Improving NLU Training over Linked Data with Placeholder Concepts

Tobias Schmitt[1], Cedric Kulbach[2], and York Sure-Vetter[1,2](✉)

[1] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
tobias.s.schmitt@web.de, york.sure-vetter@kit.edu

[2] FZI Research Center for Information Technology, Karlsruhe, Germany
kulbach@fzi.de

Abstract. Conversational systems, also known as dialogue systems, have become increasingly popular. They can perform a variety of tasks e.g. in B2C areas such as sales and customer services. A significant amount of research has already been conducted on improving the underlying algorithms of the natural language understanding (NLU) component of dialogue systems. This paper presents an approach to generate training datasets for the NLU component from Linked Data resources. We analyze how differently designed training datasets can impact the performance of the NLU component. Whereby, the training datasets differ mainly by varying values for the injection into fixed sentence patterns. As a core contribution, we introduce and evaluate the performance of different placeholder concepts. Our results show that a trained model with placeholder concepts is capable of handling dynamic Linked Data without retraining the NLU component. Thus, our approach also contributes to the robustness of the NLU component.

Keywords: Natural Language Understanding · Named Entity Recognition · Chatbots · Linked Data

1 Introduction

Modern conversational systems, also called dialogue systems (DS), are gaining access into peoples day-to-day lives and are offering an increasing number of services, especially known to the public audience in the form of chatbots. The standard DS consists of three components: the Natural Language Understanding (NLU) component, which identifies the meaning behind the incoming message and extracts relevant parts called entities, the Dialogue Manager (DM), which determines the corresponding action based on the output from the NLU, and the Natural Language Generator (NLG), which generates the response that is transmitted to the user [11].

In DS the NLU component mostly uses standard concepts from *Natural Language Processing (NLP)* tasks. It mainly consists of an *intent classifier* and a *named entity recognition (NER)* component. Both components make use of machine learning technologies, which mostly need to be trained supervised (s. Sect. 1.1). More and more data is published as Linked Data, which forms a suitable knowledge base for NLP tasks. In the context of chatbots a key challenge is developing intuitive ways to access this data to train an NLU pipeline and to generate answers for NLG purposes. Using the same knowledge base for NLU and NLG provides a self-sufficient system. An NLU component identifies the intents and entities which the NLG component requires for generating the response. However, the challenge becomes apparent when the knowledge base changes and the already trained NLU model deteriorates in the detection of intents and entities. Training on more general training data could avoid computational expensive retraining and make the NLU component more robust against changes in the knowledge base and unclear requests. In this context, we define the robustness of an NLU through the metrics of the NLU on not yet seen entity values. As a more general approach to create appropriate training data for the NLU we propose the placeholder concept where placeholder values are used as entity values instead of real ones taken from a related knowledge base. These values are then filled into predefined sentence patterns to generate the final dataset for training the NLU components. As a key result, we show which type of entity values (placeholder or database values) work best for training a NER algorithm or an intent classifier.

In a first step, we present the typical process that can be used when designing an NLU in the chatbot context. After a motivating example in Sect. 1.2 the procedure for the construction of training data for an NLU pipeline (Sect. 2) is shown. To compare the performance of the two conceptual approaches to create the NLU training dataset, we created a set of experiments that are described in Sect. 3. After evaluating the performance results of the conducted experiments in Sect. 4, we bring the paper into the context of related work (Sect. 5). An outlook is given in Sect. 6.

1.1 NLU in Chatbot Context

In current DS architectures the NLU component is the most critical component to the success of chatbots or question answering (Q&A) [6]. It aims to identify the meaning behind the user's input and extracts all the custom entity values in the incoming utterance [23]. Identifying the intent of the interlocutor is a classification problem that can be solved using supervised machine learning techniques. Available classifiers include Support Vector Machines (SVM) [3,13], deep neural networks [18,19] and embedding models [24]. The classifier is trained to predict to which of the learned intent classes the incoming utterance belongs to and to assign this label to the utterance so that it can be used by the next component [20]. All the intents that the system shall be able to match to user inputs have to be included in the training dataset. If the user input does not correspond to any of the learned intent labels, the model will still match it to one of them [16]. In closed domain DS this behavior leads to a chatbot that will answer every question, which must be taken into account during the creation process of the training dataset.

