Received: June 1, 2021. Revised: July 21, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Out-of-Scope Intent Detection on A Knowledge-Based Chatbot

Lindung Parningotan Manik[1]* **Zaenal Akbar[1]** **Hani Febri Mustika[1]** **Ariani Indrawati[1]**
Dwi Setyo Rini[2] **Agusdin Dharma Fefirenta[2]** **Tutie Djarwaningsih[2]**

[1]*Research Center for Informatics, Indonesian Institute of Sciences, Indonesia*
[2]*Research Center for Biology, Indonesian Institute of Sciences, Indonesia*
* Corresponding author's Email: lmanik@mail.informatika.lipi.go.id

Abstract: Knowledge-based chatbot (KBC) has grown in popularity in recent years and has been widely used for various use cases. Building KBC from scratch using deep learning (DL) is challenging since no prior historical data exists. Meanwhile, DL systems need a vast Volume of data to be trained. This paper proposes a novel framework to create an intent classifier of the KBC used to detect in-scope (IS) and out-of-scope (OOS) intents. We introduce an automated queries generator to create IS intents employed as the training data from an ontology input. We utilize Bidirectional Encode Representations from Transformers (BERT) fine-tuning as the backbone of our DL system. Moreover, we present a Bayesian approach as an extension of the BERT to classify OOS queries with minimal OOS training data. The experiments result show that the proposed method manages to achieve an F1 score of 100% for IS intents and 86% for OOS queries.

Keywords: Knowledge base, Chatbot, Out-of-scope, Intent classification, BERT.

1. Introduction

A knowledge-based chatbot (KBC) is an intelligent assistant that utilizes natural language processing (NLP) and embedded knowledge to drive its chats [1]. Nowadays, KBCs are designed for various purposes, such as in health care service [2], the insurance industry [3], or even to fight Covid-19 [4]. A KBC has the broad goal of assisting users in finding information faster.

Knowledge base (KB) can be represented in various forms such as database, description logic, or ontology. OntBot, for example, employed an effective mapping method to convert ontologies and knowledge into relational databases, which it then uses to power its chats [5]. IntelliBot even used the internet as one of the sources of knowledge [3]. On the other hand, KBot applied knowledge graphs or linked data to aggregate multiple KBs [1].

Natural language understanding (NLU) plays a central role in a generic KBC, as shown in Fig. 1. While the intent classifier is an element that performs a classification of an input query with a categorical intent [6], the named-entity extractor is a component that performs a multi-classification job to recognize various predefined classes and extract their entities from an input utterance [7]. The chat engine uses the intent to take proper actions. Meanwhile, the engine formulizes and generates an appropriate response after querying the information related to the entities from the KB.

Since the input utterances from the user are arbitrary, not all queries can be answered by the system because the KB is limited. Queries that do not fall into any of the systems' supported intents are defined as out-of-scope (OOS) queries [8]. On the contrary, in-scope (IS) queries or intents are expected to be answered.

The intent classifier should be able to predict a class (IS or OOS) for each query. Therefore, the IS/OOS classification task is essential to classify natural language queries' intent correctly. Several examples of queries are shown in Table 1 to demonstrate the prediction job.

When creating the chatbot from scratch, a sufficient amount of training data is not available.

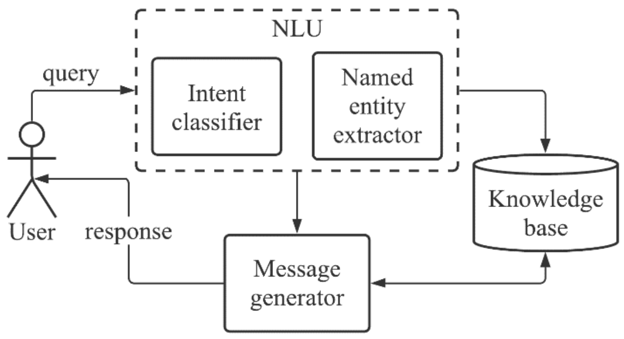


Figure. 1 A generic KBC components

Table 1. Example of IS and OOS queries

Query	Class
What kind of Capsicum has a star-like flower shape and a few millimeters length of pistil?	IS
Show me all objects that have triangular-like leaves shape!	IS
What is the market price of Capsicum?	OOS
Why is Capsicum expensive?	OOS

The system only has the KB without any historical data. Without the data, training the intent classifier that utilizes machine learning (ML) algorithms is challenging. Moreover, it is well-known that deep learning (DL) systems need a vast Volume of data to be trained [9]. In addition, determining a user's unknown intents is problematic since there is no prior information [10]. On the other hand, estimating the precise number of OOS intents is also tricky [11], if not impossible.

In this paper, we show a novel framework to tackle the problems and present two contributions. First, we introduce a template-based natural language generation (NLG) to generate IS queries as training data from an ontology input. Furthermore, we propose a Bayesian version of the Bidirectional Encode Representations from Transformers (BERT) to classify IS and OOS queries with minimal training data.

The rest of the paper is organized as follows. In section 2, we present approaches in classifying IS intents in a text. We also identify related works of detecting OOS. In other papers, OOS is also referred to as OOD, out-of-distributions, or out of domains. Furthermore, we demonstrate the proposed framework and the research method in section 3. The experiment and evaluation results are presented in section 4. Finally, the conclusion and future works are given in section 5.

2. Related works

In general, there are two approaches to detecting intent in a text. The first one uses a rule-based approach, and the second one utilizes ML algorithms. The algorithms are very accurate with a large number of training data. Meanwhile, rule-based approaches are accurate with a small or large number of data. Google Dialogflow tries both of these algorithms and selects the better of the two results [12].

Due to limited accessible datasets, the issue of OOS intent detection is not actively investigated. In [8], methods for detecting OOS samples depended on one-class classification or threshold rejection approaches based on the probability outputs for each class. A neural sentence embedding approach representing sentences in a low-dimensional vector space while emphasizing IS characteristics that differentiate from OOS cases was employed in [13]. The OOS classification could also be accomplished by mapping text embeddings of IS data to the groups' word graph space [14]. Furthermore, Kullback-Leibler (KL) divergence is applied in [15] to gather information between the intent distributions of consecutive words.

OOS classification methods are categorized into two approaches, few-shot learning (FSL) and zero-shot learning (ZSL). While FSL tries to create machine learning models using a small amount of OOS training data, such as in this work, Google DialogFlow, and [15], ZSL relies solely on IS training data to detect an OOS like in [13, 14]. However, both FSL and ZSL still require labeled IS datasets. All previous works did not address the problems of acquiring the IS datasets.

