# The Impact of Romanian Diacritics on Intent Detection and Slot Filling

Anda Diana Stoica
*Computer Science Department*
*Technical University of Cluj-Napoca*
Cluj-Napoca, Romania
andadiana.stoica@gmail.com

Andrei-Cristian Rad
*Computer Science Department*
*Technical University of Cluj-Napoca*
Cluj-Napoca, Romania
radandreicristian@gmail.com

Ioan Horia Muntean
*Computer Science Department*
*Technical University of Cluj-Napoca*
Cluj-Napoca, Romania
ihmmuntean@gmail.com

George Daian
*Computer Science Department*
*Technical University of Cluj-Napoca*
Cluj-Napoca, Romania
daian_george_01@yahoo.com

Camelia Lemnaru
*Computer Science Department*
*Technical University of Cluj-Napoca*
Cluj-Napoca, Romania
Camelia.Lemnaru@cs.utcluj.ro

Rodica Potolea
*Computer Science Department*
*Technical University of Cluj-Napoca*
Cluj-Napoca, Romania
Rodica.Potolea@cs.utcluj.ro

Mihaela Dinsoreanu
*Computer Science Department*
*Technical University of Cluj-Napoca*
Cluj-Napoca, Romania
Mihaela.Dinsoreanu@cs.utcluj.ro

***Abstract*—This paper aims to provide an empirical analysis of the effect of Romanian diacritics on the performance of two well known frameworks for intent detection and slot filling: the *RASA* framework and *Facebook's Wit.ai* API. We use a previously defined home assistant scenario and datasets, which we modify to include diacritics. The results of several alternative processing pipelines in the two frameworks suggest that diacritics introduce a new level on learning complexity, in addition to the ones considered initially - synonymy, missing slot values and class imbalance - and they contribute to augmenting the effect of those complexities on the performance of the learning models. Moreover, the slot filling task seems to be the most affected by this aspect, with a drop in performance as high as ˜30% for *RASA* and more that 40% for *Wit.ai*.**

***Index Terms*—intent detection, slot filling, Romanian home assistant, diacritics, RASA NLU, Wit.ai**

## I. Introduction

Conversational systems can be found in many domain-specific applications nowadays, such as Apple Siri, Microsoft Cortana, and Google Assistant, which provide a relatively wide range of commands, from calendar management, making calls, sending text messages and providing answers to user queries by interfacing with a search engine. Another application context of conversational systems is that of smart home assistants such as Google Home and Amazon Alexa, which – used together with devices like smart lights and a smart thermostat – can automate an entire home. There are also humanoid robots such as Pepper, Sophia and Erica, which are equipped with the ability to converse with humans. This type of systems require more breadth and depth of understanding, as they should be able to mimic the language understanding level of humans and they are not necessarily tied to a specific domain.

All the above mentioned examples have become quite proficient in specific use-cases for English and Chinese, but there have been significantly fewer to no attempts for such applications for less spread languages, such as Romanian. In [1], we have defined a home assistant scenario for the Romanian language, and generated and labeled datasets for training models for intent detection and slot filling, which included several data and language specific complexities that we expect to encounter in such a scenario. The end goal was to develop an efficient solution for intent detection and slot filling for the home assistant scenario in Romanian, by initially benchmarking the most prominent existing solutions for this problem. In this paper we further extend our benchmark study, to analyze the impact of Romanian diacritics on the performance of existing systems, namely the *Rasa NLU* module from the *RASA* framework [2] and the *Wit.ai* API provided by *Facebook* [3].

We found that – for both frameworks – diacritics introduce a new level on learning complexity, in addition to the ones we analyzed in [1]. Moreover, the presence of diacritics seems to amplify other sources of errors, especially the incapacity of word embeddings to separate between antonym verbs (which we already identified in [1]). Also, slot filling seems to be significantly affected by the usage of diacritics, since new types of errors have emerged during the current analysis.

The rest of the paper is organized as follows: Section II discusses several existing tools for intent detection and slot filling, and Section III introduces the data and the associated

978-1-7281-7166-1/20/$31.00 

challenges. In Section IV we present the main processing pipelines employed and discuss the results obtained, and we close with several concluding remarks and discussion on our current work.

## II. Related Work

Due to the recent surge in conversational AI systems, there are many existing tools available that can be used when developing Natural Language Understanding applications. We began our research by investigating which are the most popular NLU tools and platforms available on the market, taking into consideration the ones most suitable for our Romanian home assistant use case. We found that these tools can be classified into two main categories: closed-source and open-source solutions.