The second task of the NLU is to extract custom entities using sequence-labeling techniques. Conditional Random Fields (CRF) and Recurrent Neural Network (RNN) are most commonly used to label each unit in an utterance to determine the words that correspond to each of the learned entity types [10]. This is achieved by extracting features from the surrounding words (context) so that the system can predict not only the entity values present in the training data but also new values that users might use in their messages. Both components form an NLU. Examples of available NLUs include Microsoft's LUIS,[1] IBM's Watson[2] and RASA's NLU[3] [4].

```
"text"        :   "Where is the lecture  Web Science taking place?",
"intent"      :   " location_of_lecture ",
" entities "  :   [
{
"start"  :   21,
"end"    :   32,
"value"  :   "Web Science",
"entity":    "lecture"
}]
```

Listing 1.1. Example of a labelled utterance used to train the intent classifier and entity extractor of the NLU.

Example of a data point that can be either used for training or testing the NLU is presented in Listing 1.1 (s. Sect. 2). To generate the training data like the one shown, we create a set of utterances related to each intent and integrate the entity values from either a knowledge base or placeholder values at the designated places. This approach not only provides a semi-automated way for generating training datasets, such as the one depicted in Listing 1.1, it further provides the first step towards an integration of Semantic Question Answering (SQA) [21] tasks into chatbots. By generating training data as described in this work, the aim is to analyze how well a system can be trained if little or no information is available about the entity values that users might use in their utterances. In summary, we provide contributions to the following questions:

RQ 1 Which type of entity values work best for training the entity recognition algorithm?
RQ 2 Which type of entity values work best for training different intent classifiers?
RQ 3 How can linked data improve NLU performances?

1.2 Motivating Example

We describe an example that motivates our approach and experiments. The handbook for the study program *Industrial Engineering and Management* at

[1] https://www.luis.ai/home, accessed on 11.12.2018.
[2] https://console.bluemix.net/developer/Watson/documentation, accessed on 11.12.2018.
[3] https://rasa.com, accessed on 11.12.2018.

KIT is publicly available as a *.pdf* version. In order to make this information accessible by a computer program, such as a dialogue system, the relevant data were extracted and transformed to RDF. The domain of a DS trained with the RDF triplestore is defined by *lectures*, *lecturers*, *location* and *semesters*, where each *person*(lecturer) can lecture a *lecture* from a specific *module* in a given *room/building* (*location_of_lecture*) at a given *date/semester*. To answer a question like *'Where is the lecture Web Science taking place'* (*Q1*) the NLU needs to detect the intent *location_of_lecture* and the entity *lecture* with the value *'Web Science'*. Remarking that the question for a *location* is related to the entities found in the question (in *Q1 lecture*). This problem is addressed by *relation linking* (RL) [7]. Before the RL problem can be resolved, however, it must first be ensured that the correct intents and entities are found. To train the NLU a set of utterances for each intent is defined (s. Listing 1.1). In a closed domain DS the entries from the knowledge base can be used to generate utterances by replacing the entities (e.g. *"Web Science"* in Listing 1.1) from the utterances with the entries from the knowledge base. For example, with the help of the sentence pattern *'Where is the lecture lecture taking place?'* and the knowledge base, data points can be generated automatically from the *lectures* property. Whereby, *lecture* is a placeholder for the *lectures* entries from the triple store or other values. We call this concept the domain or placeholder concept (s. Sect. 2.1). The results are multiple data points with the same structure, but different entities. Taking into account that the entity values (i.e. for the entity *lecture*) can change over time, the NLU has the task of identifying intents and entities that did not exist before. We address this problem by providing a robust NLU (definition in Sect. 1) from the beginning.

2 Construction of Training Data

In the first part the general design approach is described before presenting a holistic approach that can be used to systematically create a DS and its matching training dataset.

2.1 Training Data Design Approaches

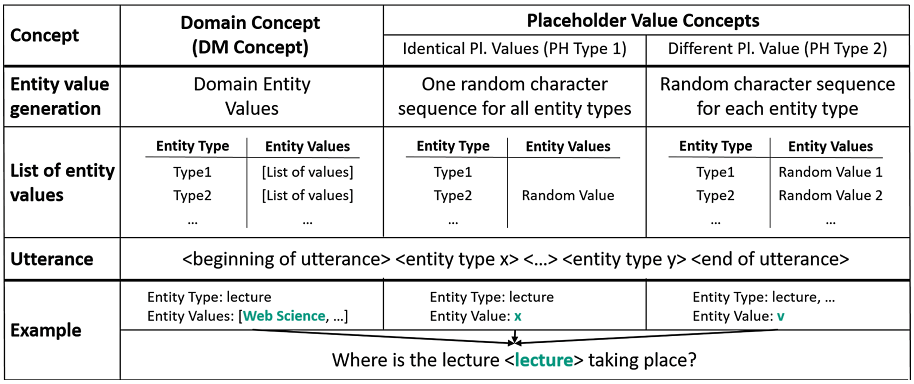
In this work, we aim to optimize the performance of the two tasks of the NLU (Intent classification and entity recognition) by optimizing the dataset that is used to train the system. Therefore we created two different design approaches that can be applied to create the training dataset for a domain-specific NLU. In this work, we focus on how the performance of the trained NLU is impacted if different types of entity values are used to create the training dataset.

