC(p) = arg \max_{q \in C(P)} w(q),\ where\ w(q) = \frac{f(q)}{\sum_{q_i \in Q} f(q_i)}$$

Where $f(q)$ is the frequency of query q in search log Q, and $C(p)$ is a collection of all query completions that begin with the prefix p.

3) Time-sensitive QAC Models: The assumption in the previous equation is that the population of queries does not change over time. The ranks will be the same independent of time and completions will be based only on historical popularity. For our system, the time factor is important because certain queries occur frequently at given times, like the exam period, registration dates, and so on, as shown in the following equation:

$$MPC_t(p) = arg \max_{q \in C(P)} w(q), where\ w(q) = \frac{\alpha f(q)}{\sum_{q_i \in Q} f(q_i)}$$

were α is the timing factor that changes according to the current period of time for query q: if the current period of time matches the the query q, the value of α increases.

4) User-centered QAC Models: QAC techniques leverage information about a user's search history and interactions with a search engine to rank the proposed completions. QAC efficacy is boosted when the user's specific context is taken into account.

B. Learning/language-based QAC Approaches

1) Learning-based QAC Approaches: The learning-based QAC techniques should take both time-related and user interactions into consideration. Because there is a lot of ambiguity in the semantic relations (more than one word has the same meaning or the text may be read in many ways), extracting characteristics from the Arabic text is not a simple procedure. To do so, we may use:

- Arabic Ontology [8].
- Lexical syntactic patterns [9].
- Automatic extraction of relationships [10].

2) Neural Language Modeling: In neural language models, the probability of a text sequence is measured, for example, if we have a sequence of words, we can find the predicted next word based on the embeddings of the previous several words. For this purpose, the Recurrent Neural Networks (RNN) [11] are used to summarize sequence of words into different states and apply the RNN at each state. The problem with neural language modeling approaches is that they are time-consuming in the training stage because they compute costly normalization terms which demand iterating over all words in the computation of word probability. To resolve this issue, un-normalized language models may solve the problem by approximating the normalization over the whole vocabulary in word probability, so these language models are the efficient language models for QAC ranking [12]. The Long Short Term Memory (LSTM) is an enhanced version of RNN that resolved the exploding and vanishing gradients problems of the RNN [13].
We adopted LSTM deep learning approach as it worked well for QAC, for example with the AOL dataset[1] [13].

C. Arabic QAC

Despite the presence of Arabic QAC in search engines, researches on Arabic QAC are few, so we are trying to increase the efficiency of Arabic QAC, and hence the quality of Arabic Question Answering (QA).

D. Question Answering (QA)

The massive expansion of online information necessitates the development of Query Answering systems to bridge the gap between end-users and the many data representations on the Web. QA is the process of responding to users' questions and providing definitive and precise answers to their inquiries rather than documents containing the answer. The following are examples of questions that may be asked:

- Who were the 10 richest people in the world in the year that Germany last won the Football World Cup?
- How many times did National University rank in the top 2% in QS World University Rankings?

There are several types of QA:

[1] https://www.kaggle.com/datasets/dineshydv/aol-user-session-collection-500k

- Free text only: the query is issued in natural language and the answers are extracted from the text.
- Structured SQL query and data stored in a database.
- Un-structured (free) Query and data stored in a database.

Clearly, answering queries from a database is most promising in terms of answer precision. The problem is that much of the web data is not structured. A possible solution to answer queries from free text is to use the methods of information extraction (IE) to convert textual data into database tuples that can be queried using database style queries. By translating free text queries to a structured language (such as SQL or SPARQL) one is able to return exact responses from the database.

The QA system we are developing here relies on database queries formulated based on natural language questions asked by the user using QAC.

General QA systems are complicated and demanding especially when the language is challenging like the Arabic language.

Difference Between Question Answering and Information Retrieval: In IR, the system returns documents that are related to the input query by measuring the distance between the possibly modified query and the document. In QA, the process can be done either by translating the standard query into structured query language [3], to get the answer from the database directly or using IE, to extract the answer from some text [14]. This the approach we use here, with the database being the University information system. In the absence of a database, the QA must isolate the exact components of the document that are the best candidates for answers and return them as potential answers. Of course, both QA and IR can benefit greatly from AC, and IR systems utilize query expansion (reformulation) extensively.

Question Answering and Database Querying: In order to return a specific answer for the asked question, the input user query should be translated to an executable structured query language [3]. Figure 1 shows the Arabic query and its mapped SQL query.

