A Comparison of Module Selection Strategies for Modular Dialog Systems

Philine Görzig[1], *Jan Nehring*[2], *Stefan Hillmann*[1], *Sebastian Möller*[1,2]

[1] *Technische Universität Berlin,* [2]*DFKI Berlin*
p.goerzig@tu-berlin.de

1 Introduction

Dialog systems (DS), also known as conversational agents or chatbots, have become increasingly popular in recent years. However, the creation and scaling of these systems can be a challenge. Each system must be tailored to a specific use case, and adding new functionality increases complexity. Additionally, transferring a system to a new use case or combining multiple systems into one can be difficult. One approach to this problem is to divide the dialog systems up into sub-systems or to use already completed dialog systems as part of a larger system. Each sub-system becomes a module of an expandable Modular Dialog System (MDS) [1]. In an MDS, a module selector (MS) chooses the appropriate module to respond to a user's input. The approach allows for scalability across multiple domains and topics without requiring repeated manual work. An MDS presents its own set of challenges, such as ensuring all components work well together and managing the additional error introduced by the MS.

2 Related Work

In MDS, the users are unaware that they are talking to multiple dialog systems [2] unless they are purposefully made aware of this fact [3]. The MS chooses which module should respond to a received message. MS is a classification problem that is typically solved by sending the utterance to each module and picking the module that returns the highest confidence value. One proposed theoretical setup for such an MDS is the bot-orchestrator [4] for combining domain expert bots.

Nehring and Ahmed [1] compare normalization methods on the confidence values of the modules and their effect on the performance of MDSs with varying numbers and sizes of modules. The results show that the performance decreases with an increasing number of modules and that the normalization methods show a slight performance improvement with balanced modules. All but one even decrease the performance with imbalanced modules as opposed to using no normalization.

Other MDSs such as CLARA [3] and AIDA [5] also have a central unit that selects the appropriate module, with decision-making processes that vary depending on the system. CLARA combines four modules, and if the confidence of one module does not pass a threshold, the utterance is passed on to the next. AIDA has six modules, and the MS chooses a module based on many factors, including user utterance, states of the modules, interaction history, task hierarchy, and user profile. COBOTS [6] is another MDS with domain expert modules for eleven domains. A machine learning model is trained for each separate domain, and the modules are then picked based on their domain prediction score. IBM Watson is used to decide whether a domain module or a backup information retrieval system should respond.

3 Setup

In this paper, multiple scenarios are set up and compared to each other based on how they perform. Each scenario uses a different combination of features and classification models. The various features are extracted from the module response and the user utterance. These module classifiers range in complexity from simple normalization methods to fully connected, multilayer, feedforward networks (MLP) and text-classification models such as BERT [7]. The scenarios are evaluated on the HWU64 dataset [8]. The three modules integrated into the MDS use Rasa NLU [9], Google Dialogflow [10], and IBM Watson Assistant [11]. The performance of the MS is measured with the f1-score of the module classification. Additionally, the performance of the complete MDS is measured through the f1-score of the intent classification. The performance is compared to the intent classification performance of dialog systems that are not modular (singular-DS). We use a framework from Nehring and Ahmed [1], which creates an MDS by randomly distributing the intents of a dataset onto the modules.