An FSL approach is commonly selected when the case is a low-resource language. However, with recent advances in NLP, a DL-based FSL can also be utilized with a small dataset. Transfer learning is applied to deal with this type of scenario by transferring information learned from addressing one problem used to solve another but related problem. In the past, the NLP was behind computer vision because it lacked fine-tuning approach, a standard method for transfer learning. Nevertheless, the situation has changed since [16] introduced the Transformer-based model. It is a DL model that employs the attention mechanism, weighing the effect of various parts of the input data.

The work in [13, 15] pre-trained the Long-Short-Term Memory (LSTM) network from scratch for neural sentence embedding. GloVe, Word2Vec, and FastText are popular methods in this context to create word embeddings [9]. However, this approach requires vast resources and high costs. BERT, a Transformer-based learning technique developed for NLP pre-training, introduced in [17], came as a rescue. The Transformer is used as an attention mechanism that learns contextual relations between words in a text. It was unsupervised trained using a plain text corpus, like Wikipedia or any text data accessible publicly on the internet, in any language. As a result, BERT performs better than conventional ML algorithms in any NLP classification task [18].

In most cases, a threshold on the classifier's probability estimate is used to predict an OOS. The OOS threshold was chosen as the value that produced the highest validation score across all intents, with OOS treated as its own intent [8]. In Google DialogFlow, for example, an intent detection confidence score, ranging from 0.0 (very uncertain) to 1.0 (completely confident), is generated while looking for a matching IS intent. The highest scoring IS intent is returned to match if its confidence score is greater than or equal to the threshold setting. Otherwise, if no IS intents fit the criteria, an OOS is matched.

Nevertheless, one significant problem rarely addressed for these models is estimation uncertainty. Most ML or DL models make predictions without considering uncertainty [19]. Since its introduction, Bayesian DL has been used widely in computer vision tasks [20]. It combines DL with Bayesian probability theory and offers uncertainty estimation from DL architectures. However, its study on NLU tasks is limited, especially in detecting OOS queries from a KB represented by an ontology.

Epistemic and aleatoric are two types of uncertainties handled very well by Bayesian DL [19]. Epistemic uncertainty comes from the lack of information, while aleatoric captures uncertainty concerning information that data cannot explain [20]. In detecting OOS intent, epistemic uncertainty happens more often than aleatoric due to given small datasets. The uncertainty can be explained away given enough data.

A natural language generator (NLG) can produce a small dataset from a KB required by the DL's fine-tuning. A few works have been performed to generate queries from a KB in a knowledge-based system (KBS). For example, in response to the user's natural language inquiries to the KBS, [21] offered queries that the system can answer. Eight question types were identified: event-centered, identification, find a value, how many, comparison, relationship, definitional, and yes/no. The generated questions were used to serve as the suggestions facility. In another work, [22] made multiple-choice questions from an ontology using description logic. Moreover, [23] developed questions from linked data. We adopt these approaches to provide natural queries to a KB that scopes user expectations about what a chatbot can answer.

In line with the previous related works, we contribute to two folds. The first is an algorithm to generate an IS training dataset out of a knowledge base to tackle problems in acquiring IS datasets when building chatbots from scratch. The second is an FSL approach in detecting OOS queries that utilize BERT to tackle low-resource language problems and Bayesian uncertainty estimation to be used as a threshold method.

3. Research method

This section represents the research method, as shown in Fig. 2. Three main building blocks in the pipeline, colored yellow, are described in the following subsections. The building blocks take three kinds of input data, colored green, intent query template, ontology, and OOS data.

3.1 Automatic IS queries generator

In this subsection, we describe the building block used to generate IS queries automatically from a KB. An intent query template and an ontology are used as the inputs. An ontology in the web ontology language (OWL) is an abstract representation of knowledge entities (object, concepts, and relations) in a domain of interest.

In this proposed approach, the KB is referred to as a description logic KB, equivalent to a theory in first-order logic (FOL) or an ontology in OWL. It is represented by a pair of *TBox* and *ABox* [24]. A *TBox* is a set of terminological axioms of concept inclusions of the form $C \sqsubseteq D$ and concept equivalences of the form $C \equiv D$, where C and D are concepts. Meanwhile, an *ABox* is a set of assertional axioms of form $C(a)$, $R(a, b)$ where a and b are individual names, C is a concept, and R is a role. A concept corresponds to unary predicates in FOL or a class in OWL. Meanwhile, roles correspond with binary predicates in FOL or properties (either object property or datatype property) in OWL. In contrast to [22] that utilized *TBox* axioms only to generate questions from an ontology, our approach employs the *ABox* assertions.

The query template needs to be in a specified structure so that the queries generator can recognize the query type and apply the algorithm to replace the set of tokens appropriately. In the context of our problem, a query type corresponds to an IS intent. To simplify the problem, we only took five of eight query types determined in [21] such as definitional, comparison, identification, find a value, and yes/no. The template sample is illustrated in Table 2.

Received: June 1, 2021. Revised: July 21, 2021.