Before going into the specifics of the approaches it has to be noticed that the utterance patterns have to be created. This is necessary so that the entity values can later be filled in automatically. For each of the defined intents, a set of utterances have to be created, where each one contains one or more entity values that the system shall learn to detect. Because we want to be able to insert different types of values automatically, an empty slot of matching type is inserted at the position where an entity value shall be inserted during the creation of the training data. Looking at utterance from the motivating example, we replaced the value of type lecture (*Web Science*) by an empty slot of type lecture (*'Where is the lecture* {*lecture*} *taking place'*). Now we are able to insert different types of entity values into the utterances without having to change the utterances manually. Both approaches use the same utterances for each intent but are filled with different kinds of entity values.

The two approaches described in the following are called the **Domain Concept** and **Placeholder Concept**. As the names suggest, we used entity values from a related knowledge base to create the training dataset within the database concept and placeholder values in the placeholder concept to create the dataset for training the NLU.

Looking at the domain concept, it can be seen that the related knowledge database is queried for each of the defined entity types with the goal to extract all available values and store them into a list. These values are then used to fill the empty slots in the utterances, with respect to the entity type restriction. Table 1 shows how both concepts work and further depicts an example for each of them. The example shows how one of the entity values of type *lecture* is used to fill the empty slot of matching type in the example utterance. This utterance together with the appropriate labels can then be used to train the component of the NLU.

Table 1. Conceptual approaches used to create the dataset for training and testing the NLU of the task-oriented component-based dialogue system.



Concept	Domain Concept (DM Concept)	Placeholder Value Concepts	
		Identical Pl. Values (PH Type 1)	Different Pl. Value (PH Type 2)
Entity value generation	Domain Entity Values	One random character sequence for all entity types	Random character sequence for each entity type
List of entity values	**Entity Type** / **Entity Values** Type1 / [List of values] Type2 / [List of values] … / …	**Entity Type** / **Entity Values** Type1 / Type2 / Random Value … /	**Entity Type** / **Entity Values** Type1 / Random Value 1 Type2 / Random Value 2 … / …
Utterance	<beginning of utterance> <entity type x> <…> <entity type y> <end of utterance>		
Example	Entity Type: lecture Entity Values: [Web Science, …]	Entity Type: lecture Entity Value: x	Entity Type: lecture, … Entity Value: v
	Where is the lecture <lecture> taking place?		

The second approach is called **Placeholder Concept** and refers to the fact that instead of real values, taken from some knowledge database, placeholder values are inserted into the utterances to create the dataset for training the NLU. In general, the placeholder values are values which consist of one or multiple random words of varying length. The random words used in this work have e.g. been created by randomly selecting one or multiple letters from the English alphabet. Within this concept, we followed two different ways of creating the dataset. The first one called **Identical Placeholder Values Concept (PH Type 1)**. As the name suggests in this approach only one random value is created and used to fill all the empty slots in the utterances regardless of the entity type. In the example shown in Table 1 the letter x was selected to fill all of the empty slots. The second approach is called **Different Placeholder Value Concept (PH Type 2)**. In this approach different random values are used to fill the different types of empty slots. For each of the defined entity types, one unique random value is created and used to fill the corresponding empty slots.

With the experiments described in Sect. 3 we aim to determine which design concept is best for training a domain-specific NLU. Based on the design specification of the concepts it can be assumed that if a dataset is created that contains all available entity values the results are likely to be highest.

In the next part of the section, we introduce a holistic approach that can be used to create the dataset matching the requirements of a domain-specific NLU of a task-oriented DS.

2.2 Training Data Creation Process

In this subsection, we describe an approach that can be used to design the NLU of a task-oriented DS and to create a dataset matching the requirements. The complete approach is depicted in Fig. 1 and is based on the procedure described by Grötz [8].