- ما هي متطلبات مساق الذكاء الاصطناعي؟
- **SELECT** prerequisites **FROM** subjects **WHERE** subjects. Name = "AI"

Fig. 1: Arabic Question to SQL Query

E. Auto-completion Effect on Question Answering

With our type of AC, we guide the user to select one of the suggested queries which can be easily translated to a structured query language, allowing the system to quickly return a specific answer. Without AC, the problem would be more complicated because the large number of possible natural language queries cannot be easily translated to structured query language. Our challenge is to find a mapping from user entered partial queries to standard queries easily translatable to SQL.

F. Datasets

1) General QAC Datasets: AOL dataset is the most popular dataset used for QAC. This collection consists of 20M web queries collected from 650k users over three months. The goal of this collection is to provide real query log data from real users. It can be used for personalization, query reformulation or other IR tasks.

2) Our QAC Dataset: For our system, the dataset should contain the following:

- The standard questions that will be translated to structured query language.
- Each standard query has variations of queries that students usually ask.
- Each standard query should be combined with additional parameters: studying level of student asking the question and the period of time the question usually asked, that may be used for better ranking.

Figure 2 depicts several options for posing question expressing the following information need: "What are the Operating System (OS) course prerequisites?"

- ما هو متطلب مساق أنظمة التشغيل
  - شو لازم اوخذ قبل مساق الـ OS
  - متى لازم أنزل مساق الـ OS
  - شو متطلبات مساق الـ OS
  - ايش بفتح OS

Fig. 2: Different Ways for Asking a Specific Question

We tried two approaches for collecting the dataset:

- Data scraping from social media sites.
- Google form that contains several standard and students were asked to add variations reflecting ways they may ask each standard question. We elaborate on the details of dataset collection later in this paper.

III. METHODOLOGY

In this section, we describe the design, implementation and testing for our QAC system depicted in Figure 3.

A. Design Flow

According to the diagram in Figure 3, the student starts using the system by entering his/her information: student ID and major/minor.

- **Student ID**: from the student ID, we can infer the study Level. According to this information, we can suggest relevant entities (courses, instructors, etc) because our data stores majors, instructors, and subjects and the corresponding levels. We can extract the level of the course from the course number, for example ENCS332 indicates that this course is a third year course (first digit is 3) and the number of credits (the

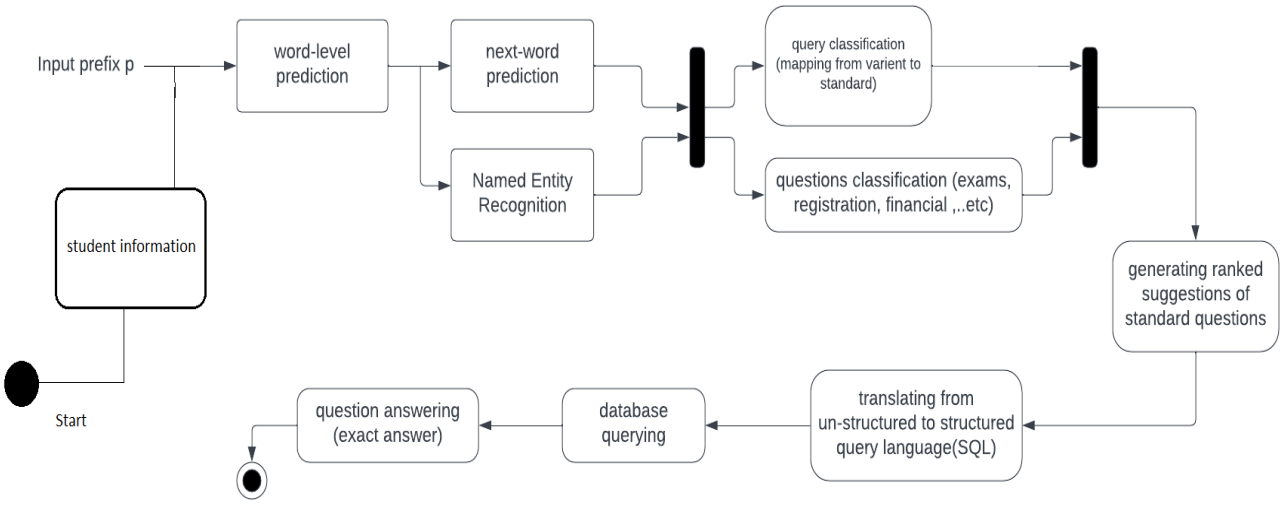

Fig. 3: QAC System Design

second digit). Figure 5 shows the type of data we may have readily available (instructor and courses taught).