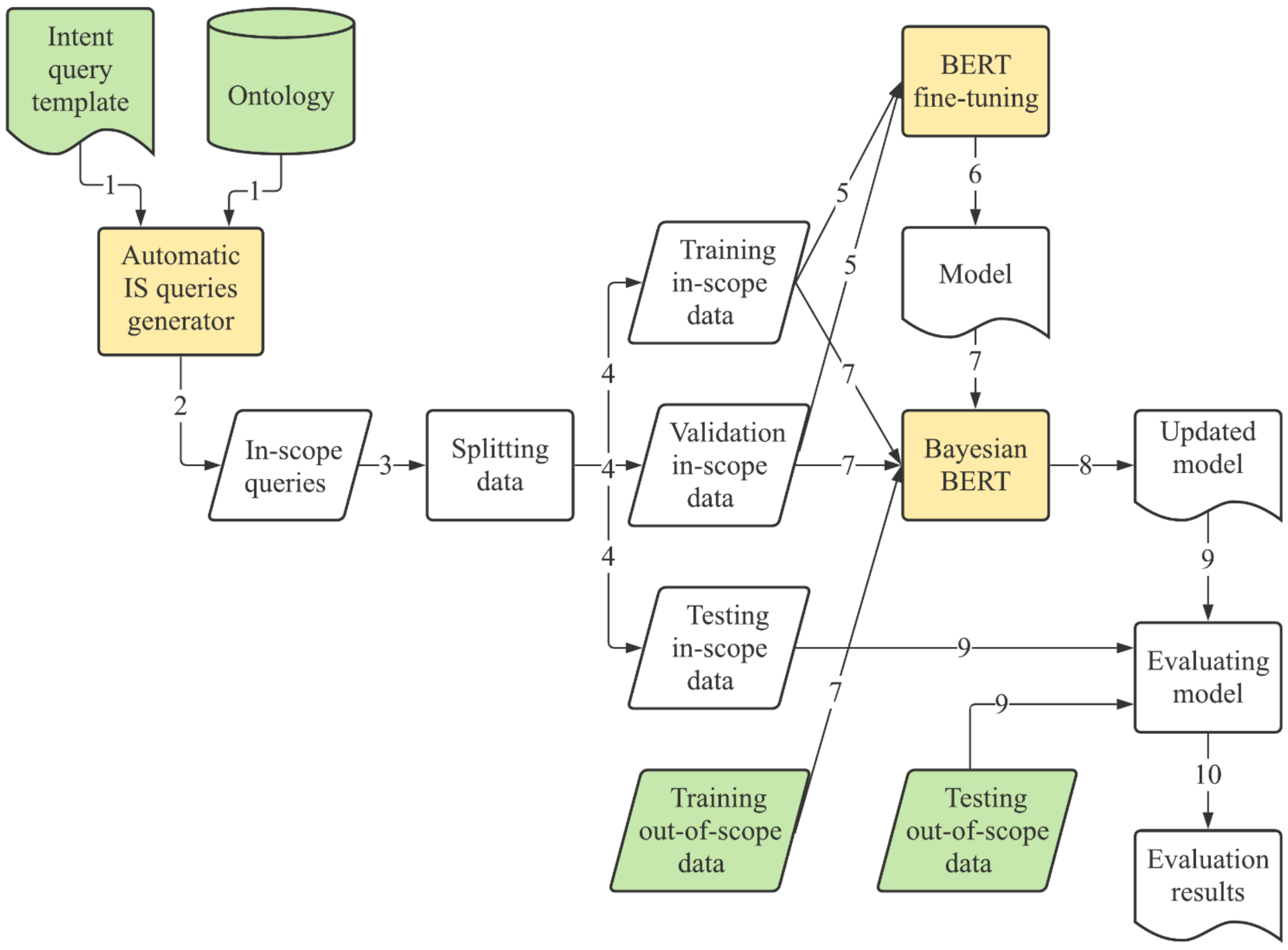


Figure. 2 Research method

Table 2. Sample of the intent query template

Intent	Sample templates
Definitional	-What is a_i -Define a_i
Comparison	-What are the differences between a_i and a_{i+1} -What are the similarities between a_i and a_{i+1}
Identification	-What *a* type(C_i) value(R_i b_i), where b_i value(U_i v_i)
Find a value	-What is *v* of U_i of b_i, where a_i type(C_i) value(R_i b_i)
Yes/no	-Is it true a_i type(C_i) value(R_i b_i), where b_i value(U_i v_i) -True or false: a_i type(C_i) value(R_i b_i), where b_i value(U_i v_i)

As the input of our approach, class C_n in the ontology that becomes the center of the knowledge base needs to be defined initially. Then, as the prerequisites, there should be individuals $\{a_1, a_2, \ldots, a_n\}$ that instantiate class C_n. An individual, a_n, could have several datatype properties $\{U_1, U_2, \ldots, U_n\}$ that have literal values of $\{v_1, v_2, \ldots, v_n\}$. An individual, a_n, also could have a few relations $\{R_1, R_2, \ldots, R_n\}$ with other objects $\{b_1, b_2, \ldots, b_n\}$ that instantiate classes other than C_n. Like any other individuals, these objects could also have their datatype and object properties.

The algorithm used to generate IS queries for class C_n is described in Fig. 3. Besides the query templates, such as definitional queries $\{d_1, d_2, ..., d_n\}$, comparison queries $\{c_1, c_2, ..., c_n\}$, identification queries $\{i_1, i_2, ..., i_n\}$, find a value queries $\{f_1, f_2, ..., f_n\}$, or yes/no queries $\{y_1, y_2, ..., y_n\}$, and the ontology, a reasoner is also needed as a tool to perform reasoning tasks.

3.2 BERT fine-tuning

This subsection explains the neural network building block that takes training and validation IS data input produced by the previous building block and generates a classification model. First, we utilized BERT and evaluated several pre-trained models in the experiment. Then, fine-tuning was applied to the pre-trained network by integrating the training and validation data. The generated model is

Received: June 1, 2021. Revised: July 21, 2021.

```
Automatic IS Queries Generator Algorithm
Input  : ontology, reasoner, $C_n$, query template of definitional, comparison, identification, find a value, yes/no
Output : IS queries results
 1: results ←{}
 2: {$a_1$, $a_2$, …, $a_n$} ← reasoner.getInstances($C_n$)
 3: foreach individual $a_j$ from $a_1$ to $a_n$:
 4:     foreach template d ε definitional:
 5:         query ← replace({$a_i$, $a_j$}) from d
 6:         insert query into results
 7:     end for
 8:     foreach template c ε comparison:
 9:         query ← replace({$a_i$, $a_j$}, {$a_{i+1}$, $a_{j+1}$}) from c
10:         insert query into results
11:     end for
12:     {$R_1$, $R_2$, …, $R_n$} ← ontology.getObjectPropertyAssertionAxioms($a_j$)
13:     foreach objectProperty $R_j$ from $R_1$ to $R_n$:
14:         {$b_1$, $b_2$, …, $b_n$} ← reasoner.getObjectPropertyValues($a_j$, $R_j$)
15:         foreach individual $b_j$ from $b_1$ to $b_n$:
16:             {$U_1$, $U_2$, …, $U_n$} ← ontology.getDataPropertyAssertionAxioms($b_j$)
17:             foreach dataProperty $U_j$ from $U_1$ to $U_n$:
18:                 {$v_1$, $v_2$, …, $v_n$} ← EntitySearcher.getDataPropertyValues($b_j$, $U_j$, ontology)
19:                 foreach literal $v_j$ from $v_1$ to $v_n$:
20:                     foreach template i ε identification:
21:                         query ← replace({$R_i$, Rj}, {$b_i$, $b_j$}, {$U_i$, $U_j$}, {$v_i$, $v_j$}) from i
22:                         insert query into results
23:                     end for
24:                     foreach template f ε find a value:
25:                         query ← replace({$U_i$, $U_j$}, {$a_i$, $a_j$}, {$R_i$, $R_j$}, {$b_i$, $b_j$}) from f
26:                         insert query into results
27:                     end for
28:                     foreach template y ε yes/no:
29:                         query ← replace({$a_i$, $a_j$}, {$R_i$, $R_j$}, {$b_i$, $b_j$}, {$U_i$, $U_j$}, {$v_i$, $v_j$}) from y
30:                         insert query into results
31:                     end for
32:                 end for
33:             end for
34:         end for
35:     end for
36: end for
37: return results
```

Figure. 3 Automatic IS queries generator algorithm

expected to recognize the IS intents with high accuracy using relatively small epochs.