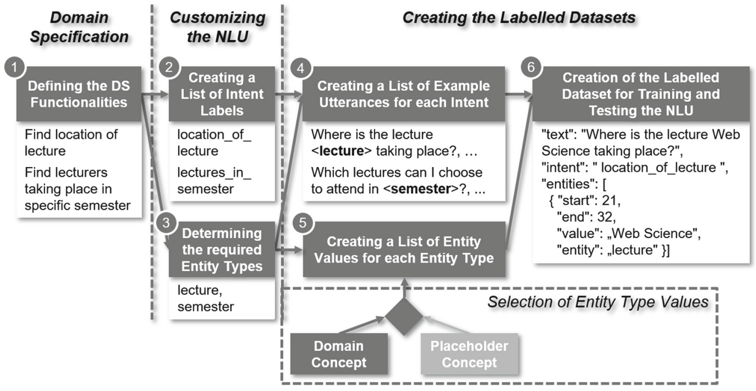

Domain Specification
Customizing the NLU
Creating the Labelled Datasets
1 Defining the DS Functionalities
Find location of lecture
Find lecturers taking place in specific semester
2 Creating a List of Intent Labels
location_of_ lecture
lectures_in_ semester
3 Determining the required Entity Types
lecture, semester
4 Creating a List of Example Utterances for each Intent
Where is the lecture <lecture> taking place?, ...
Which lectures can I choose to attend in <semester>?, ...
5 Creating a List of Entity Values for each Entity Type
6 Creation of the Labelled Dataset for Training and Testing the NLU
"text": "Where is the lecture Web Science taking place?", "intent": " location_of_lecture ", "entities": [{ "start": 21, "end": 32, "value": „Web Science", "entity": „lecture" }]
Selection of Entity Type Values
Domain Concept
Placeholder Concept

Fig. 1. Process for designing a customized NLU and creating the corresponding labeled dataset to train and test the system

The process consists of six processing steps which can be categorized into three areas. The first area focuses on defining the functions/tasks of the DS. The processes in the second area are to derive a set of intent and entity type labels that the NLU needs to be able to assign to an incoming utterance. In the processes of the last area, the previously defined intents and entity types are used to create a matching dataset for training (and testing) the NLU.

The first area is called **Domain Specification** and consists of one process during which a set of functions/tasks are defined that the dialogue system shall be able to handle. According to Grötz [8], it is recommended to start with a small set of functions and to use the collected experience over time to improve them and to successively add new ones. In this work, we created the NLU of a DS which aims to support students at the KIT in acquiring information related to their study program (s. Sect. 1.2). Two of the defined functions, depicted in Fig. 1, aim to find the location where a certain lecture takes place and to identify lectures that take place in a specific semester. In our approach, the information required to answer the students' questions are stored within an RDF knowledge base. SPARQL queries are used to extract the demanded information from the incoming question.

The second area is called **Customizing the NLU** during which a list of intent labels and entity type labels has to be defined that the NLU shall be able to assign to the incoming utterances. Within the second process step called *Creating a List of Intent Labels*, one intent label is created for each of the previously defined functions. Following the example depicted in Fig. 1, one intent label is created for each of the two functions. The intent label related to the first function is called *location_of_lecture* and the one related to the second function is called *lectures_in_semester*. In the third step, the types of entity values are determined which the NLU needs to be able to extract from the incoming utterances. These values are required to perform the functions defined in the first process step. The types of entity values that the NLU has to be able to extract can e.g. be derived from the underlying SPARQL queries. This is essential since the entity values are required to perform the query in order to retrieve the demanded information from the knowledge base. In the presented example, entity values of type *lecture* or of type *semester* are required to execute the underlying query.

After having defined the required parameters, the process steps within the third area focus on creating an optimal dataset for training the NLU. Within the fourth step, a list of utterances is created for each of the defined intents following the procedure described in the previous section. At the positions in the utterances where an entity value of a certain type shall be inserted, an empty slot of matching type is placed. Furthermore, the utterances have to match the language usage of the target users (e.g. formal or informal) [8]. In Fig. 1 one utterance for each of the two intents is depicted where each includes one of the two defined entity types. In the sixth step, a list of entity values for each type is created that is then used to fill the empty slots in the utterances in order to create the final dataset. As explained in the previous section there are two approaches that can be applied for replacing the empty slots in the utterances. The first one is depicted in step 5.1 where a list of 'real' entity values is extracted from a related knowledge base. As described in Sect. 1.2 we created a RDF knowledge graph that contains all information related to the industrial engineering and management study program at the KIT. The second option is to use placeholder values instead of real values. One value is assigned to each of the entity types, which can be either identical or different as shown in Table 1.

In the last process step the empty slots in the utterances from step 4 are replaced using one of the lists created in step 5. At last information about the two sets of labels are added to each utterance. This includes the intent label, the entity type, the entity value and the position at which the entity values can be found in the utterance. This information is stored in one of the formats such as JSON.

