Indonesian Question Answering System for Factoid Questions using Face Beauty Products Knowledge Graph

Mahanti Indah Rahajeng¹, Ayu Purwarianti¹²

¹School of Electrical Engineering and Informatics, Bandung Institute of Technology
²U-CoE AI-VLB
Bandung, Indonesia
¹¹3517085@std.stei.itb.ac.id
²ayu@staff.stei.itb.ac.id

Abstract— Question answering (QA) system is developed to find the right answers from natural language questions. QA systems can be used for building chatbots or even search engines. In this study, we’ve built an Indonesian QA system that uses Anindya Knowledge Graph as its data source. The idea behind this QA system is translating questions into SPARQL queries. The proposed solution consists of four modules, namely question classification, information extraction, token mapping, and query construction. The question classification and the information extraction modules were experimented using SVM, LSTM, and fine-tuning IndoBERT. The text representations were also tested to find the best result among tf-idf, FastText, and IndoBERT. In our experiment, we found that the fine-tuning IndoBERT model had obtained the best performance on both question classification and information extraction modules.

Keywords— question answering; knowledge graph; SVM; LSTM; IndoBERT

I. INTRODUCTION

Question answering (QA) is a research field in natural language processing, developed for finding the right answers from given natural language questions. QA systems can be used for building chatbots or even search engines. QA systems can be built over various data sources, such as textual document and knowledge graph. Knowledge graphs construction using Indonesian language has been done in several studies, for example Anindya Knowledge Graph [1]. However, there is no QA system yet that uses the Indonesian knowledge graph. Therefore, this study will be focused on building Indonesian QA system that uses Anindya Knowledge Graph for its data source.

The idea behind QA systems over knowledge graph is translating questions into SPARQL query. Common processes in QA systems are question analysis, phrase mapping, disambiguation, and query construction [2]. The question analysis process aims to extract important information from question. The phrase mapping process aims to map extracted token to possible related resource in the knowledge graph by using lexical or semantic similarity. The disambiguation process aims to find the best pair of extracted token and knowledge graph resource. Lastly, the query construction process aims to construct SPARQL query based on the information from the previous processes. The example of translated question is as follows:

Berapa ukuran dari Emina Creammate?

(What is the size of Emina Creammate?)

SELECT DISTINCT ?property
WHERE {
  akg:measurement ?property .
}

In this study, the QA system is built using four modules, namely question classification, information extraction, token mapping, and query construction. The question classification module and the information extraction module will carry out the question analysis process. The token mapping module will carry out the phrase mapping and disambiguation.

Question classification and information extraction are common tasks in natural language processing, namely text classification and sequence tagging. Both tasks can be built using shallow learning and deep learning. Hence, for the question classification module and the information extraction module, we compared three models: SVM [3][4], LSTM [5][6], fine-tuning IndoBERT [7] and three text representations: tf-idf, FastText [8], IndoBERT [7] to find the best model for building those modules. We chose FastText as non-contextual embedding due to its ability to get the representation of OOV word by summing its subword representation. Other than that, we used
IndoBERT as contextual embedding because it is a state-of-the-art language model for Indonesian based on the BERT model. We also used IndoBERT with two adaptation methods of transfer learning, which are fine-tuning and feature extraction.

To sum up, in this study we have built an Indonesian QA system from Anindya Knowledge Graph and conducted experiments in the question classification module and the information extraction module to find the best model and text representation to use. We will discuss the methodology used and the experiment results in the later section.

II. METHODOLOGY

There are some processes conducted in this study, starting from knowledge graph analysis, dataset building, system design, and modules implementation.

A. Knowledge Graph Analysis

Anindya Knowledge Graph [1] is a domain specific knowledge graph that contains face beauty products data from three brands: Emina, Sariayu, and Wardah. This knowledge graph has two prefixes: akgr denotes resources and akgs denotes semantics. Resources are product and organization entities, while semantics are properties, relations, and classes. Property information includes name, brand, texture, measurement, variant, and usage. Relation information includes property relation, product relation, and class relation. Classes consist of product class and organization class. An illustrative example of Anindya Knowledge Graph can be seen in Figure 1.

![Figure 1 Illustration of Anindya Knowledge Graph](Image)

Because this study is limited to factoid questions, the possible answer types of the given questions will only relate to organization entities, product entities, and product properties.