We used densely connected layers in the fine-tuning architecture. We applied one input layer, several hidden layers consisting of a few dense(1024), a dense(512), a dense(256), one output layer, also a dense one, and dropout layers. A dense layer offers learning capabilities based on all of the previous layer's feature combinations. In addition, the dense layers use the Rectified Linear Units (ReLU) activation function, allowing models to learn faster and perform better than predecessor ones such as Sigmoid or Tanh. Furthermore, an Adam optimizer is used, as suggested in [17].

3.3 Bayesian BERT

We incorporated uncertainty assessment into the fine-tuned BERT models produced by the previous building block. The Monte Carlo (MC) dropout sampling technique is used in this building block to model epistemic uncertainty. Dropout is a common regularization strategy that aims to minimize overfitting by randomly setting activations in a given layer to zeros. In the previous building block, the dropout is activated during the training. Meanwhile, in this building block, the dropout is also activated during the inference. By performing so, the fine-tuned BERT becomes Bayesian BERT since the dropout is analogous to using Bernoullis distribution over the network's weights. Moreover, a softmax layer is added to get classes' probability and to compute the prediction at the network's end.

Deactivating dropout during the inference of both IS and OOS queries makes the output deterministic. On the contrary, activating dropout makes the model intrinsically random. It could generate different outputs for the same input since different weights are sampled at each forward pass. It also produces the distribution of the network's output. Thus, uncertainty measures can be obtained. Entropy is used as the uncertainty metric in this approach. Let *C* is classes to predict, and we sample our model *T* times for inference, then the entropy *H* is calculated by Eq. (1). While $\hat{y}_{c,t}$ is the probability of class *c* of the *t*-th sample, the probability of class *c* is calculated by Eq. (2).

$$H = -\frac{1}{C}\sum_{c=1}^{C}\hat{y}_c \log(\hat{y}_c) \quad (1)$$

$$\hat{y}_c = \frac{1}{T}\sum_{t=1}^{T}\hat{y}_{c,t} \quad (2)$$

When the input query does not belong to the distribution of the IS training data, a higher entropy value is expected. Therefore, uncertainty alone can be used to detect an OOS query. If the uncertainty value is higher than a threshold, then the query is classified into an OOS. Standard evaluation metrics such as accuracy, as defined in Eq. (3), precision, as defined in Eq. (4), or recall, as defined in Eq. (5), and F1 score, as defined in Eq. (6), can be used as criteria to find an optimal threshold value. However, the recall metric is more critical in this problem since we do not want an IS query to be falsely classified as an OOS [8]. Let h is a set of uncertainties results, then the uncertainty threshold (UT) can be found using Eq. (7).

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

$$precision = \frac{TP}{TP+FP} \quad (4)$$

$$recall = \frac{TP}{TP+FN} \quad (5)$$

$$F1\ score = 2 \times \frac{precision \times recall}{precision + recall} \quad (6)$$

$$UT = \underset{i \epsilon h}{\operatorname{argmax}}\ recall(i) \quad (7)$$

4. Experimental setup

We created an Indonesian chatbot using a simple ontology of Capsicum morphology [25]. Capsicum is a genus of flowering plants in the Solanaceae family, often known as "peppers" or "chili peppers." Indonesian biodiversity laypeople will use the chatbot to improve their species literacy. The visualization of the ontology is presented in Fig. 4. There are four individuals of the Capsicum plant in the ontology, *Capsicum Annuum*, *Capsicum Chinense*, *Capsicum Frutescens*, and *Capsicum Pubescens*.

The IS queries generator was built using Java and used two necessary dependencies, the OWL API, a Java API for OWL ontologies [26], and HermiT, an ontology reasoner written in OWL [27]. As inputs, the generator accepts a .txt file containing the queries template and .owl file containing the ontology. Five intents, as defined in Table 2, and its' queries templates were created in the Indonesian language. In Table 3, we list the sample of queries in English for the sake of readability. Besides the ontology and the intents' query templates, the list of OOS queries is also needed to be used as training and test data. Thus, we crawled Twitter data with the keyword of "cabai" (Capsicum translation in Indonesian native language) to find queries related to the Capsicum that are not related directly to the knowledge of its morphology. Instead, some of them are related to the price, the cultivation, and the food. The examples of OOS queries are shown in Table 4.

Table 3. List of intents and its' sample queries

Intent	**Sample queries**
Definitional	-What is *Capsicum Chinense*
Comparison	-What are the differences between *Capsicum Chinense* and *Annuum* -What are the similarities between *Capsicum Pubescens* and *Frutescens*
Identification	What plant has a hairy texture stem
Find a value	What is the shape of flower of *Capsicum Pubescens* plant has
Yes/no	- Is it true *Capsicum Frutescens* plant has elongated shape fruit -True or false: *Capsicum Frutescens* plant has elongated shape fruit

Table 4. Example of OOS queries

Topic	**OOS queries**
Cultivation	-How to plant Capsicum? -Which one is easier taking care of Capsicum in the rainy or dry season?
Food	-Where to buy the atomic seasoning with Capsicum powder in Solo? -Is the shrimp broth with Capsicum powder delicious?
Price	-How much does 1 kg of Capsicum cost? -How come the price of Capsicum increased from 25 to 40 thousand?

Received: June 1, 2021. Revised: July 21, 2021.