3 Experiments

Based on the previously introduced approach we created a task-oriented NLU to determine which of the approaches from Subsect. 2.1 is best for training such a system. In the first part, we describe the development of the training datasets which were used to train the NLU, which we then evaluated to compare the performance that can be achieved by following the different design approaches. The applied pipeline of the NLU is described as part of the state of the art within the context of related work (s. Sect. 5).

3.1 Creation of Domain Specific Dataset

In order to evaluate the different approaches previously described, we created several datasets to train the NLU of the DS introduced in Sect. 1.2. Following the process from Sect. 2.2 we first defined the functionality of our DS and used these to derive a set of intents and entity types for creating the NLU. Next, we created a set of utterances with empty slots for each intent and created three entity type lists with different values to fill the empty slots. In the last part of the section we describe the experimental datasets used to evaluate the design approaches from Sect. 2.1

System Specification and Creation of Utterances. Following the process described in Fig. 1 we defined 16 functions that our DS shall be able to perform. For the configuration of the NLU, we created one intent label per function, which the intent classifier shall be able to assign to incoming utterances after training. In addition, we derived the types of entity values that are required to perform the succeeding processing step, such as making a database inquiry (not realized in this work). In total, the NER component of the NLU needs to be able to recognize and extract six different types of entity values. An extract of the complete list of the intents and the corresponding entity values can be seen in Table 2. The first column shows the name of the intent and the last column the entity value type that is required for further processing.

Furthermore, the table shows how many utterances have been manually created for each intent. As described in Subsect. 2.1 we inserted empty slots at the position in the utterance where one of the entity values shall be included in the final step. In total 299 utterances were created, which were split into a train (80%) and a test (20%) set. These utterances are used by both design approaches to create the final datasets for training and testing the NLU.

Table 2. Number of training utterances created for each intent and their corresponding entity types.

Intent	Utterance			Entity type
	Train (80%)	Test (20%)	Combined	
lecturer_of_lecture	18	5	23	lecture
lectures_in_the_current_semester	8	3	11	semester, subject
semester_of_lecture	8	2	10	lecture, semester
subject_of_lecture	8	2	10	lecture, subject
modules_within_subject	16	4	20	subject
subject_affiliation_of_module	16	5	21	module
location_of_room	19	5	24	building
office_of_lecturer	16	4	20	person
...	...	...	...	...
Total	**234**	**65**	**299**	

Entity Values. As explained in Sect. 2.1 there are two options to replace the empty slots with a corresponding entity value. Following the domain concept, we extracted all values related to each of the six entity types from a related RDF file as explained in the motivating example. The values were retrieved by using one SPARQL query for each type, which was then stored into a list. One list was created for each entity type where all matching values were stored. As with the utterance, each entity list was split into a training and testing set. The combined set included all values found. Table 3 depicts the number of values found in the RDF file which relates to one of the six entity types. The second column in the table indicates how many empty slots in the utterances exist, which need to be filled in order to create the final dataset. Having extracted all possible values that the system needs to be able to recognize, the empty slots were replaced by looping through the created entity list and filling in a value of the matching type into the existing utterances. If there were more empty slots than unique entity values, some values were used more then once which were selected randomly. If there were more unique entity values than empty slots, some utterances were used more than once. In that case, we randomly selected a matching number of utterances from the list of utterances that only have an empty slot of that specific type. Those were then used to fill in the remaining utterances to finish the replacement process.

To create the utterances following the placeholder concept we created two sets of placeholder values. The type 1 consists of one value which is used to replace all empty slots in the utterances independent of the type. The type 2 list contains one unique value for each entity type, which is then used to replace the empty slots of matching type. The values we used to create our datasets are depicted in the last two columns of Table 3. In the last step, the previously created lists with entity value(s) can now be used to create the datasets for training and testing the different NLUs.

Table 3. Placeholder values and unique domain values used to replace the empty slots.

Entity type	Empty slots	Domain values			Placeholder values	
		Train (80%)	Test (20%)	Combined	Type 1	Type 2
lecture	137	69	18	87	x	v
person	46	36	10	46	x	p
semester	21	11	3	14	x	s
subject	41	4	1	5	x	f
module	61	44	11	55	x	m
building	24	19	5	24	x	g
Total	**330**	**183**	**48**	**231**	**1**	**6**

Experimental Datasets. In order to answer the research questions introduced in Sect. 1.1 we conducted a total of five experiments. Thereby we want to determine which type of entity values are best suited to create the training data and how the trained NLU performs of different test datasets.