- **Major/Minor**: we can make suggestions related to student major/minor according to the data in figure 5 and major/minor study plans.

Then the student starts entering the query, and the following elements are activated:

*1) **Word-Level Completion**:* As the student enters a prefix in the search bar, the system begins the word-level completion. Our word-level system generates the 10 most frequent words that start with the entered prefix. The frequency is based on the query variants we have, but may also be based on that individual usage history.

*2) **NER**:* Each of the 10 possible completions, generated for the current prefix, is processed using the NER model. The model is activated with every new entered/predicted word. So, if the student entered a new word, the entire input prefix is processed by our NER model and the same for the predicted word from the next-word prediction model. These classifications are used for generating suggestions/completions and ranking them, for example, if the student entered an entity name like the instructor name, the suggestions will be related to the entity name entered by the student.

*3) **Next-word Prediction**:* We predict the next n words for each one of our 10 completions. We'll determine the best value for n after experimentation, by setting n = 3 and see the quality of our classification model (next section), n = 4, n = 5, and so on. If n = 5 and the entered prefix consists of m words where $m < n$, we'll predict the next $5 - m$ words, and if the number of entered words more than 5, we'll predict the next word, because n = 5 is the best value of n to classify the variant queries.The prediction will form a tree of words, each path from the root node to leaf node will form a variant query. The root is the entered prefix, that consists of k words, where k is the number of the entered words by the student. When the student tries to enter any word, our word-level prediction model predicts its completion, and the next-word prediction reformulate the prediction when the word becomes ready. The prediction quality becomes better when the number of previous words increases because the model can detect the dependencies between the next word and the previous words. The student's entered query has more priority than the predictions, and the suggestions will become less each time the student adds more information about the questions that he wants to ask (limiting his need). The following example in figure 4 shows the expected tree of the model when the student enters the word: مين.

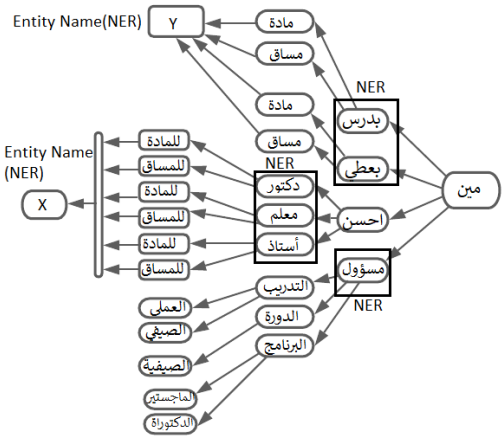

Fig. 4: Arabic Question to SQL Query

*4) **Mapping variants to standard queries**:* We have 10 variants of queries, each of them consisting of n words, where n is the best number of words for each variant in which it's possible for the system to map this variant to its standard suggestion and each one of them is classified

A	B
تخصص هندسة الحاسوب	المواد
عدنان حسين محمود يحيى	انظمة التشغيل, او اس, أو أس, OS, ENCS339
	مختبر الالكترونيات الرقمية وتنظيم الحاسب, مختبر ديجيتال, مختبر اورجا, لاب ديجيتال, لاب اورجا, ENCS211
	الذكاء الاصطناعي, AI, ENCS434
	أخلاقيات هندسة الحاسوب, اخلاقيات, ايثكس, أخلاقيات, ايثكس, ENCS521
	مقدمة مشروع التخرج, مقدمة, ENCS520
	مشروع التخرج, ENCS530
	ان ال بي, nlp, NLP, معالجة اللغات الطبيعية, إن آل بي, أن آل بي, ENCS539
ابو السعود أحمد ابو السعود حنني	مختبر الالكترونيات الرقمية وتنظيم الحاسب, مختبر ديجيتال, مختبر اورجا, لاب ديجيتال, لاب اورجا, ENCS211
	تنظيم الحاسوب والمعالج الدقيق , أورجا, اورجا, مايكرو, كمبيوتر اورجانيزيشن, ENCS238
	مختبر تطبيقات الحاسوب, لاب مايكرو, مختبر مايكرو, ENCS411
	مقدمة مشروع التخرج, مقدمة, ENCS520
	مشروع التخرج, ENCS530

Fig. 5: Major-related Data

to its standard query. Standard queries contain variables representing entities like Instructor names, TAs and Courses. These symbols are replaced with specific entities based on the student study level. The process is repeated with changes to the the entered prefix because student input is given high priority in classification. For example, if the user rejects suggestions and completes his word manually, next-word prediction will be based on the manually completed word, and if the word was an entity (Course, Instructor, etc) the standard completions will be related to the entered entity.