After exploring this knowledge graph, we found that the classes and relations in Anindya Knowledge Graph are written in English because they follow the schema from Schema.org. The product size property also has no writing standards or unit of measure rules for products. For example, the gram unit can be written as g and gr. We used simple approaches such as lexical similarity to overcome this problem.

B. Dataset Building

The question classification and the information extraction modules need labelled data for training and testing purposes. There are three labels or classes of answer type for the question classification module: Organization, Product, and Property. For the information extraction module, the token type labels used are Product, Organization, Relation, Property, and Class, which use Begin-Inside-Other (BIO) notation.

We have collected data for 503 questions. We made about 80% of the total questions and the rest were collected from several people. We also performed data labelling and ground truth queries building.

C. System Design

The proposed solution of the QA system consists of four modules, namely question classification, information extraction, token mapping, and query construction. The question classification module is needed to determine the ‘SELECT’ clause, while the information extraction module determines the ‘WHERE’ clause. The token mapping module is needed to map extracted tokens into possible related knowledge graph resources and determine the best pair of it. The query construction module is needed to construct a SPARQL query that matches the given questions. The general system architecture can be seen in Figure 2.

![Figure 2 General System Architecture](Image)

D. Modules Implementation

The implementation of each module is explained as follows:

1) Question Classification: This module receives question text and determines the answer type class that matches the given question. We compared seven models for this module: SVM tf-idf, SVM FastText, SVM IndoBERT, LSTM FastText, LSTM IndoBERT, fine-tuning IndoBERT, and fine-tuning IndoBERT with auxiliary sentence [9]. For the LSTM model, we employed two types of model: single layer BiLSTM and single layer BiLSTM along with a global pooling mechanism [5]. We also employed early stopping callback while training the LSTM models. For the fine-tuning IndoBERT model, we used single sentence classification task and sentence pair classification task for the one with
auxiliary sentence. We used the addition of auxiliary sentence in the fine-tuning IndoBERT model because it can increase the amount of training data.

2) Information Extraction: This module receives question text and results a set of extracted tokens from the question along with the token type label that can be mapped to resources from Anindya Knowledge Graph. We compared five models for building this module: SVM FastText, SVM IndoBERT, LSTM FastText, LSTM IndoBERT, and fine-tuning IndoBERT. For the LSTM model, we employed two types of model: single layer BiLSTM and single layer BiLSTM with CRF [6]. We also employed early stopping callback while training the LSTM models. For the fine-tuning IndoBERT model, we used a single sentence tagging task. For the SVM model, we used some features, such as the current token, the previous token, the label of the previous token, POS tag of the current token, and POS tag of the previous token.

3) Token Mapping: This module receives a set of extracted tokens along with the token type label and lexicalization dictionary from the Anindya Knowledge Graph resources. We built the lexicalization dictionary using the translation results and synonyms found in the training data. This dictionary also stores resource types. Resource types used in the dictionary are brand, class, company, measurement, name, product, relation, texture, type, usage, and variant. Each extracted token is calculated for its lexical similarity to all resources of the same type. Lexical similarity is calculated using Jaccard and Levenshtein. The token and resource pair with the highest similarity value is taken as input for the next module.

4) Query Construction: This module receives the token mapping result and the answer type class. This module uses simple query templates based on the Anindya Knowledge Graph schema. The list of ‘WHERE’ clause from query templates used is written as follows:

- [Organization entity] a akgs:Organization
- [Product entity] a akgs:Product
- [Product entity] [Property relation] [Property value]
- [Organization entity] akgs:produces [Product entity]

Once the query is constructed, it needs to be run to get the corresponding answer of the given question.

III. EXPERIMENT AND RESULT

Experiments were performed for the question classification and the information extraction modules. The experiments aim to get the best configuration of each model used. Each model with the best configuration was then compared to the other models. Cross validation method was used to find the best configuration. The question data was divided into 90% training data and 10% test data. After the best configuration was found, the model was retrained using the complete training data and evaluated using the test data. The best model was used to build the QA system.

A. Question Classification

The best hyperparameter configuration of each model can be seen in Table I, while the evaluation results can be seen in Table II.