The datasets for training the NLU have been created by filling the designated training utterances with some related entity values, as described in Subsect. 2.2. Table 4 contains an overview of the experiments and the datasets used to evaluate the performance of the NLU. The first two experiments are related to the domain concept. In the first experiment (EX 1) the training dataset contains a subset of the entity values that have been extracted from the available knowledge base. Thereby we want to analyze how well the NLU can perform the two tasks if the test set contains unknown utterances and unknown values taken from the knowledge base. In addition, we want to determine how well the NLU performs if the utterances are filled with entity values taken from another domain, in this case, the DBpedia knowledge graph. To determine how well the NLU performs if all domain related entity values are used for training, we conducted the second experiment (EX 2).

The third and fourth experiments (EX 3 and 4) have been created to evaluate how the performance of the NLU changes if placeholder values are used to train the system. In EX 3 the train utterances have been filled with the PH Type 1 values and in EX 4 they have been filled with PH Type 2 values.

In the last experiment, we filled the train utterances with the values extracted from the DBpedia and merged this one with the EX 1 dataset. Thereby we aim to determine if the performance can be approved when the dataset is enriched with values taken from another domain. Because we were not able to extract entity values form all of the six types, we only used the utterances that contain an entity type of at least one of the following types: lecture, building or person.

The datasets used to test the performance has been created by using either the test set of the domain values or the test set of the DBpedia values to fill the test utterances. Because the DBpedia set does not contain values of type semester, subject and module the domain values of those types have been used to create the *Test DBpedia* dataset. For determining and evaluating the performance of the different conceptual approaches we calculated the precision, recall and F1-score of the trained NLUs. No cross-validation has been applied to evaluate the performance.

Table 4. Utterances and entity values used to create the experimental datasets.

Training datasets	
EX 1	Train Utterances + Test Domain Entity Values
EX 2	Train Utterances + All Domain Entity Values
EX 3	Train Utterances + PH Type 1 Entity Values
EX 4	Train Utterances + PH Type 2 Entity Values
EX 5	EX1 extended by
	Train Utterances + Train DBpedia Entity Values
Testing datasets	
Domain test	Test Utterances + Test Domain Entity Values
DBpedia test	Test Utterances + Test DBpedia Entity Values

4 Evaluation Results

In this chapter, the results of the different experiments are evaluated. Table 5 provides an overview of the performance values that have been used to measure the performance of the NER and the intent classifier of the NLU. In the first part of the section, we analyze the results of the NER and intent classifier before giving a recommendation about which approach to use for training the two components of the NLU.

4.1 Performance NER

The first part of Table 5 shows the results when using the Domain Test dataset for evaluating the performance of the differently trained NLUs and the second part shows the results when using the DBpedia test dataset for testing. The first part of the table clearly shows that the datasets related to EX 1, 2 and 5 lead to the best NER performances. From those, it can be derived that using more unique entity values lead to better results. If all potential entity values that an NLU shall be able to extract are known in advance it is best to use them all for training. Enlarging the training dataset with utterances that are filled with values from another domain does not lead to better results. When using the DBpedia test dataset for evaluating the results clearly show that the F1-score of EX 5 is highest and therefore most suited for training. The results related to EX 1 and 2 are in this case far lower. In this case, the discrepancy between EX 1 and 2 and EX 5 is between 11.7 and 15.6% points. In the previous test, the results were much closer with a discrepancy between 3.2 and 6.1% points. In both cases training the NER with placeholder values lead to the lowest results. Although using PH type 1 values lead so slightly higher results the performance is still much lower than that of the other approaches. Due to the low results which are more than 50% lower, compared to the other approaches, they are not suited for training the NER component of the NLU.

Based on these results we recommend to use the approach related to EX 1 or 2 for training the NER component if the NLU shall be optimized for a certain domain. If instead, the NLU shall perform well on several domains we recommend to merge the datasets following the approach described in EX 5 to maximize the NER's performance.

Table 5. Performance results of the conducted experiments.

	NER			Embedding classifier		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score
	Domain test					
EX 1	0.995	0.895	0.940	0.8626	0.8458	0.8452
EX 2	1.000	0.967	0.979	0.8497	0.8151	0.8101
EX 3	0.893	0.286	0.428	0.8663	0.8308	0.8288
EX 4	0.928	0.220	0.351	0.8311	0.8151	0.8029
EX 5	0.991	0.856	0.918	0.8657	0.8308	0.8269
	DBpedia test					
EX 1	0.878	0.770	0.812	0.8448	0.8151	0.8103
EX 2	1.000	0.806	0.851	0.8522	0.8151	0.8170
EX 3	0.835	0.312	0.451	0.8907	0.8769	0.8722
EX 4	0.848	0.220	0.351	0.8317	0.8151	0.8024
EX 5	0.992	0.950	0.968	0.8479	0.8308	0.8185

4.2 Performance Intent Classifier

The performance results of the different experiments when using the domain test for the evaluation show, that overall all different approaches perform well with F1-scores greater than 80%. By comparing EX 1 and 2 it can be noticed that when more unique entity values are used for training the performance of the classifier decreases. We assume this to be the case because several entity values are used multiple times within different utterances that belong to different intents. Because the classifier learns which words relate to which intent, we assume that this approach causes a distortion of the vector space which results in lower performance results. Therefore EX 1 performs better than EX 2 and is overall the best approach for training a domain-specific intent classifier. It has to be noticed that when using the other dataset for testing, the results of EX 2 are slightly higher than that of EX 1, which could indicate that there are other factors that have a significant impact of the performance.