3.1 Features

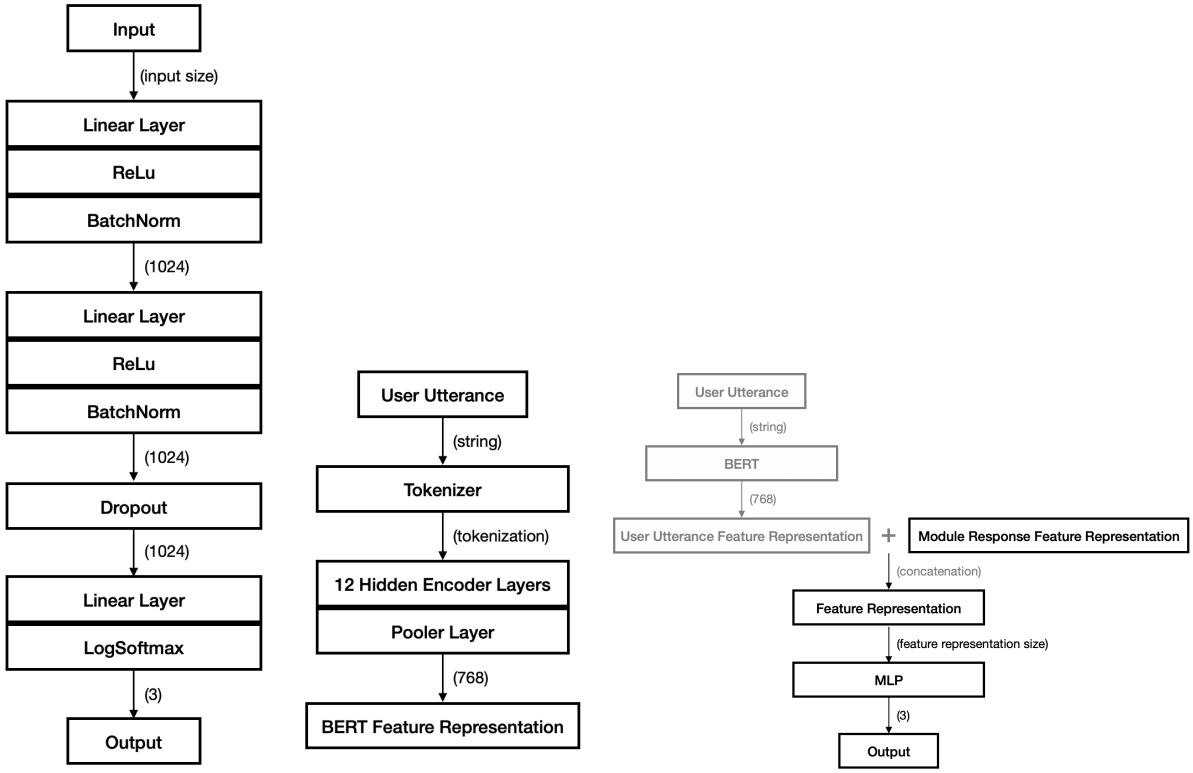
The confidences seem like the obvious choice for MS since each dialog system returns a confidence to show how sure it is that it classified the intent correctly. However, the confidence does not show how sure the dialog system is that the intent belongs to it. Dialog systems are usually not built to work in a modular setting and are not built to work in combination with one another. They are built with the expectation that they are each trained on the full scope of intents they are used with.

We explore four different MS inputs: One is the module's confidence for the most likely intent (*max*). The second uses the confidence and the knowledge of which intent was detected (*intent*). This is done by creating a vector with the same length as the number of unique intents in the module's training data. The third focuses on the entities, creating a vector that contains the entity confidence values for each detected entity (*entity*). A combination of all three (*all*) contains everything that can be extracted from the module responses. The last explored input is the user utterance.

3.2 Models

The MLP model has two linear layers with 1024 neural units each. Each linear layer is concluded with a rectified linear unit (ReLU) activation function and a batch normalization layer (BatchNorm). The model is then concluded with a dropout layer (dropout probability of 0.1), a final linear layer that maps the data to the output dimension (equal to the number of modules), and lastly, a log softmax activation function (see Fig. 1a). The input size varies depending on the feature representation used. During training, the model is optimized with the Adam optimizer using the negative log-likelihood loss (NLLoss). The learning rate and batch size were chosen through a hyperparameter grid search. The combinations of both hyperparameters depend on the specific combination of features and models in each scenario. While a learning rate of 2e-5 performed best in all scenarios, the batch sizes vary depending on the scenario.