**Table I**
The Best Configuration of Question Classification Module

<table>
<thead>
<tr>
<th>Model</th>
<th>Avg. Accuracy</th>
<th>Parameter Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM tf-idf</td>
<td>0.8518357</td>
<td>Kernel: sigmoid C: 100 Gamma: 0.1</td>
</tr>
<tr>
<td>SVM FastText</td>
<td>0.8717391</td>
<td>Kernel: rbf C: 100 Gamma: 0.01</td>
</tr>
<tr>
<td>SVM IndoBERT</td>
<td>0.9579710</td>
<td>Kernel: poly C: 0.1 Gamma: 1</td>
</tr>
<tr>
<td>LSTM FastText</td>
<td>0.8870531</td>
<td>Dropout rate: 0.1 Batch size: 8 Optimizer: Adam Model type: Single layer BiLSTM Epoch: 18</td>
</tr>
<tr>
<td>LSTM IndoBERT</td>
<td>0.9713044</td>
<td>Dropout rate: 0.1 Batch size: 32 Optimizer: Adam Model type: Single layer BiLSTM with global pooling mechanism Epoch: 15</td>
</tr>
<tr>
<td>Fine-tuning IndoBERT</td>
<td>0.9911859</td>
<td>Epoch: 3 Batch size: 16 Learning rate: 5e-5</td>
</tr>
<tr>
<td>Fine-tuning IndoBERT auxiliary</td>
<td>0.9867150</td>
<td>Epoch: 4 Batch size: 32 Learning rate: 3e-5</td>
</tr>
</tbody>
</table>

**Table II**
The Evaluation Result of Question Classification Module

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM tf-idf</td>
<td>0.8431372</td>
</tr>
<tr>
<td>SVM FastText</td>
<td>0.8823529</td>
</tr>
<tr>
<td>SVM IndoBERT</td>
<td>0.9803922</td>
</tr>
<tr>
<td>LSTM FastText</td>
<td>0.9215686</td>
</tr>
<tr>
<td>LSTM IndoBERT</td>
<td>0.9803922</td>
</tr>
<tr>
<td>Fine-tuning IndoBERT</td>
<td>1.0000000</td>
</tr>
<tr>
<td>Fine-tuning IndoBERT auxiliary</td>
<td>0.9803922</td>
</tr>
</tbody>
</table>

The majority of prediction errors occurred in questions with Organization answer type. This answer type has less data than the other types. The example of predictions error is as follows:
Wardah Perfect Curl Mascara itu produk dari mana? (Where does Wardah Perfect Curl Mascara come from?)

Almost all models misclassified this question. This may be due to the irregular or unusual sentence structure. However, the fine-tuning IndoBERT model obtained the best evaluation result on the test data. Therefore, we used this model to build the question classification module.

The experiment results showed very high accuracy for all models. This may be due to the small amount of data, also the data used was very specific to the knowledge graph, and the lack of structural and vocabulary variations in the question data. In general, LSTM models performed better than SVM models and fine-tuning IndoBERT models had a higher accuracy than LSTM but the two were quite the same. This showed that more complex models provide better performance. Complex means being able to learn the context of word sequences or the context of the whole sentences. Other than that, the use of word embedding has better performance compared to the tf-idf representation. Contextual word embedding (IndoBERT) also provides better performance compared to the non-contextual one (FastText).

B. Information Extraction

The best hyperparameter configuration of each model can be seen in Table III, while the evaluation results can be seen in Table IV.

<table>
<thead>
<tr>
<th>Model</th>
<th>Avg. F1-score</th>
<th>Parameter Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM FastText</td>
<td>0.8330707</td>
<td>Kernel: rbf C: 100 Gamma: 0.01</td>
</tr>
<tr>
<td>SVM IndoBERT</td>
<td>0.8428162</td>
<td>Kernel: linear C: 0.1 Gamma: 1</td>
</tr>
<tr>
<td>LSTM FastText</td>
<td>0.9519655</td>
<td>Dropout rate: 0.1 Batch size: 1 Optimizer: Adam Model type: Single layer BiLSTM Epoch: 14</td>
</tr>
<tr>
<td>LSTM IndoBERT</td>
<td>0.9530432</td>
<td>Dropout rate: 0.1 Batch size: 8 Optimizer: Adam Model type: Single layer BiLSTM with CRF Epoch: 11</td>
</tr>
<tr>
<td>Fine-tuning IndoBERT</td>
<td>0.9581007</td>
<td>Epoch: 3 Batch size: 8 Learning rate: 5e-5</td>
</tr>
</tbody>
</table>