Looking at the performance of the experiments that applied the placeholder concept, the results show that this approach is highly applicable for training a high-performance intent classifier. Especially when using PH type 1 values the discrepancy between EX 1 and EX 3 is only 1.64% points. Furthermore, it is possible to train a much more robust classifier using PH type 1 values. As can be seen from the results where the DBpedia test dataset has been used for testing the trained classifiers, the one which has been trained using PH type 1 values performs better than all the other trained classifiers. Therefore it can be said that this approach is better suited when we want to train a classifier that can perform well in several domains. This approach increases the robustness of the NLU which performs best when entity values form the DBpedia domain are used in the test utterances.

Based on the results at hand, we recommend applying the domain approach following the EX 1 construction when training an intent classifier that shall only perform well in a certain domain. When aiming towards training a more robust and open domain intent classifier we recommend to used PH type 1 values to construct the training dataset. Although the performance in some domains might be lower, compared to using domain-specific values for training, the performance overall domains will be higher.

In order to optimize the performance of the placeholder concept, differently designed placeholder values can be tested. We created values of different word length and also created values which consisted of two or more random words. Although we were not able to increase the performance, it might be possible to find values that can be used to increase the performance.

5 Related Work

Our contribution in training an NLU targets the research field of chatbots, as well as SQA. While most chatbot frameworks (IBM Watson, Microsoft Bot Service) are based on deep learning technologies for Intent and Entity Recognition as one NLU component, most SQA systems use static n-gram strategy [22] or Entity Linking Tools [5]. The *DBpedia Bot* [1] is one example for a rule-based, static SQA realization. This static approach of Q&A over knowledge graphs (KGs) has the disadvantage of only being able to react conditionally to sentence conversions. The idea of the *Frankenstein* Framework [22] is to link these static approaches by generalizing SQA into 3 steps (Named Entity Recognition and Disambiguation, Relation Linking and Query Building). Considering the SQA task our work addresses the NER and NED component, whereby an intent classification task is also taken into account and could improve the query building component. In general, it is possible to train multiple closed domain systems, which would make the NLU applicable in multiple domains [17]. For the present study, the closed domain knowledge is stored in a database and used to create the training data for the NLU. The database contains all entity values that users might use in their utterances.

Bapat et al. [2] already presented an end-to-end pipeline for simplifying the NLU training process, where the first sentences are defined and extended for the following training. While the extension of the training dataset is skipped and only classified into 5 categories of possible extension methods, our approach mainly targets the class of generating big pools of parameter values. The following NLU training was conducted by using the state-of-the-art and open-source software of the Berlin-based company Rasa [16]. The extraction of entities and the classification of intents can be regarded as two separate tasks that can be achieved by two different pipelines that are merged into one coherent NLU pipeline. The intent classification pipeline uses the tokenized utterances created by the spaCy model [9]. During training, each token and intent label is represented as a feature vector, except for digits, all of which are assigned to the same feature vector. The embeddings model is based on the StarSpace model developed by Facebook [24]. During training, the embeddings classifier learns its own embeddings for each of the words in the training dataset, thereby taking into account domain-specific uses of words [15]. The created feature vectors are enriched by an additional three dimensions using the *intent_featurizer_ngrams*. Again, the three most common n-grams in the training data are determined and the three added dimensions are used to indicate whether a given token includes one of these n-grams. The NER pipeline tokenizes the incoming utterance into its elements by also using the spaCy model and automatically assigns POS tags to each word in the utterance. Since only CRF [12] is supported as a NER algorithm in Rasa, it was applied for the experiments. Placeholder concepts could be considered as a way to increase the number of training examples and thus improve the NLU performance.

6 Conclusion and Outlook

Three different design approaches for creating labeled training datasets were developed and integrated into a holistic development process to design the NLU of a task-oriented DS and to create a corresponding dataset for training the component. While the experiments for RQ 1 clearly show that using more unique Entities improves the performance of the NER component, a placeholder concept only affects the intent classifier (RQ 2) slightly. In terms of robustness, the evaluation of EX 5 on different test datasets shows, that the performance of the NER component can be increased by including train datasets from different domains. With placeholder values from different domains, we show how Linked Data can help to increase (RQ 3) not only NLU robustness but also overall performance in open domains.