### A. Closed source

In this category we can find those solutions that offer NLU as a software service. When employing such solutions, the developer of the conversational system interacts with the NLU engine via an online user interface or through an API. Examples of such services are Facebook's *Wit.ai* [3], Google's *DialogFlow* [4], Microsoft's *Luis.ai* [5] and IBM's *Watson* [6]. At the time of writing, *Wit.ai* is the only platform out of this list that offers support for Romanian.

The fact that the developer interacts with the conversational models through a well-defined interface makes these platforms easy to use, but also comes with the big disadvantage of not having the freedom to customize the model to a specific use case. Additionally, there are limits imposed by the current lack of support for certain languages.

### B. Open source

This type of solution offers more flexibility in terms of language support and choice of algorithms. The most popular tools from this category are *RASA* [7] and *Snips* [8]. These tools have the advantage that they can run offline, on-device, so there is no dependency to an active Internet connection. Although the latest releases at the time of writing do not support Romanian out of the box, both of these tools can be extended to include new languages.

For our experiments, we chose the *Rasa NLU* module from the *RASA* framework due to the fact that it has comparable performance with that of *Snips* for English [9] and also because its documentation provides a more comprehensive guide for how to add new languages to the tool (via its underlying use of the NLP library *SpaCy* [10]).

## III. Data and Challenges

For our home assistant use case, we defined a set of representative intents belonging to three different areas (intent super-classes): light, temperature and media [1]. A total of 13 intents were defined (see Table I for more details - we provide the intent names in English for reference).

Next, we identified a set of potential challenges that are likely to appear in such an application context, which we enumerate below:

TABLE I
Intents defined for the home assistant scenario

| Intent Name (Romanian) | Intent Name (English) | Slots | Category | Abbrev. |
|---|---|---|---|---|
| AprindeLumina | TurnOnLight | Room | Light | L_ON |
| StingeLumina | TurnOffLight | Room | Light | L_OFF |
| ScadeIntensitate Lumina | DecreaseIntensity Light | Room Level | Light | L_LO |
| CresteIntensitate Lumina | IncreaseIntensity Light | Room Level | Light | L_HI |
| CresteTemp | IncreaseTemp | Room Level | Temp | T_HI |
| ScadeTemp | DecreaseTemp | Room Level | Temp | T_LO |
| SeteazaTemp | SetTemp | Room Level | Temp | T_SET |
| PuneMuzica | PlayMusic | Artist | Media | M_ON |
| OpresteMuzica | StopMusic | – | Media | M_OFF |
| SchimbaVolum | ChangeVolume | Level | Media | M_VOL |
| PornesteTV | StartTV | Channel | Media | TV_ON |
| OpresteTV | StopTV | – | Media | TV_OFF |
| SchimbaCanal | ChangeChannel | Channel | Media | TV_CH |

1) Synonymy - considering that the same meaning of a sentence can be conveyed using various language constructs, we want to evaluate how well the tools deal with synonymy.
2) Missing slot values - this asks the question of how the model deals with slot values that haven't been encountered before in the training set, or even with sentences that are expected to have some type of slots but are missing them entirely (e.g. "Aprinde, te rog, lumina." – en. "Turn the lights on, please." - no room specified). From the intent classification and slot filling point of view, we wanted to investigate whether the lack of slot information, or the appearance of new slot values affect classification accuracy.
3) Class imbalance - this problem arises when one or more intents are under-represented in the dataset compared to the others. We want to evaluate how this challenge affects the intent classification and slot labelling performance.

To match these data challenges, we generated multiple datasets, which from now on we will refer to as "scenarios". The datasets for the three complexities have been generated incrementally - i.e. the data for the third scenario contains all three challenges in it - synonymy, missing slot values and class imbalance.

*1) Baseline scenario:* This scenario represents the baseline case, without any added complexities.

*2) Scenario 1 - synonyms:* For this scenario, we focused on employing different formulations for specifying the intent (i.e. the verb identifying the intent in the sentence) in the training and testing data.

*3) Scenario 2 - missing slots:* For this scenario, we used different slot values in the train and test sets, as well as purposefully missing slots in some examples. Only eleven out of the thirteen intents have slots, and we considered that the *ChangeVolume* intent must always have the level information associated. Therefore, only ten intents were considered for this scenario.