In addition to the information the MS can extract from the module responses, the MS also holds the utterance that is passed on to the modules as an additional feature for the MS. Natural language classifiers are very powerful. Especially the BERT model works well on classification problems. The utterance is processed by the NLU model BERT. The string is first tokenized with the BERT tokenizer that has truncation, and padding strategies on. The token representation of the utterance is then run through a BERT model to turn it into features the MLP can use for



(a) MLP **(b)** BERT **(c)** Full Model

Figure 1 – The separate architectures of the MLP and BERT models used. a) The architecture of the MLP. The MLP has two hidden layers consisting of 1024 units. Each hidden layer has a ReLu activation function and a BatchNorm layer. The output is normalized with a LogSoftmax function. b) The architecture of the BERT model. The user utterance is tokenized and run through 12 hidden encoder layers and a pooler layer. The output has 768 units. c) The combined model consists of the MLP and the BERT model. The input into the MLP is a feature representation that is either the module response or the user utterance feature representation. When both feature representations are used, they are concatenated.

classification. The BERT model consists of twelve hidden encoder layers and a pooler layer. All these layers have a dimensionality of 768. The pooler layer consists of a linear layer and a *tanh* activation function (see Fig. 1b). The BERT model is built into the classification model and learns with backpropagation. Only the tokenization of the BERT model is static. The BERT and the MLP model are combined by concatenating the pooler output of the BERT model with a vector that holds all additional features before passing this new feature vector to the MLP (see Fig. 1c).

3.3 Dataset

We use a large, noise-included benchmark dataset for textual data, the HWU64 dataset [8]. It is a benchmark dataset for task-oriented dialog systems, containing 25,716 annotated user utterances in English. Each utterance is assigned to one of 68 intents, belonging to one of 18 domains from the topic of home automation. The dataset contains no out-of-scope data, but since the data is divided among the modules, each module will see data outside of its scope. The dataset was generated through crowd-sourcing and is considered a more difficult NLU dataset [12]. The distribution of the intents and domains is uneven, with some intents having as few as 412 utterances and others having as many as 6102.

For our experiment, we produce module datasets of varying sizes by randomly distributing the 18 domains onto three modules, meaning 6 of the 18 domains are assigned to each module.

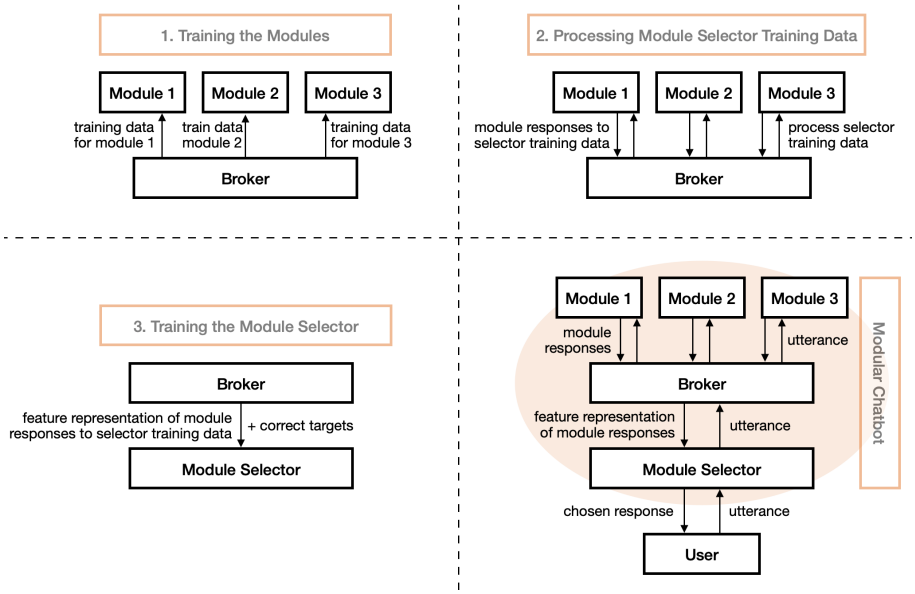


Figure 2 – The setup of the MDS framework. 1. The modules are trained. The dataset is split into three, and each module receives its dataset to train on. 2. The modules process the training data for the MS. Each module receives the complete MS training set and returns its responses to each utterance to the Broker. 3. The MS is trained. The Broker passes the responses of the modules to the MS, including the user utterance and the correct target module. 4. The MDS is in operation. The user sends an utterance to the MS. The utterance is passed through the Broker to each module. The module responses are translated and combined into a feature representation. The MS uses the feature representation and/or the utterance itself to pick one module to respond to the user.