A challenge that appears with RQ 1–2 is the generalizability of the proposed concepts. We mainly address small, domain-specific databases, whereby an evaluation on larger datasets with multiples domains could lead to synergy effects within the creation of the NLU training dataset. For further research, the NLU component could be integrated into the Frankenstein framework and evaluated on the SQA challenge dataset [14].

References

1. Athreya, R.G., Ngomo, A.N., Usbeck, R.: Enhancing community interactions with data-driven chatbots-the DBpedia chatbot (2018). https://doi.org/10.1145/3184558.3186964

2. Bapat, R., Kucherbaev, P., Bozzon, A.: Effective crowdsourced generation of training data for chatbots natural language understanding (2018). https://doi.org/10.1007/978-3-319-91662-0_8
3. Bocklisch, T., Faulkner, J., Pawlowski, N., Nichol, A.: Rasa: open source language understanding and dialogue management. CoRR abs/1712.05181 (2017)
4. Braun, D., Hernandez-Mendez, A., Matthes, F., Langen, M.: Evaluating natural language understanding services for conversational question answering systems (2017)
5. Buscaldi, D., Rosso, P., Soriano, J.M.G., Sanchis, E.: Answering questions with an n-gram based passage retrieval engine. J. Intell. Inf. Syst. **34**(2), 113–134 (2010). https://doi.org/10.1007/s10844-009-0082-y
6. Diefenbach, D., Lopez, V., Singh, K., Maret, P.: Core techniques of question answering systems over knowledge bases: a survey. Knowl. Inf. Syst. **55**, 529–569 (2018)
7. Dubey, M., Banerjee, D., Chaudhuri, D., Lehmann, J.: EARL: joint entity and relation linking for question answering over knowledge graphs. CoRR abs/1801.03825 (2018)
8. Grötz, R.: Sprich mit mir! iX - Magazin für Professionelle Informationstechnik **6**, 50 (2018). https://www.heise.de/-4054854
9. Honnibal, M., Montani, I.: spaCy 2: natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing (2017)
10. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. CoRR abs/1508.01991 (2015)
11. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall Series in Artificial Intelligence, 2nd edn. Prentice Hall Pearson Education International, Upper Saddle River (2009)
12. Lafferty, J., McCallum, A., Pereira, F.C.N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, p. 10 (2001)
13. de Mori, R., Bechet, F., Hakkani-Tur, D., McTear, M., Riccardi, G., Tur, G.: Spoken language understanding. IEEE Signal Process. Mag. **25**(3), 50–58 (2008). https://doi.org/10.1109/MSP.2008.918413
14. Napolitano, G., Usbeck, R., Ngomo, A.N.: The scalable question answering over linked data (SQA) challenge 2018 (2018). https://doi.org/10.1007/978-3-030-00072-1_6
15. Nichol, A.: Supervised word vectors from scratch in Rasa NLU
16. Petraityte, J.: Deprecating the state machine: building conversational AI with Rasa stack (PyData 2018)
17. Ramesh, K., Ravishankaran, S., Joshi, A., Chandrasekaran, K.: A survey of design techniques for conversational agents. In: Kaushik, S., Gupta, D., Kharb, L., Chahal, D. (eds.) ICICCT 2017. CCIS, vol. 750, pp. 336–350. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-6544-6_31
18. Ruder, S.: An overview of multi-task learning in deep neural networks. CoRR abs/1706.05098 (2017)
19. Sarikaya, R., Hinton, G.E., Deoras, A.: Application of deep belief networks for natural language understanding (2014). https://doi.org/10.1109/TASLP.2014.2303296
20. Serban, I.V., Lowe, R., Henderson, P., Charlin, L., Pineau, J.: A survey of available corpora for building data-driven dialogue systems (2018)
21. Shen, D., Lapata, M.: Using semantic roles to improve question answering (2007)
22. Singh, K., et al.: Why Reinvent the Wheel: Let's Build Question Answering Systems Together, Lyon, France (2018). https://doi.org/10.1145/3178876.3186023

23. Wang, X., Yuan, C.: Recent advances on human-computer dialogue. CAAI Trans. Intell. Technol. **1**(4), 303–312 (2016). https://doi.org/10.1016/j.trit.2016.12.004
24. Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., Weston, J.: StarSpace: Embed all the things! CoRR abs/1709.03856 (2017)