*4) Scenario 3 - class imbalance:* For scenarios 1 and 2 we tried to keep a balanced class distribution in both training and test datasets. The third scenario makes changes specifically to this data characteristic: we under-represented each category of intents in turn, resulting in three different train/test datasets (three different sub-scenarios), which were considered independently.

More details on the format of the datasets and how they were generated can be found in [1].

## IV. Evaluation

### A. Experimental Setup

In order to evaluate how diacritics impact the tasks of intent detection and slot filling, we used two different tools: the *Rasa NLU* module from the *RASA* framework [2] and the *Wit.ai* API provided by *Facebook* [3]. We performed, in a comparative manner, a qualitative and quantitative analysis on how well the tasks are performed on two datasets, one that contains diacritics and one that does not. For the evaluation of the dataset without diacritics, we used the results presented in [1], while for the evaluation of the other dataset we introduced diacritics in the initial one and ran them through the same processing pipeline.

To estimate the average performance, we employed repeated percentage split evaluations, by sampling 90% of the data for training and 10% for testing 5 times, and reporting the average F1-Scores for the resulting models.

*1) Rasa NLU:* Rasa NLU is a tool from the RASA Framework designed for understanding user's intention and extracting the afferent parameters, in the context of conversational systems. As opposed to other frameworks such as Wit.ai, it provides configurable pipelines and components, allowing for a larger degree of freedom and flexibility. The pipelines that were used for evaluation were *pretrained_embeddings_spacy* and *supervised_embeddings*. While both pipelines contain common components such as tokenizers, featurizers, entity extractors and intent classifiers, the major difference between the two is the origin of the word embeddings that are used.

The *pretrained_embeddings_spacy* pipeline extends the components of spaCy [10], a multi-language NLP library. Since Romanian is not a language supported by default by spaCy, a new language model had to be created. The steps for creating the model included gathering pre-trained word vectors (FastText [11], [12] word vectors were employed) and training the statistical models, including the part-of-speech tagger, parser and named-entity recognition using the Romanian Universal Dependencies treebank [13].

The *supervised_embeddings* pipeline, however, makes use of word embeddings that are trained exclusively on the provided data, offering language independence and preservation of the domain-specific meaning of words and relationships.

*2) Wit.ai:* Wit.ai is a closed source Natural Language Understanding tool developed by Facebook, which provides NLU as a service. The learning curve for using the platform is not very steep, making it very suitable for developers without a lot of prior NLU experience. However, with ease of use come the disadvantages of lack of flexibility and lack of control. Moreover, because Wit.ai is a closed source tool the user has no knowledge of its underlying algorithms and configuration parameters.

The platform exposes a set of functionalities through an API, through which training and inference is made possible. The sequence of steps required when training a Wit.ai model is as follows: first, the set of slot types from the dataset have to be posted, after which the training samples can be uploaded. The generated datasets (see [1] for details on generation method) were formatted to fit Wit.ai's expected format.

### B. Results

*1) Rasa NLU:* The results of the *pretrained_embeddings_spacy* pipeline were first evaluated. Table II presents a comparison between the average F1 intent detection and slot filling scores for the models trained on the datasets without and with diacritics. The results indicate that adding diacritics impacts the performance of the model in most of the scenarios. Intent classification is negatively influenced by it throughout all the scenarios, with differences between 0.06 and 0.11, whereas slot labeling only shows significant difference in scenarios 2, 3.1 and 3.2.

TABLE II
RASA NLU - PRETRAINED_EMBEDDINGS_SPACY PIPELINE, WITH DIACRITICS (*dia*) AND WITHOUT (*no dia*)

| | 0 | 1 | 2 | 3.1 | 3.2 | 3.3 |
|---|---|---|---|---|---|---|
| Intent F1 (*no dia*) | 100 | 55.38 | 45.67 | 44.87 | 45.38 | 40.26 |
| Intent F1 (*dia*) | 99.81 | 49.21 | 37.84 | 34.21 | 35.67 | 29.28 |
| Slot F1 (*no dia*) | 100 | 84.49 | 74.14 | 73.59 | 74.44 | 72.55 |
| Slot F1 (*dia*) | 100 | 87.16 | 52.02 | 55.69 | 42.50 | 71.92 |

For a more detailed analysis, we have computed pairs of augmented confusion matrices for intent detection for each pipeline, before and after adding diacritics (Fig 1 and Fig 2 for the *supervised_embeddings* pipeline, Fig 3 and Fig 4 for the *pretrained_embeddings_spacy* pipeline). Each cell contains the scenarios in which a confusion between the predicted intent (column) and the actual intent (line) appeared. The aim was to determine whether new confusions are being introduced with the addition of diacritics and whether it causes existing errors to appear earlier. The abbreviations presented in Table I were used for each intent in the confusion matrices.