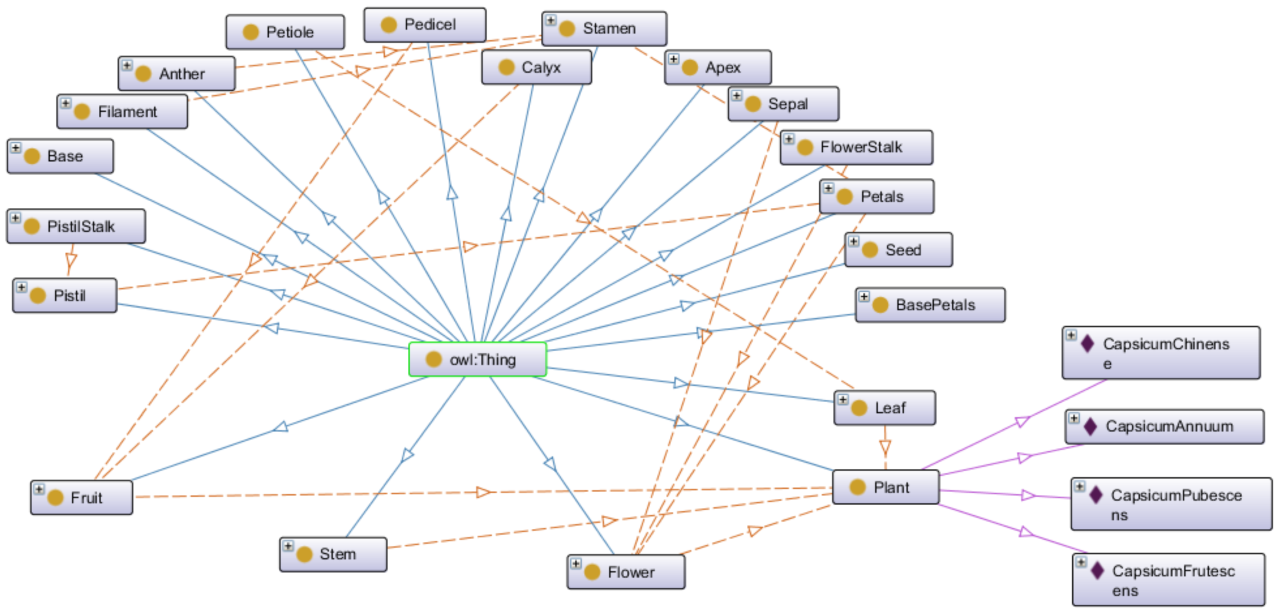


Figure. 4 Capsicum ontology

We evaluated three versions of IndoBERT, a pre-trained language model based on BERT architecture for the Indonesian language. IndoNLU model trained 12 corpus collections, both formal and colloquial Indonesian sentences, containing 23.43 GB of texts, four billion words, with around 250 million sentences [28]. Moreover, [29] trained three primary sources containing 220 million sentences. Meanwhile, [30] trained 16 GB of raw texts, around two billion words.

5. Results and discussions

We obtained 548 IS queries after applying the generator algorithm. We divided the queries into 324 training data, 112 validation data, and 112 test data, as shown in Table 5. We also collected 236 OOS queries from Twitter. Since we would like to test the reliability of the proposed method in detecting OOS, we used minimal OOS queries related to price topics only as training data.

We tried three versions of IndoBERT models and tuned two hyperparameters to classify the IS queries, such as learning rates (LR) and batch sizes (BS). We selected 5e-5, 3e-5, and 2e-5 for the learning rates. Furthermore, we chose 16 and 32 for batch size. These values are based on suggestions from [17]. However, we used 20 epochs, different from the recommendation, where 2-4 epochs are suggested. The architecture of our model is shown in Fig. 5. Meanwhile, a dropout rate of 0.2 is applied when training and 0.1 while inferencing with MC iterations, T = 50. The classification performance results, such as accuracy (A), precision (P), recall (R), and F1 score, are shown in Table 6.

As shown in the results, all models classified IS queries with an F1 score higher than 80%. Even two versions of IndoBERT models, [28] and [30], managed to classify IS queries with a 100% F1 score. Meanwhile, [29] got the highest F1 score of 98% when using a learning rate of 5e-5 and a batch size of 16. Thus, we employed this learning rate and batch size value for subsequent evaluations.

The results indicated that both generated IS datasets and classification algorithms are sufficient. We emphasize this finding by experimenting with the prediction of the same datasets using Google DialogFlow. The generated IS train and validation dataset were fed into the training phrases of each intent in the DialogFlow. It managed to achieve a 100% F1 score when classifying the IS intents on the test dataset. Meanwhile, the confidence scores are between 0.69-0.82, as shown in Table 7. The higher values imply higher probabilities that the detected intents were correct.

Table 5. IS dataset

Intent	Training	Validation	Test
Definitional	20	8	8
Comparison	20	8	8
Identification	114	39	39
Find a value	84	28	28
Yes/no	86	29	29

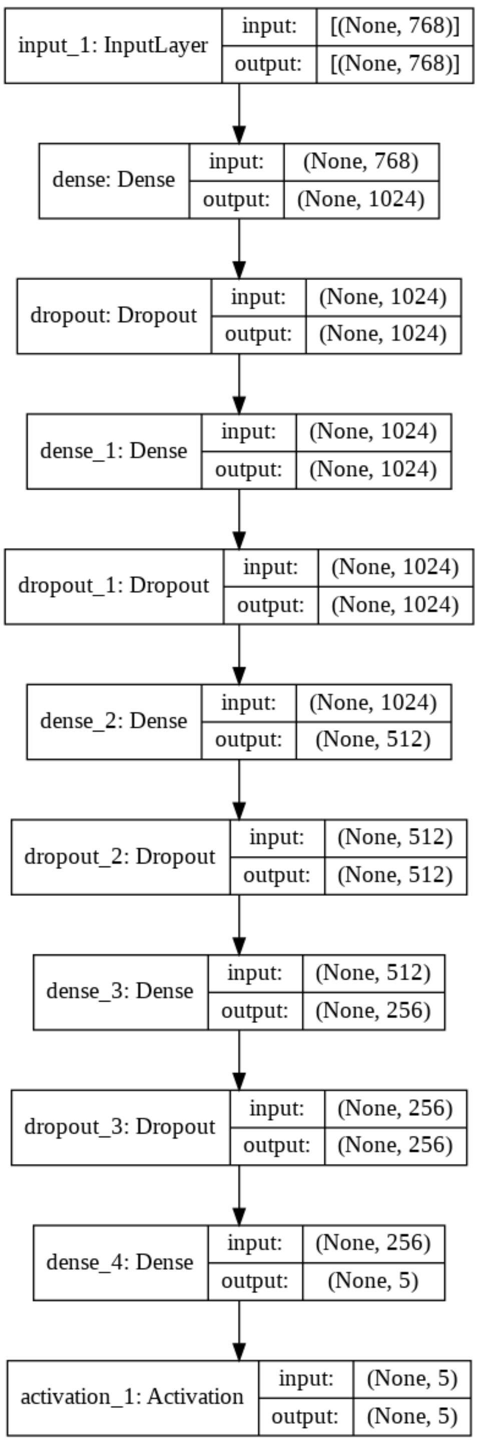


Figure. 5 Model's architecture

The comparison of the IndoBERT models is shown in Fig. 6. Model of [28, 30] converged faster than [29]. At the epoch 10 and 12, the accuracy of [28, 30] yielded 100%, respectively. Meanwhile, [29] reached an accuracy of 100% at epoch 18.