*5) **Question Classification**:* Each of the standard queries is manually classified to a class (exams-related, registration-related, etc). The standard completions may be re-ranked based on factors like the current time of the semester.

*6) **Translating Standard Queries to Structured Query Language**:* Now, when the student chooses one of the suggestions, our QA model maps the selection to a structured query (e,g, SQL) to return the specific answer from our database. The selection click is stored to calculate the Click Through Rate (CTR) which is used for system evaluation as elaborated later. Figure 6 shows the output for prefix: ميدتي of the word: ميدتيرم entered by a final year computer engineering student in the midterm exams period of the semester.

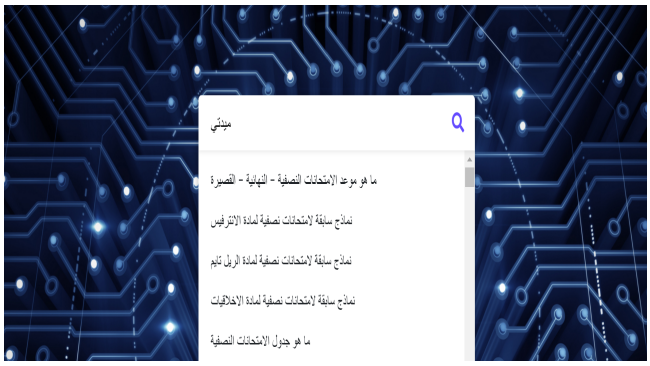

Fig. 6: QAC System Output

B. Dataset Collection

For the dataset building, we tried two methods: data scrapping of student queries from popular social media sites and direct data collection from students by filling forms.

1) Data Scraping of Student Queries from Social Media Sites: Social media sites recently have become the most common communication platform for students. There are several groups where students usually ask their questions about majors, exams, instructors, etc. So, we tried to scrap data from these sites as follows:

- A python script was written to scrap the content of potentially relevant posts.
- The scrapped data were stored in an excel file.
- Although there are some posts with more than 50 words, we scrapped posts with a max of 20 words, because we estimated that larger posts are probably not related to questions about university matters.

We found that the scrapped data needed substantial pre-processing, and we need to classify it between queries related to our system, and those related to matters not within our scope like elections, political matters, and administrative matters. Also, the data on the scrapped sites is frequently deleted, so its quantity is somewhat small. That's why we decided to use the direct approach to get a cleaner dataset.

2) Direct Data Collection from Students: In this approach, we created a google form and shared with students. The form contains standard queries in English to avoid biasing student responses. Students were asked to express the English question in Arabic, and in as many variations as they can. Figure 7 shows samples of our collected data. So far, we have received about 35 responses (from 30 students) with 45 standard queries and 25 variations for each, on average.

1	Standard	Variants
2	هل يعتبر الحضور والغياب مهم للمعلم X	دكتور X بركز على الحضور
3		الدكتور X بوخذ حضور و غياب
4		الدكتور X بيهتم في الحضور
5		بوخد الدكتور X حضور وغياب
6		بركز الأستاذ X عالحضور
7		الأستاذ X باخد حضور دايماً
8		دكتور X بهتم للحضور
9		بوخذ حضور الأستاذ X
10		بوخذ الدكتور X حضور كل محاضرة
11		كيف الدكتور X بالحضور
12		بركز الدكتور X على الحضور
13		الدكتور X بوخذ حضور دايماً
14		الدكتورX بدقق كثير ع الغياب او مش فارقة معه

Fig. 7: Dataset Samples

So our dataset consists of standard queries (45), variants (25 per standard query) plus the following additional parameters:

- Variant queries have variables representing the entities used like instructor name, year of studying, subject name, etc.
- Student level as a parameter to represent at which year the query is asked the most. Each standard query has this parameter (assigned manually). The student fills his personal data which includes his studying level, as a result, the suggestions ranking will be changed accordingly. For example, a student in his final year is supposed to see some suggestions related to graduation projects course (usually taken in the final year of study).
- The asking time parameter represents the time of year the query is asked the most. The time is assigned

manually for each standard query, and according to the time that the student uses the system, the suggestions ranking will be changed. For example, if the student uses the system in June he is supposed to see some suggestions related to exams.