While analyzing the sentences which were incorrectly predicted, we noticed that the error types which were previously present in the dataset without diacritics persist. Furthermore, the errors related to intents having an antonymy relationship has significantly increased (e.g. *TurnOnLight* and *TurnOffLight*

| Pred/Act | L_ON | L_OFF | L_LO | L_HI | T_SET | T_LO | T_HI | M_ON | M_OFF | M_VOL | TV_ON | TV_OFF | TV_CH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L_ON | | 1 | - | 1 | 1, 2 | 3,2 | - | 1 | - | - | 1, 2, 3 | - | 2, 3.1, 3.2 |
| L_OFF | 1, 2, 3 | | - | 2 | 2, 3.1 | - | - | 2, 3.1 | 1, 3 | - | 2 | 3 | 2, 3 |
| L_LO | 2, 3.2, 3.3 | - | | 1, 2, 3 | 2 | - | - | - | - | 3,1 | - | - | - |
| L_HI | 2 | 3,3 | 1, 2, 3.2, 3.3 | | - | - | - | - | - | 1, 3.2, 3.3 | - | - | - |
| T_SET | - | - | - | - | | 1, 2, 3.2, 3.3 | 1, 2, 3 | 2, 3.2, 3.3 | - | 3.1, 3.2 | 2 | - | 1 |
| T_LO | - | - | - | - | 1, 2, 3 | | 2, 3 | - | 3,2 | 3.2, 3.3 | - | - | - |
| T_HI | - | - | - | 2, 3.2 | 1, 2, 3 | 1, 3.2, 3.3 | | - | - | 1, 3.1, 3.3 | - | - | - |
| M_ON | - | 3,3 | 3,2 | - | 2, 3.2, 3.3 | - | 3,1 | | - | 3,3 | - | - | 2, 3 |
| M_OFF | 3,1 | 1, 3 | - | - | - | - | - | 3,1 | | 1, 3.1 | - | 3,1 | 3,1 |
| M_VOL | 1 | - | - | - | 1, 3.1 | - | - | 3,1 | - | | - | - | 1, 3 |
| TV_ON | 1 | - | - | - | - | - | - | 3,1 | 3,2 | 3,1 | | - | 1, 2, 3 |
| TV_OFF | - | 1, 3 | - | - | - | - | - | - | 3,1 | 1, 3 | - | | 1, 3 |
| TV_CH | - | - | - | - | - | - | - | - | - | 1, 3 | - | - | |

Fig. 1. Augmented confusion matrix for RASA NLU – SUPERVISED_EMBEDDINGS without diacritics

| Pred/Act | L_ON | L_OFF | L_LO | L_HI | T_SET | T_LO | T_HI | M_ON | M_OFF | M_VOL | TV_ON | TV_OFF | TV_CH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L_ON | | 2, 3.3 | - | - | 2 | - | - | 3,3 | - | - | 1, 2, 3 | - | 2, 3.1 |
| L_OFF | 1, 2, 3 | | - | 1, 2 | 3,3 | - | 2 | 2 | 1, 3.1, 3.3 | - | 2, 3.2 | 3 | 2, 3.1 |
| L_LO | 2, 3 | 2, 3.2, 3.3 | | 1, 2, 3 | 3,1 | - | - | - | - | - | - | - | - |
| L_HI | 2, 3.1, 3.3 | 0, 3 | 2, 3 | | 3,1 | - | - | - | - | 3,2 | - | - | - |
| T_SET | 1 | - | - | - | | 1, 2, 3.1, 3.2 | 1, 2, 3 | 2, 3.1 | - | 3,3 | 2, 3.1, 3.2 | - | - |
| T_LO | - | - | - | - | 1, 2, 3 | | 1, 2, 3 | - | - | - | - | - | - |
| T_HI | - | - | - | 2 | 1, 2, 3 | 3.1, 3.2 | | - | - | 1, 3.2 | 2 | - | - |
| M_ON | - | 2, 3.1, 3.2 | 3,3 | - | 2, 3.1, 3.3 | - | 3,2 | | - | 3,3 | 1 | 3,2 | 2, 3 |
| M_OFF | - | 1, 3 | - | - | - | - | 3,3 | - | | - | - | 3.1, 3.2 | 3,3 |
| M_VOL | - | - | - | - | 1, 3.1, 3.3 | 3,2 | 3,2 | - | - | | - | - | 3,3 |
| TV_ON | 1 | - | - | - | 2 | 3,2 | - | 2 | 3 | 3,3 | | - | 1, 2, 3 |
| TV_OFF | 1 | 1, 3 | - | - | - | - | - | - | - | 1, 3.1, 3.2 | - | | 3 |
| TV_CH | - | - | - | - | - | - | - | - | - | 3 | - | - | |