Module datasets of varying sizes are purposefully not prevented since it often occurs to have modules that vary in importance, size, and versatility. The domain separation between the modules is maintained to create a realistic module split where each module has different use cases. The dataset is also randomly split into a test dataset, a validation dataset, and a training dataset, with 15% of the data going to the test and validation datasets and the remaining 70% to the training dataset. To avoid overfitting, the training dataset is further divided into two sets: one for training the modules and one for training the MS. Specifically, 75% of the training data is used to train the modules, and another 75% is used to train the MS, resulting in 50% of the training dataset being shared between the two sets.

3.4 Runs

The evaluation is done on five separate runs with different data splits to rule out any dependence on the split and to ensure an equal distribution of data between the modules. The final performance scores are the average of the five runs. The main scenario is evaluated with three modules, each using a different NLU framework. Additional scenarios are used for comparison as a secondary experiment and are evaluated on only one run.

3.5 Framework

Our MDS is built with a framework for MDS [1]. The framework constructs an MDS by combining two or more fully functioning dialog systems as modules. The complete MDS consists of the MS, the Broker, and the fully trained modules. When the user sends an utterance to the MDS, the utterance gets sent to the MS. The MS passes the utterance to the Broker. The Broker sends the utterance to every module and turns the module responses into the feature representation the MS requires. The MS then receives the feature vector that represents the responses and uses them to determine which of the modules should answer (see Fig. 2).

	all				targets			
	min	max	avg	std (of avg)	min	max	avg	std (of avg)
Rasa	0.2	1	0.88	0.03	0.28	1	0.99	0.002
Dialogflow	0	1	0.56	0.14	0	1	0.95	0.010
Watson	0	1	0.55	0.06	0	1	0.96	0.007

Table 1 – Average, minimum, maximum, and standard deviation of confidence scores using Rasa, Dialogflow, or Watson. Dialogflow and Watson show very similar behavior. The table is split between the statistics for all confidence values and statistics for confidence values of the data points assigned to the module (targets).

4 Experiment

The following scenarios are first evaluated in terms of MS and compared to an MS baseline. The intent classifications of the MDSs with the best-performing MS methods are then compared to intent classification baselines and run with varying module setups.

4.1 Baselines

The baseline for MS is the commonly used method of choosing the module with the highest confidence value (*max*). Additional intent classification baselines are set to compare the performance of the MDS with traditional, non-modular DSs (*singular-DS*) using the same data. These baselines are the intent classification performances of singular-DSs trained on all domains and intents instead of splitting the domains onto modules. Each singular-DS uses either Rasa, Dialogflow, or Watson.

4.2 Normalization

The evaluated normalization methods for the confidence scores are simple min-max normalization, average normalization, and a combination of both. Tab. 1 shows the average, minimum, maximum, and standard deviation of the confidences for each NLU framework (Rasa, Dialogflow, and Watson). The Dialogflow and Watson confidences always range from 0 to 1, while the Rasa modules have a minimum confidence of around 0.2. The average confidences also show that the Rasa modules generally return higher confidence than Dialogflow and Watson modules. This means that min-max normalization methods would not affect Dialogflow and Watson scores as they already range from 0 to 1, but they could adapt Rasa scores to those of the other NLU frameworks. The statistics also show that the average confidences are much higher for all frameworks when looking at the target confidence scores only.

4.3 Scenarios

In addition to normalization, the MLP models, with and without BERT, are combined with the introduced features, resulting in six further scenarios for MS: MLP+conf, MLP+intent, MLP+entity, MLP+all, BERT+intent, and BERT+all.

5 Results

5.1 Module Selection

The performance of different normalization methods, MLP models without BERT, and MLP models with BERT integrated are compared to the baseline by using the f1-score. The baseline

	f1	recall	precision
Rasa	0.65	0.96	0.50
Dialogflow	0.64	0.50	0.91
Watson	0.65	0.54	0.92

Table 2 – The recall and precision scores of the MS-baseline over each NLU framework.