C. QAC Word Prediction and Named Entity Recognition (NER)

1) Word Completion: We used the Trie data structure[2] for completing words from its prefix (set of input characters). The trie data structure is represented as a tree, each node contains word prefix. According to the entered prefix, all possible words will be formed by going through the tree to its all possible nodes. We built a trie for all the vocabulary (unique words) in our dataset. The frequency of each word was used to rank the words so the word with highest frequency will be ranked first. Figure 8 shows the construction of the Trie data structure.

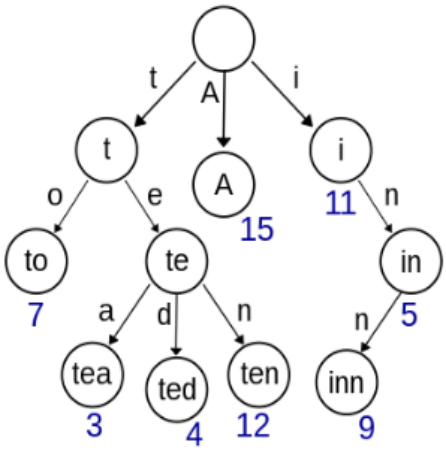

Fig. 8: Trie Data Dtructure

2) NER: The output queries have variables representing entities. The reason for that is when the student starts entering the query, our NER model starts classifying the words to NEs like INSTRUCTOR, TA, COURSE, etc., and each one of these entities is represented by a variable. This variable will be replaced by the correct entity based on the information of the issuing student (major, studying year, etc), the current time of the semester (midterm/final exams dates, registration, courses withdrawal and addition) as well as the Instructors and their courses in the current semester. Table I shows the entities and their mapped tags in the NER model.

TABLE I: NER Classes

Entity	Tag
Instructor	INSTRUCTOR
Teaching assistant	TA
Subject	COURSE
Evaluation (exams, projects)	EVAL

We tested our dataset with two approaches: CAMeL Tools with rule-based and machine learning techniques, we adopted the approach that achieved best results.

NER with CAMeL Tools: CAMeL tools are a collection of open-source tools for Arabic Natural Language Processing (NLP), for NER using ANERcorp [15] dataset.

CAMeL Tools with Rule-Based NER: As our system is for universities, the number of entities is limited, so CAMeL library with rules is a good solution. We have different sets of instructors, teaching assistants, courses, etc. So we can distinguish these entities easily once the CAMeL detects a person's name (PER) or overall entity (O). When CAMeL detects person name (PER) it should be either instructor's name or TA name. Based on our lists of TAs and instructors we can make that distinction easily. In addition to the lists, we used the previous words to detect the entity, (e,g. دكتور، بروفيسور for Instructor, أستاذ، مساعد، معلم for TA, and مادة، مساق، كورس for COURSE). The same for overall entities which should be either course or evaluation: we built a list that stored all possible words used for courses and evaluations, also we relied on the previous words. Out of 130 words, our model classified 125 correctly with a success rate of 96%. Figure 9 shows a sample output of our NER model. Table II shows the percentage of correct classifications for a

```
<=====<=====<====<=====<=====<=====<====<====<====
بغلب مشروع التخرج مع الدكتور احمد السعدة
                                        بغلب: OTHER
                                      مشروع: COURSE_1
                                     التخرج: COURSE_2
                                         مع: OTHER
                                     الدكتور: OTHER
                                     احمد: INSTRUCTOR_1
                                     السعدة: INSTRUCTOR_2
<=====<=====<====<=====<=====<=====<====<====<====
في مجال انشر مشروع التخرج مع الدكتور حكم
                                         في: OTHER
                                      مجال: OTHER
                                      انشر: OTHER
                                      مشروع: COURSE_1
                                     التخرج: COURSE_2
                                         مع: OTHER
                                     الدكتور: OTHER
                                     حكم: INSTRUCTOR_1
<=====<=====<====<=====<=====<=====<====<====<====
```

Fig. 9: NER Rule-based Output

set of random test cases from different students.