Fig. 2. Augmented confusion matrix for RASA NLU – SUPERVISED_EMBEDDINGS with diacritics

being mixed up). For instance, when comparing the average number of misclassifications between the intents mentioned above, a 500% increase in scenario 1 (from 5 to 32 misclassifications), and a 900% increase in scenario 2 (from 2 to 21) could be observed on the dataset that contained diacritics. A similar increase could be observed for the intent pairs *IncreaseIntensityLight - DecreaseIntensityLight* and *IncreaseTemp - DecreaseTemp*.

By analyzing the corresponding augmented confusion matrices, it can be determined which intent misclassifications appear earlier (see Figures 1 - 2). Some are caused by a marginal confidence difference when predicting intents of opposite meaning, which appeared as early as the base scenario for the diacritics dataset, while they only appear starting from scenario 1 on the dataset without diacritics. A possible cause for this is the cosine similarity difference between the same words written with and without diacritics. For instance, the pair ”opresti”/”pornesti” has a cosine similarity of 0.58, while ”opresți”/”pornești” has a higher similarity, of 0.68.

From the same matrices, we can determine which additional confusions were generated only by the dataset with diacritics. Most of them are errors inside the same super-class, especially inside the media one, in the last scenario, and inside the light super-class, starting as early as scenario 1.

For the slot filling task, we analyzed the scenarios with a significant decrease in performance: 2, 3.1 and 3.2. A new error that did not occur in the first dataset is predicting the word ”grade” (”degrees”) as having the slot type Artist, in sentences in which a plural temperature (e.g. ”douăzeci de grade.” – ”twenty degrees”) is at the end of the sentence, right after the location (Room type slot). This error maps exactly for the scenarios which were mentioned above. In addition, for scenario 3.2, ”dormitor” (”bedroom”) is not successfully labeled as a Room slot when it is present in the middle of a sentence, before a Level type slot.

Table III presents the results obtained on the *supervised_embeddings* pipeline on the datasets without and with diacritics.

TABLE III
RASA NLU - SUPERVISED_EMBEDDINGS PIPELINE, WITH DIACRITICS (*dia*) AND WITHOUT (*no dia*)

| | 0 | 1 | 2 | 3.1 | 3.2 | 3.3 |
|---|---|---|---|---|---|---|
| Intent F1 (*no dia*) | 100 | 43.08 | 34.00 | 41.03 | 33.33 | 33.08 |
| Intent F1 (*dia*) | 99.90 | 36.37 | 35.51 | 18.79 | 19.19 | 16.14 |
| Slot F1 (*no dia*) | 100 | 85.55 | 68.72 | 71.04 | 75.07 | 72.55 |
| Slot F1 (*dia*) | 99.94 | 88.89 | 60.79 | 74.93 | 46.18 | 75.62 |

Once again, intent classification is negatively influenced by the addition of diacritics throughout all the scenarios. For scenario 1, new in-class errors appear for each class. An error which appeared only in the new dataset is the prediction of the ”scadeTemperatura” intent, when the actual intent was ”pornesteTV”. However, the error is encountered only once, only in scenario 3.2.

Slot filling only shows a significant decrease in scenarios 2 and 3.2. For scenario 2, the temperature slot is not labeled at all if written with letters instead of numbers, compared to scenario 1 where it is partially labeled (”de grade”, instead of ”douăzeci și patru de grade”, for instance). In scenario 3.2, for the imbalanced temperature class, in some cases,