	-	*confidence*	*intent*	*entity*	*all*
random, baseline	0.33	**0.66**	-	-	-
norm	-	**0.74**	-	-	-
MLP	-	0.82	0.87	0.74	**0.88**
text-classification	0.94	-	0.94	-	**0.95**

Table 3 – Overview of MS F1-scores, the MS methods, and the features used. All evaluated MDSs have three modules. The MS baseline is picking the maximum confidence. The method *text-classification* uses the *user utterance* as a feature. When no additional features are used, the MS is done without first communicating with the modules. Combinations without a score have not been evaluated.

always selects the module with the highest confidence score. Tab. 1 shows that Rasa modules have a much higher average confidence score. Consequently, *max* tends to select the Rasa module over the Dialogflow and Watson modules. Rasa's high recall and low precision scores confirm this behavior and can be found in Tab. 2.

The effectiveness of normalization depends on the behavior of the confidence scores as well. Normalization is mostly useful when the confidences of the modules show varying behavior, which is the case when combining different NLU frameworks in one system. All normalization methods improve the baseline, but the best results occurred when normalizing with the min, max, and average target confidences instead of the respective overall confidences.

Tab. 3 gives an overview of each module selection strategy and its performance. Every proposed MS method is an improvement from the baseline MS. The *text-classification* MS method (BERT) outperforms every other scenario. After *text-classification*, the MLP showed the best results. The table depicts the results of the best-performing normalization method, which is a combination of the min-max and average normalization methods on the target confidences.

5.2 Intent Classification

The intent classification performance is dependent on MS and the performance of each module's NLU. We compare the scores of the MDS scenarios to the singular-DS baselines. All results are found in Tab. 4. The MDS scenarios have a higher score than the singular-DS, indicating that the MDS improves the singular-DS scenario. Since MS is an additional source of error, we did not expect to exceed the singular-DS baseline. Further, the performance of MDSs using multiple different NLU frameworks (combined MDS) is compared to systems using the same framework for all modules.

6 Discussion

All evaluated features hold enough information to build a good enough classifier to distinguish the three modules. However, some setups work better than others. The text classification MS performs very well on the module classification task, and it is suggested that a text classification model is considered for the MS.

The performance of the proposed MS methods varies for each NLU framework. Normal-

	sg.-DS	max	norm	MLP		BERT		
				+conf	*+all*		*+intent*	*+all*
R+D+W	-	.6151	.6877	.7649	.8162	.8561	.8558	**.8573**
Rasa	.8336	.7306	.7298	.6449	.7803	.8537	**.8609**	.8560
Watson	.8038	.7371	.7371	.7435	.7808	.8130	**.8172**	.8154
Dialogflow	.7450	.7960	.7960	.7917	.8123	**0.8427**	.8393	.8403

Table 4 – Averaged micro f1-scores over the five runs of intent classification where all setups learned the same intents. The setups include the singular-DSs (*sg.-DS*) and the MDSs with three modules using only Rasa, Dialogflow, and Watson, as well as one using one of each (*R+D+W*). The best score of each combination is highlighted in bold. The best score overall is underlined. We compare the performance of normalization (*norm*), MLP models without Bert (*MLP*), and MLP models with BERT (*BERT*) to the max-baseline (*max*).

ization improves the performance of combined MDSs. However, when all modules use the same framework, the normalization methods cannot beat the baseline. This indicates that the NLU frameworks already produce fairly comparable scores across independent modules. The overall performance of the combined MDS is strong and can compete with MDS that use just one framework. The Rasa MDS has the best performance overall.

Using MDS has its trade-offs, as it adds the MS, another decision component, to the DS. The success of an MDS is dependent on the efficiency of the selection methods of the MS component. However, even if the performance of the MDS is not better than the performance of a singular-DS, it makes it possible to combine completed dialog systems, which requires less effort than building a singular-DS with combined capabilities. Additionally, the modular approach also allows the incorporation of completed dialog systems that can no longer be changed. Furthermore, modular systems have other advantages, such as being more scalable, expandable, and combinable. It is also worth noting that the number of integrated modules and the use of a good MS are crucial in maintaining high classification scores. In our evaluated scenarios, the performance of MDS is shown to beat the performance of singular-DS, despite the added source of error through the module selection.