TABLE II: NER Classes

Entity	Precision	Recall	F-mesure
INSTRUCTOR	94.6%	86.1%	90.1%
TA	96.6%	92.1%	94.3%
COURSE	94.5%	81.9%	87.7%
EVAL	91.9%	87.4%	89.6%

Machine Learning NER: In addition to Rule-Based method, we trained a machine learning NER model using some of Scikit-Learn's libraries[3]. The data we used was annotated with POS tags to help the NER model to classify the words correctly. We trained 6834 words with the following NEs: INSTRUCTOR (460 words), COURSE (708) words, TA (129 words) and EVAL (120 words), and the rest was

[2]https://iq.opengenus.org/ autocomplete-using-trie-data-structure/

[3]https://scikit-learn.org/stable/

classified as O (others) and it has 5417 words. We used the following classifiers: Perceptron, SGD, Naive Bayes and Passive Agressive Classifier. 25% of the dataset is used for testing. See table III which shows the results of our machine learning model.

TABLE III: Machine Learning NER Results

Classifier	Entity Tag	Precision	Recall	F-measure
Perceptron	INSTRUCTOR	98.0%	95.0%	97.0%
	TA	91.0%	89.0%	90.0%
	COURSE	93.0%	92.0%	93.0%
	EVAL	94.0%	87.0%	91.0%
	O	98.0%	99.0%	99.0%
SGD	INSTRUCTOR	98.0%	94.0%	96.0%
	TA	95.0%	94.0%	95.0%
	COURSE	98.0%	91.0%	94.0%
	EVAL	97.0%	79.0%	87.0%
	O	98.0%	97.0%	98.0%
Passive Agressive	INSTRUCTOR	98.0%	95.0%	97.0%
	TA	91.0%	82.0%	86.0%
	COURSE	98.0%	89.0%	93.0%
	EVAL	94.0%	85.0%	89.0%
	O	97.0%	96.0%	97.0%
Naive Bayes	INSTRUCTOR	75.0%	93.0%	83.0%
	TA	82.0%	90.0%	86.0%
	COURSE	79.0%	91.0%	84.0%
	EVAL	73.0%	92.0%	82.0%
	O	98.0%	94.0%	96.0%

3) Next-Word Prediction: We used LSTM for next word prediction. We built a sequential model for next-word prediction. An embedded layer was created to represent individual words. We added another LSTM layer to the model, and we gave it 100 units with all the units being specified true so as to pass through another LSTM layer, another LSTM layer was added to this model but with a specified value false by default. After that, we started passing the model through hidden layer with 100 node units using dense layer and ReLU as the activation function. In order to ensure that we acquire a range of probabilities for a single vocabulary size, the last layer was then sent through an output layer with a specific vocabulary size and Softmax activation function. See figure 10 which shows our model. We trained it on our dataset with