TABLE IV

The Evaluation Result of Information Extraction Module

<table>
<thead>
<tr>
<th>Model</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM FastText</td>
<td>0.81</td>
</tr>
<tr>
<td>SVM IndoBERT</td>
<td>0.95</td>
</tr>
<tr>
<td>LSTM FastText</td>
<td>0.95</td>
</tr>
<tr>
<td>LSTM IndoBERT</td>
<td>0.98</td>
</tr>
<tr>
<td>Fine-tuning IndoBERT</td>
<td>0.98</td>
</tr>
</tbody>
</table>

The majority of prediction errors occurred in questions with product entities or product properties. The example of misclassified question is as follows:

Berapa ml ukuran produk eye and lip makeup remover wardah? (How many milliliters of wardah eye and lip makeup remover ?)

The fine-tuning IndoBERT model predicted the question as follows:

Berapa ml ukuran produk eye and lip makeup remover wardah ?
B-Produk I-Produk I-Produk I-Produk O

While the ground truth label was as follows:

Berapa ml ukuran produk eye and lip makeup remover wardah ?
I-Produk I-Produk I-Produk I-Produk O

However, the LSTM IndoBERT model and the fine-tuning IndoBERT model obtained the highest evaluation result on the test data. Because the fine-tuning IndoBERT model also obtained the best average result for the validation data, we used this model to build the information extraction module.

Same as the question classification module, LSTM models performed better than SVM models and fine-tuning IndoBERT model was better than LSTM but the two were quite the same. Other than that, the use of contextual embedding (IndoBERT) provided better performance compared to the non-contextual one (FastText).

C. Query Construction

The evaluation result of this module can be seen in Table V. Query generation tasks can be evaluated using two accuracy metrics, namely execution accuracy and logical form accuracy [10]. The logical form accuracy had a very low score because it only performed exact matching on the ground truth queries. Many predicted queries have different forms from the ground truth queries, but they have the same query execution results. This can
be seen from the execution accuracy score, which was much higher than the logical form accuracy.

### Table V

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution accuracy</td>
<td>0.8235294</td>
</tr>
<tr>
<td>Logical form accuracy</td>
<td>0.0392157</td>
</tr>
</tbody>
</table>

Prediction errors were caused by several things, such as the question and the knowledge graph use different language, token cannot be mapped to the correct product if only part of the product name is given, ambiguity in property labels, limited query templates, etc. The example of incorrect query along with the given question are as follows:

Berapa ukuran Sariayu Color Trend Mascara? (What is the size of Sariayu Color Trend Mascara?)

SELECT DISTINCT ?property
WHERE
{ akgr:SariayuCreamyFoundation a akgs:Product .
   akgr:SariayuCreamyFoundation akgs:measurement ?property .
}

This query is incorrect because the product tokens could not be mapped to the correct product resource. Instead of `akgr:SariayuColorTrend2015Mascara`, the product tokens are mapped to `akgr:SariayuCreamyFoundation`. This is because the Jaccard score for the creamy foundation is the highest.

### D. QA System

The question classification and the information extraction modules were built using the fine-tuning IndoBERT model. The evaluation result of the QA system can be seen in Table VI.

### Table VI

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro precision</td>
<td>0.8823529</td>
</tr>
<tr>
<td>Macro recall</td>
<td>0.8418301</td>
</tr>
<tr>
<td>Macro F1-score</td>
<td>0.8499703</td>
</tr>
</tbody>
</table>

The QA system was evaluated based on the answers obtained from the query execution results. System performance was highly dependent on the performance of each module used. The question classification and the information extraction modules already have good performance. However, the query construction module still makes prediction errors in some cases.

### IV. Conclusion

The conclusion that can be drawn from this study is that the fine-tuning IndoBERT model has the best performance result for classifying a question based on its answer type in the question classification module and tagging the question text in the information extraction module. Other than that, the pipelined system architecture makes the performance of a module highly dependent on the previous modules.

For further improvement, more question data can be collected from more people to get various question patterns, both structurally and vocabulary, so that the model can be evaluated more fairly. Moreover, the resource lexicalization dictionary can be built not manually. Manual dictionary construction requires knowledge of the resources and their lexicalization itself. Moreover, it relies heavily on the training data, which causes a limited lexicalization dictionary. Lastly, token mapping can use semantic similarity. One of the drawbacks of the current lexical similarity approach is that it cannot map resources if only part of token or phrase is given. Semantic similarity is expected to handle this case.

### References