7 Conclusion and Future Work

Extracting more information from the module responses changes the performance of MS. Even the entities provide enough information to build a classifier that beats the baseline. Combining all this information for MS works best. The full module response contains detailed information about the recognized intent and the intent confidence, as well as the entities and the entity confidences. Further, MS using features extracted from the user message performed even better than only using the module response. Using the BERT model and no additional features from the module response is doing well enough to consider dropping the integration of the module responses into MS. Since it is both costly and time-consuming to train a BERT model and to obtain the training data for the MS from the modules beforehand, the results indicate that doing both does not improve the result enough to make the costs worthwhile. Although the much simpler normalization methods also exceed the baseline, they are significantly worse than the MLP scenarios. Further, the normalization methods still require a certain amount of pre-training to calculate the minimum, maximum, and average of target values.

In addition, the performance of the MDS with BERT exceeds the performance of the singular-DSs. This result can be observed with each NLU framework. After comparing multiple DSs trained on the same data and using varying NLU frameworks, we found that the best-performing system is an MDS composed of three Rasa NLU modules. It also works well

to combine modules that use several different NLU platforms.

Possible future work is testing a varying amount of modules, testing other datasets, experimenting with different dataset splits, and exploring the use of other NLU models that may be faster or more efficient. Other dataset splits should also be evaluated to find the right balance of overlap between the MS and module training sets.

References

[1] NEHRING, J. and A. AHMED: *Normalisierungsmethoden für intent erkennung modularer dialogsysteme*. In *Proc. ESSV 2021*, pp. 264–271. TUDpress, 2021.

[2] PLANELLS, J., L. HURTADO OLIVER, E. SEGARRA, and E. SANCHIS: *A multi-domain dialog system to integrate heterogeneous spoken dialog systems*. In *Proc. Interspeech 2013*, pp. 1891–1895. 2013. doi:10.21437/Interspeech.2013-459.

[3] D'HARO, L. F., S. KIM, K. H. YEO, R. JIANG, A. I. NICULESCU, R. E. BANCHS, and H. LI: *Clara: a multifunctional virtual agent for conference support and touristic information*. In *Natural language dialog systems and intelligent assistants*, pp. 233–239. Springer, 2015.

[4] STUCKI, T. and S. D'ONOFRIO: *Architekturmuster für Multi-Chatbot-Landschaften: Bot-Orchestrator und Alternativen*. *HMD Praxis der Wirtschaftsinformatik*, 57(6), pp. 1187–1205, 2020.

[5] BANCHS, R. E., R. JIANG, S. KIM, A. NISWAR, and K. H. YEO: *AIDA: Artificial intelligent dialogue agent*. In *Proc. SIGDIAL 2013*, pp. 145–147. ACL, Metz, France, 2013.

[6] SUBRAMANIAM, S., P. AGGARWAL, G. B. DASGUPTA, and A. PARADKAR: *Cobots - A cognitive multi-bot conversational framework for technical support*. In *Proc. Int. Conf. on Autonomous Agents and MultiAgent Systems 2018*, pp. 597–604. 2018.

[7] DEVLIN, J., M.-W. CHANG, K. LEE, and K. TOUTANOVA: *BERT: Pre-training of deep bidirectional transformers for language understanding*. In *Proc. NAACL 2019*, pp. 4171–4186. ACL, 2019. doi:10.18653/v1/N19-1423.

[8] LIU, X., A. ESHGHI, P. SWIETOJANSKI, and V. RIESER: *Benchmarking natural language understanding services for building conversational agents*. In *Proc. IWSDS 2019*, pp. 165–183. Springer, 2021. doi:10.1007/978-981-15-9323-9_15.

[9] *Rasa open source version: 3.x docs*. `https://rasa.com/docs/rasa/`, Accessed: 2021-11-28.

[10] *Google dialogflow documentation*. `https://cloud.google.com/dialogflow/docs/`, Accessed: 2021-11-28.

[11] *Ibm cloud api docs / watson assistant v1*. `https://cloud.ibm.com/apidocs/assistant-v1?code=python`, Accessed: 2021-11-28.

[12] BUNK, T., D. VARSHNEYA, V. VLASOV, and A. NICHOL: *DIET: Lightweight Language Understanding for Dialogue Systems*. 2020. doi:10.48550/arXiv.2004.09936.