Typically, a selected UT should maximize the F1 score on the training data in determining an OOD. However, the case is different with OOS queries.

Table 6. IS' intents classification performance

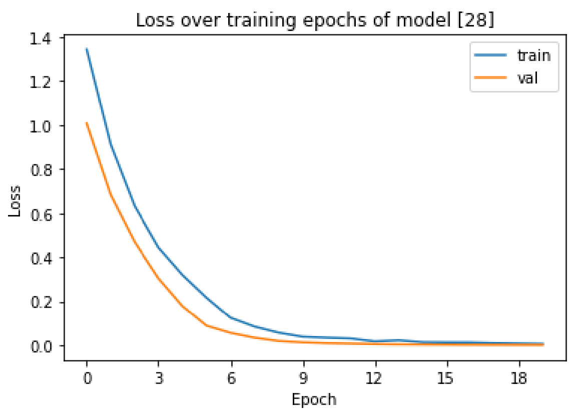
IndoBERT version	**LR**	**BS**	**A**	**P**	**R**	**F1**
[28]	5e-5	16	1.00	1.00	1.00	1.00
[28]	3e-5	16	1.00	1.00	1.00	1.00
[28]	2e-5	16	1.00	1.00	1.00	1.00
[28]	5e-5	32	1.00	1.00	1.00	1.00
[28]	3e-5	32	1.00	1.00	1.00	1.00
[28]	2e-5	32	0.99	0.99	0.99	0.99
[29]	5e-5	16	0.98	0.98	0.98	0.98
[29]	3e-5	16	0.95	0.95	0.95	0.95
[29]	2e-5	16	0.93	0.94	0.93	0.93
[29]	5e-5	32	0.96	0.96	0.96	0.95
[29]	3e-5	32	0.93	0.94	0.93	0.93
[29]	2e-5	32	0.84	0.80	0.84	0.81
[30]	5e-5	16	1.00	1.00	1.00	1.00
[30]	3e-5	16	1.00	1.00	1.00	1.00
[30]	2e-5	16	1.00	1.00	1.00	1.00
[30]	5e-5	32	1.00	1.00	1.00	1.00
[30]	3e-5	32	1.00	1.00	1.00	1.00
[30]	2e-5	32	0.98	0.98	0.98	0.98

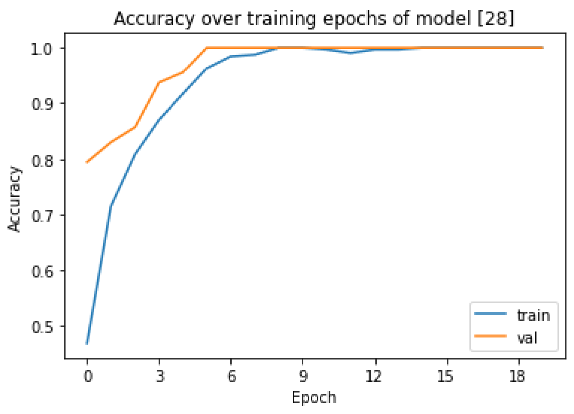
Table 7. DialogFlow's classification performance

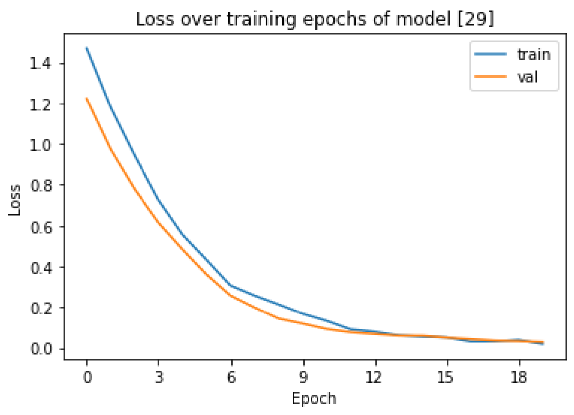
Intent	**F1**	**Detection confidence**
Definitional	1.00	0.82
Comparison	1.00	0.79
Identification	1.00	0.69
Find a value	1.00	0.75
Yes/no	1.00	0.73

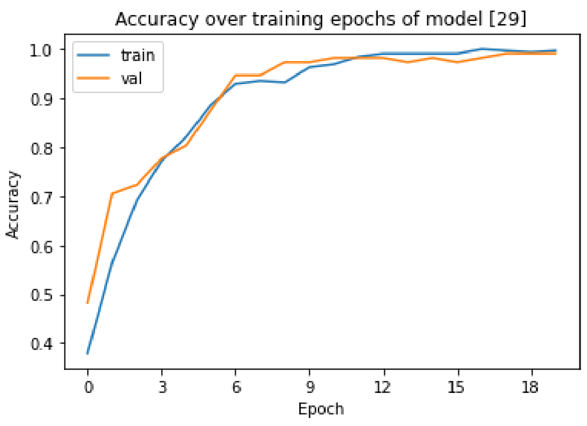
The UT should maximize the recall. Fig. 7 shows a UT example that maximizes the recall (a) and a UT that maximizes the F1 score (b). Two experiments for each approach and each model with different OOS training and test data sizes were performed. The results are shown in Table 8. Model of [28] achieved the best performance where the F1 score yielded 86% and 74% using 10 and 5 OOS training data, respectively. Meanwhile, the training data sizes did not significantly affect the model's performance of [29] and [30], where the F1 score yielded 53-58% and 68%, respectively.

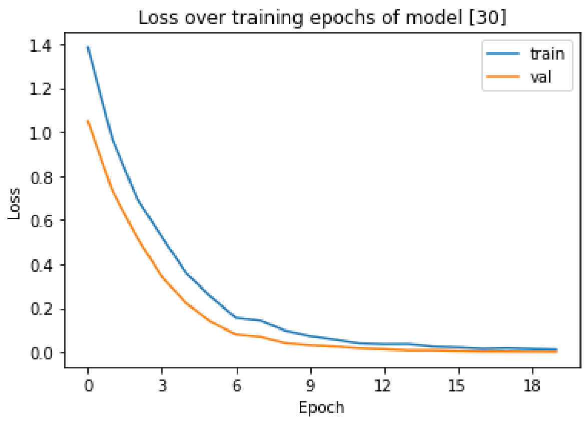
Since the OOS training data size was tiny, there was no significant difference between approach (a) and approach (b). Nevertheless, the differences were explained when experimenting with the model of [29] using 10 training data. It showed the trade-offs between the approaches. The recall yielded 99%, but the F1 score was only 53% using approach (a). However, in approach (b), the F1 score increased to 58%, but the recall decreased to 90%.