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 3, 10)             7800

 lstm (LSTM)                 (None, 3, 100)            44400

 lstm_1 (LSTM)               (None, 100)               80400

 dense (Dense)               (None, 100)               10100

 dense_1 (Dense)             (None, 780)               78780

=================================================================
Total params: 221,480
Trainable params: 221,480
Non-trainable params: 0
```

Fig. 10: LSTM Model

350 epochs which indicates the number of iterations of the entire training dataset that the learning algorithm completed, and with batch size equals to 32, where batch size refers to the number of training examples utilized in one iteration. We got an accuracy of 84.51% and loss value of 0.27, where the loss value refers to how bad the model's prediction was on a single example (value between 0 and 1).

D. Classification

The goal of this step is to map the input (variant query which is resulted from word-level and next word prediction models) to its class (standard query). After pre-processing, we tested classification using different machine learning and deep learning models as follows:

1) Data Preprocessing: Data preprocessing may be an important process to achieve better results in ML. Here, it consists of the following steps:

1) Removing words duplicates.
2) Removing stop words
   من، الى، عن، على، .. الخ. We used NLTK python library that contains all Arabic stop words [4].
3) Normalization: removing confusing letters
   آ أ إ، ه ة، ى ئ ي، ؤ و، !؟
4) Removing non-Arabic characters.
5) Removing Diacritics
   مُعلّم : معلم.
6) Removing tide in letters (duplicate letters):
   وو : و
   يي : ي
   اا : ا

2) Deep Learning Classification (Multiclass-Keras-Neural Networks): Keras is a powerful, free, open source Python library for developing and evaluating deep learning models. We used both ReLU (Rectified Linear Unit) and Sigmoid as activation functions represented by equations 2 and 3, respectively.

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{1}$$

$$R(z) = max(0, z) \tag{2}$$

Figure 11 shows the plot for those two functions.

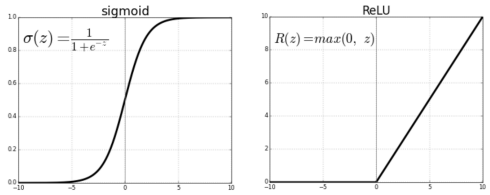

Fig. 11: Sigmoid VS ReLU

The problem with ReLU is that any negative input given to the activation function turns the value into zero, which affects the results by not mapping the negative values appropriately.

[4]https://github.com/mohataher/arabic-stop-words/blob/master/list.txt

we used the multi-class neural networks model for our classification and the **precision** was about 88%, **recall** was 87% and **F-measure** was 87.49% . Figure 12 shows sample output of the classifications.

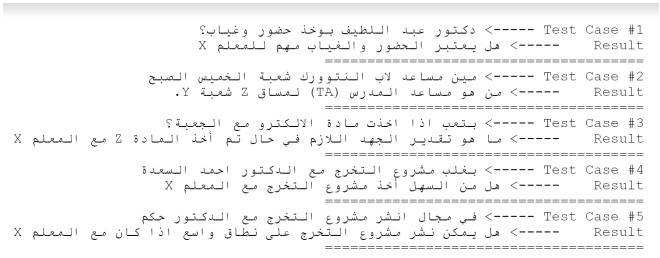

Fig. 12: Deep Learning Classification Output

3) Machine Learning Classification: We experimented with different classifiers using Weka tool on our dataset. We vectorized our queries using StringToWord Weka filter, then we tested these vectors using several supervised machine learning approaches and 10-Fold cross validation to test the models[5]. Table IV shows the classification results.

TABLE IV: Machine Learning Classification Results of Variants into Standard Queries

Classifier	Precision	Recall	F-measure
Decision Tree	71.0%	70.6%	70.8%
Random Forest	81.9%	81.8%	81.1%
SVM	86.0%	84.7%	84.5%
NaiveBayes	87.4%	87.0%	86.8%

The deep learning and ML models are comparable, but we adopted the deep learning model as it achieved slightly better results. The mapping from variant to standard is very important process for our QA system, it's easy for us to translate the standard queries rather than the the variants.

4) Time-Sensitivity and Personalization: Each standard query has two parameters:

The studying year in which this question is usually asked: This point is related to the subjects taught in different years/semesters, each course in our data has the study year as a parameter, and the names of the teachers who teach this course. The ranking of the suggested list may change depending on these parameters: for example if the student is looking for an elective, courses beyond his current level will be given lower priority.

In what period is the question usually asked: There are questions related to exams, and there are questions related to registration matters.

E. Questions Classification

We classified the standard queries manually as: exam-related, registration-related, graduation projects-related, financial matters as shown in Table V. This classification will help us in the suggestions ranking stage, by changing the rank according to the current period of the semester and the student's studying level.

[5]https://machinelearningmastery.com/k-fold-cross-validation/

TABLE V: Standard Questions Classification

Class	Question
Exams	نماذج سابقة لامتحانات الذكاء الاصطناعي
Registration	متى يبدأ التسجيل للفصل الأول
Graduation	متى ممكن أنزل مشروع التخرج
Financial	متى يبدأ التقسيط للفصل الأول

F. QAC Suggestions Ranking

The suggested queries should be ranked in a satisfying way for the student who asks the question, with the shortest entered prefix. The user is supposed to see the desired completion that reflects the student's need. For example, if we are in the exam period, the student probably will ask about exam dates, previous exam forms, and so on. If the student is in his first year of studying, he will ask about first-year subjects. Sometimes, students share the same questions, for example, at the beginning of the semester, all students ask about subject registration and lecture dates. So the frequency of a question (number of times the question is being asked) during a specific period of time (e.g. the first month of the semester) is good to for ranking questions. There are several criteria to rank our suggested completions as shown in the following equation:

$$RScore(c) = \alpha\frac{f(c)}{F} + \beta\frac{\sum_{q_s \in S_p} cosSim(c, q_s)}{N_p} + \gamma\frac{N_E}{N_C}$$

Suppose we are in the exams period. $RScore(c)$ represents the ranking score for completion c, $f(c)$: the general frequency of completion c over a period of time, F: the total number of occurrences for all standard queries within the considered time period, q_s is a standard query from the set of standard queries S_p that are related to exams period, N_p is the number of standard queries that are related to the exam period. N_E is the number of entities related to the studying level of the student in the completion c, and N_C is the total number of words in the completion c. We selected the values for α, β, γ to be 0.2, 0.4, and 0.4 respectively because the student prefers to see suggestions that match his need, we increased the values of β and γ compared to α.

G. QAC Evaluation

A good QAC system is one that returns the user's need with the shortest entered prefix p. The following specifies the parameters that are taken into account for our QAC system evaluation.

$$Score(p) = \alpha CTR(q_c) + \beta\frac{1}{Length(p)}, CTR(q_c) = \frac{NC}{I}$$

where p is the input prefix from the user and $Length(p)$ is the length of p; q_c is the suggested completion and $CTR(q_c)$ is its click-through rate defined through NC: the number of clicks, and I: the impression which represents the number of times the completion q_c is shown to the user, every time the query is suggested, its impression value will be increased. For example, if the completion $\hat{q_c}$ was shown as a suggestion

100 times and was chosen (clicked) by a student 10 times, then $CTR(\hat{q_c}) = 0.1$. In our initial experiments, we set α to 0.6 and β to 0.4. We are working now on building the system and testing it and we can add any valuable parameter for evaluation. To adopt this evaluation for the entire system, we can store the score for every prefix p entered from different students during a specified period of time, then by calculating the score average, we can evaluate the entire system.

IV. CONCLUSION AND FUTURE WORK

In this paper, we studied the problem of Arabic QAC for QA and described all its aspects. In addition, we studied the effect of QAC on QA and how it can improve the efficiency of the QA by reducing the number of queries and improving their quality, an important issue for many applications for interaction with users. Our system dealt with university students academic questions. However, we believe that with the proper datasets and NLP tools, the approach can be easily extended to other fields/applications. We will improve our system to be used for other tasks, say in the legal of finance fields. Also, we will try to expand our dataset and make it available to researchers interested in Arabic QAC.

REFERENCES

[1] Y. Li, J. Amelot, X. Zhou, S. Bengio, and S. Si, "Auto completion of user interface layout design using transformer-based tree decoders," 2020.

[2] S. Abiteboul, Y. Amsterdamer, T. Milo, and P. Senellart, "Auto-completion learning for xml," 05 2012.

[3] K. Arkoudas and M. Yahya, "Auto-completion for question answering systems at bloomberg," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018* (K. Collins-Thompson, Q. Mei, B. D. Davison, Y. Liu, and E. Yilmaz, eds.), pp. 1351–1352, ACM, 2018.

[4] F. Cai and M. de Rijke, "A survey of query auto completion in information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 10, no. 4, pp. 273–363, 2016.

[5] L. Li, H. Deng, A. Dong, Y. Chang, H. Zha, and R. Baeza-Yates, "Analyzing user's sequential behavior in query auto-completion via markov processes," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, (New York, NY, USA), p. 123–132, Association for Computing Machinery, 2015.

[6] S. Whiting and J. Jose, "Recent and robust query auto-completion," pp. 971–982, 04 2014.

[7] Z. Bar-Yossef and N. Kraus, "Context-sensitive query auto-completion," in *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, (New York, NY, USA), p. 107–116, Association for Computing Machinery, 2011.

[8] M. Jarrar, "The arabic ontology - an arabic wordnet with ontologically clean content," *Applied Ontology Journal*, vol. 16, no. 1, pp. 1–26, 2021.

[9] N. Loukil, K. Haddar, and A. Ben Hamadou, "A syntactic lexicon for arabic verbs.," 01 2010.

[10] M. G. Al Zamil and Q. Al-Radaideh, "Automatic extraction of ontological relations from arabic text," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 26, p. 462–472, dec 2014.

[11] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[12] A. Sethy, S. Chen, E. Arisoy, and B. Ramabhadran, "Unnormalized exponential and neural network language models," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5416–5420, 2015.

[13] A. R. A. Qureshi and M. A. Akcayol, "Long short-term memory based query auto-completion," in *2021 8th International Conference on Electrical and Electronics Engineering (ICEEE)*, pp. 259–266, 2021.

[14] L. Qiu, H. Zhou, Y. Qu, W. Zhang, S. Li, S. Rong, D. Ru, L. Qian, K. Tu, and Y. Yu, "QA4IE: A question answering based framework for information extraction," *CoRR*, vol. abs/1804.03396, 2018.

[15] O. Obeid, N. Zalmout, S. Khalifa, D. Taji, M. Oudah, B. Alhafni, G. Inoue, F. Eryani, A. Erdmann, and N. Habash, "CAMeL tools: An open source python toolkit for Arabic natural language processing," in *Proceedings of the 12th Language Resources and Evaluation Conference*, (Marseille, France), pp. 7022–7032, European Language Resources Association, May 2020.