(a)


(b)


(c)


(d)


(e)

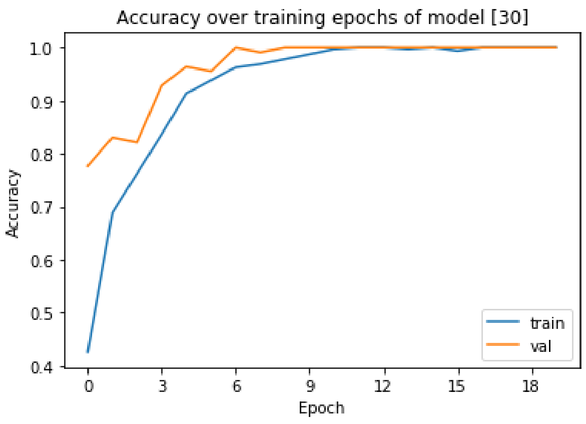

(f)

Figure. 6 Losses and accuracies over training epochs of: (a),(b) model [28]; (c),(d) model [29]; and (d),(e) model [30]

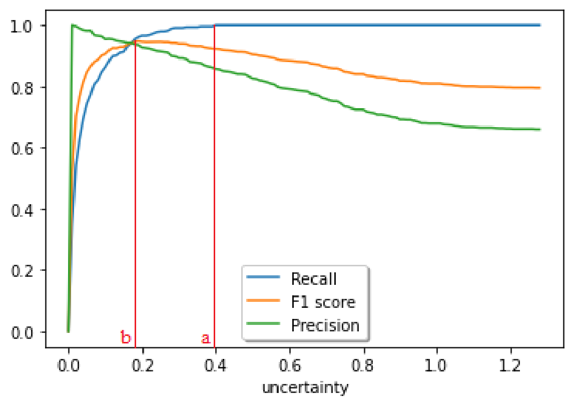


Figure. 7 Two uncertainty threshold approaches

To conduct more evaluation, we compared the performance results with the DialogFlow. The OOS training dataset was supplied into training phrases of default fallback intent. Its performance on predicting the OOS test queries was not as good as classifying IS intents, although the recall yielded 100%. The performance results are represented in Table 9.

In the first experiment, the F1 score yielded only 69%, 17% lower than our results. Since the training phrase sizes were very small, there was a possibility

Table 8. IS and OOS query classification performance results

IndoBERT version	Train OOS	Test OOS	Approach (a)					Approach (b)				
			UT	A	P	R	F1	UT	A	P	R	F1
[28]	10	226	0.28	0.90	0.76	0.98	0.86	0.28	0.90	0.76	0.98	0.86
[29]	10	226	1.11	0.43	0.37	0.99	0.53	0.83	0.57	0.43	0.90	0.58
[30]	10	226	0.54	0.69	0.51	0.99	0.68	0.54	0.69	0.51	0.99	0.68
[28]	5	231	0.41	0.77	0.59	0.99	0.74	0.41	0.77	0.59	0.99	0.74
[29]	5	231	0.98	0.52	0.41	0.98	0.57	0.98	0.52	0.41	0.98	0.57
[30]	5	231	0.43	0.69	0.51	0.99	0.68	0.43	0.69	0.51	0.99	0.68

Table 9. DialogFlow performance results

Train OOS	Test OOS	A	P	R	F1
10	226	0.71	0.53	1.00	0.69
5	231	0.42	0.36	1.00	0.53
25	211	0.91	0.79	1.00	0.88

that DialogFlow uses a more rule-based than ML approach in classifying the intents. This statement is strengthened when we employed smaller and larger OOS training data in the subsequent experiments.

In the second DialogFlow experiment, its precision dropped to 36%, as shown in Table 9, since the training data size is reduced to five. However, the performance increases when we added more data. The results were comparable to our approach when DialogFlow used 25 training data. The results indicated that the uncertainty estimation method proposed in this paper is better than the threshold on the classifier's probability estimate used by DialogFlow when implementing a few-shot approach to detect an OOS. This finding supports [31] and [32], where uncertainty quantification using Bayesian MC Dropout (MCD) is sufficient to predict OOD, or OOS in our case.

6. Conclusion and future works

A novel framework to address challenges on building a KBC from scratch has been proposed in this study. Furthermore, an automated IS queries generator from an ontology has been introduced to make the IS data in tackling data acquisition problems. As a result, we managed to generate 548 IS queries by using the proposed algorithm with a minimal queries template as an input.

Moreover, the proposed FSL, the Bayesian BERT with MCD uncertainty estimation method, solved detecting not only IS intents but also OOS intent using a few training data. The best performance yielded a 100% F1 score for IS intents. When predicting OOS, the performance overcame a threshold on the classifier's probability estimation approach used by Google Dialogflow by 17% F1 score when using 10 OOS training data and 21% when using 5 data.

We will investigate a more seamless approach to make IS and OOS data using an advanced NLG like Generative Pre-trained Transformer (GPT) family for future works. Furthermore, we would like real users to evaluate our chatbot. Therefore, feedbacks from laypeople is needed to perfect the chatbot.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, LPM and ZA; methodology, LPM; software, LPM; validation, HFM, AI, and ZA; formal analysis, ZA; investigation, LPM, HFM, AI; resources, LPM; data curation, LPM, HFM, AI, TD, and ADF; writing—original draft preparation, LPM; writing—review and editing, ZA and DSR; visualization, LPM; supervision, ZA; project administration, DSR and ZA; funding acquisition, DSR, TD, and ADF.

Acknowledgments

This work is supported by the funding from the Ministry of Research and Technology of the Republic of Indonesia (Ristek) - Managing Agency of Endowment Fund for Education of the Republic of Indonesia (LPDP) with the contract number: 39/EI/PRN/2020.

References

[1] A. A. Mlouk and L. Jiang, “KBot: A Knowledge Graph Based ChatBot for Natural Language Understanding Over Linked Data”, *IEEE Access*, Vol. 8, pp. 149220-149230, 2020.

[2] K. Chung and R. C. Park, “Chatbot-based heathcare service with a knowledge base for cloud computing”, *Cluster Comput.*, Vol. 22, No. 1, pp. 1925-1937, 2019.

[3] M. Nuruzzaman and O. K. Hussain, “IntelliBot:

A Dialogue-based chatbot for the insurance industry”, *Knowledge-Based Syst.*, Vol. 196, p. 105810, 2020.

[4] S. Ouf and N. Hamza, “The Role of Machine Learning to Fight COVID-19”, *Int. J. Intell. Eng. Syst.*, Vol. 14, No. 2, pp. 121-135, 2021.

[5] H. A. Zubaide and A. A. Issa, “OntBot: Ontology based chatbot”, In: *Proc. of International Symposium on Innovations in Information and Communications Technology*, pp. 7-12, 2011.

[6] S. D. Shivnikar, H. Arora, and H. B. S. S, “A character representation enhanced on-device Intent Classification”, In: *Proc. of the 17th International Conference on Natural Language Processing*, pp. 40-46, 2020.

[7] T. A. Moslmi, M. G. Ocaña, A. L. Opdahl, and C. Veres, “Named Entity Extraction for Knowledge Graphs: A Literature Overview”, *IEEE Access*, Vol. 8, pp. 32862-32881, 2020.

[8] S. Larson, A. Mahendran, J. J. Peper, C. Clarke, A. Lee, P. Hill, J. K. Kummerfeld, K. Leach, M. A. Laurenzano, L. Tang, and J. Mars, “An evaluation dataset for intent classification and out-of-scope prediction,” In: *Proc. of EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pp. 1311-1316, 2020.

[9] L. P. Manik, A. F. Syafiandini, H. F. Mustika, A. F. Abka, and Y. Rianto, “Evaluating the Morphological and Capitalization Features for Word Embedding-Based POS Tagger in Bahasa Indonesia”, In: *Proc. of 2018 International Conference on Computer, Control, Informatics and its Applications: Recent Challenges in Machine Learning for Computing Applications, IC3INA 2018 - Proceeding*, 2019.

[10] T. E. Lin and H. Xu, “Deep Unknown Intent Detection with Margin Loss”, In: *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5491-5496, 2019.

[11] T. E. Lin and H. Xu, “A post-processing method for detecting unknown intent of dialogue system via pre-trained deep neural network classifier”, *Knowledge-Based Syst.*, Vol. 186, p. 104979, 2019.

[12] U. Bharti, D. Bajaj, H. Batra, S. Lalit, S. Lalit, and A. Gangwani, “Medbot: Conversational Artificial Intelligence Powered Chatbot for Delivering Tele-Health after COVID-19”, In: *Proc. of 2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pp. 870-875, 2020.

[13] S. Ryu, S. Kim, J. Choi, H. Yu, and G. G. Lee, “Neural sentence embedding using only in-domain sentences for out-of-domain sentence detection in dialog systems”, *Pattern Recognit. Lett.*, Vol. 88, pp. 26-32, 2017.

[14] P. Cavalin, V. H. A. Ribeiro, A. Appel, and C. Pinhanez, “Improving Out-of-Scope Detection in Intent Classification by Using Embeddings of the Word Graph Space of the Classes”, In: *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3952-3961, 2020.

[15] E. H. Yilmaz and C. Toraman, “KLOOS: KL Divergence-Based Out-of-Scope Intent Detection in Human-to-Machine Conversations”, In: *Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2105-2108, 2020.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is All you Need”, In: *Proc. of Advances in Neural Information Processing Systems*, Vol. 30, 2017.

[17] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, In: *Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171-4186, 2019.

[18] S. G. Carvajal and E. C. G. Merchán, “Comparing BERT against traditional machine learning text classification”, *arXiv:2005.13012*, 2021.

[19] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”, In: *Proc. of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, pp. 1050-1059, 2016.

[20] A. Kendall and Y. Gal, “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?”, In: *Proc. of Advances in Neural Information Processing Systems*, Vol. 30, 2017.

[21] V. K. Chaudhri, P. E. Clark, A. Overholtzer, and A. Spaulding, “Question Generation from a Knowledge Base”, In: *Proc. of Knowledge Engineering and Knowledge Management*, pp. 54-65, 2014.

[22] T. Alsubait, B. Parsia, and U. Sattler, “Ontology-Based Multiple Choice Question

Generation", *KI - Künstliche Intelligenz*, Vol. 30, No. 2, pp. 183-188, 2016.

[23] M. d'Aquin and E. Motta, "Extracting Relevant Questions to an RDF Dataset Using Formal Concept Analysis", In: *Proc. of the Sixth International Conference on Knowledge Capture*, pp. 121-128, 2011.

[24] G. D. Giacomo and M. Lenzerini, "TBox and ABox Reasoning in Expressive Description Logics", In: *Proc. of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, pp. 316-327, 1996.

[25] Z. Akbar, H. F. Mustika, D. S. Rini, L. P. Manik, A. Indrawati, A. D. Fefirenta, and T. Djarwaningsih, "An Ontology-Driven Personalized Faceted Search for Exploring Knowledge Bases of Capsicum", *Future Internet*, Vol. 13, No. 7, p. 172, 2021.

[26] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL Ontologies", *Semant. Web*, Vol. 2, No. 1, pp. 11-21, 2011.

[27] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang, "HermiT: An OWL 2 Reasoner", *J. Autom. Reason.*, Vol. 53, No. 3, pp. 245-269, 2014.

[28] B. Wilie, K. Vincentio, G. I. Winata, S. Cahyawijaya, X. Li, Z. Y. Lim, S. Soleman, R. Mahendra, P. Fung, S. Bahar, and A. Purwarianti, "IndoNLU: Benchmark and Resources for Evaluating Indonesian Natural Language Understanding", In: *Proc. of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pp. 843-857, 2020.

[29] F. Koto, A. Rahimi, J. H. Lau, and T. Baldwin, "IndoLEM and IndoBERT: A Benchmark Dataset and Pre-trained Language Model for Indonesian NLP", In: *Proc. of the 28th International Conference on Computational Linguistics*, pp. 757-770, 2020.

[30] S. L. Sariwening, "IndoBERT: Transformer-based Model for Indonesian Language Understanding", *M.S. Thesis, Department of Information Management, National Taiwan University of Science and Technology*, 2020.

[31] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges", *Inf. Fusion*, Vol. 76, pp. 243-297, 2021.

[32] A. Schwaiger, P. Sinhamahapatra, J. Gansloser, and K. Roscher, "Is Uncertainty Quantification in Deep Learning Sufficient for Out-of-Distribution Detection?", In: *Proc. of AISafety@ IJCAI*, 2020.