| Pred/Act | L_ON | L_OFF | L_LO | L_HI | T_SET | T_LO | T_HI | M_ON | M_OFF | M_VOL | TV_ON | TV_OFF | TV_CH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L_ON | | 3,3 | 2, 3 | 1, 2, 3 | - | - | 2, 3.1 | - | 3.1, 3.2 | 3.2, 3.3 | 2, 3.1 | 3.1, 3.2 | 2 |
| L_OFF | 1, 2, 3 | | 2, 3 | 1, 2, 3 | - | 3,1 | 2, 3.1 | - | 3,1 | 3,3 | 3,1 | 3.1, 3.2 | 2 |
| L_LO | - | - | | 1, 2, 3 | - | 2, 3.1, 3.2 | 2 | - | - | 3,2 | - | 3,1 | - |
| L_HI | 2, 3 | - | 2, 3 | | 3,1 | 3.1, 3.3 | 3,3 | - | - | - | - | 3.1, 3.2 | - |
| T_SET | - | - | 1 | 1, 3.2 | | 1 | 1, 2, 3 | 1, 2, 3 | - | 3,1 | - | 3,2 | - |
| T_LO | - | - | 2 | 1, 3.2 | 1 | | 1, 2, 3 | - | - | 3,2 | - | - | - |
| T_HI | 2 | - | 3,1 | 1, 2, 3 | 3.1, 3.3 | 1, 2, 3.1 | | 3,2 | - | 3,2 | - | - | - |
| M_ON | - | - | - | 3,3 | - | - | - | | 3.1, 3.3 | - | - | - | 2 |
| M_OFF | - | 3,3 | 3.2, 3.3 | 1, 3.3 | - | - | - | 3,3 | | - | 3.1, 3.3 | - | - |
| M_VOL | - | - | 3,3 | - | - | - | 3,3 | - | - | | - | 1, 3 | - |
| TV_ON | - | - | - | 1, 2 | 3,3 | - | - | 3,1 | - | - | | 3,1 | 1, 2, 3 |
| TV_OFF | - | 3,3 | 3,3 | 3.2, 3.3 | - | - | - | 3,3 | 1, 3 | - | 1 | | - |
| TV_CH | - | - | 3,3 | 3,3 | - | - | - | 3,3 | 3,3 | - | 1, 2, 3 | 3.1, 3.2 | |

Fig. 3. Augmented confusion matrix for RASA NLU – PRETRAINED_EMBEDDINGS_SPACY without diacritics

| Pred/Act | L_ON | L_OFF | L_LO | L_HI | T_SET | T_LO | T_HI | M_ON | M_OFF | M_VOL | TV_ON | TV_OFF | TV_CH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L_ON | | 3 | 2, 3 | 2, 3 | - | 2, 3.1 | 3,1 | 2 | 3,2 | 3 | 2, 3.1 | 3 | 2 |
| L_OFF | 1, 2, 3 | | 1, 2, 3 | 1, 2, 3 | - | 2, 3.1, 3.3 | 3 | 2 | 3 | 3 | 2 | 3.1, 3.2 | - |
| L_LO | 1, 2, 3.1, 3.2 | 1, 2, 3 | | 1, 2, 3 | 3,1 | 3.1, 3.3 | 2, 3.1, 3.3 | 2 | - | 3,1 | - | 3 | - |
| L_HI | 2, 3 | 3,2 | All | | 3,1 | 2, 3.1, 3.3 | 2, 3.1 | - | - | 3,1 | - | 3.1, 3.2 | - |
| T_SET | - | - | 1, 3.2 | 1, 3.1, 3.2 | | 0, 1, 2 | 1, 2, 3 | 2, 3.1, 3.2 | - | - | - | - | - |
| T_LO | - | - | 1, 3.2 | 1, 3.2 | 1, 2, 3 | | All | 3,2 | - | 3.1, 3.2 | - | - | - |
| T_HI | - | - | 2, 3 | 1, 2, 3 | 1, 2, 3 | All | | 3,2 | - | 1, 3.2 | 1 | - | - |
| M_ON | - | - | 2, 3.3 | 3,3 | - | - | - | | 3 | 3,3 | - | - | 3,3 |
| M_OFF | 1, 3.3 | 1, 3.2, 3.3 | 1, 3 | 3,3 | - | - | - | - | | 3,3 | 3 | 1, 3 | 3,3 |
| M_VOL | - | - | 3,3 | 3,1 | - | - | - | - | 3,3 | | 3,3 | - | - |
| TV_ON | - | - | 1, 2, 3.2, 3.3 | 1, 2, 3.2, 3.3 | 3,3 | - | - | 2, 3.1, 3.2 | 3,3 | - | | 1 | 1, 2, 3 |
| TV_OFF | 1, 3.1 | 3,3 | 3 | 3,3 | - | - | - | - | 1, 3 | 3,3 | 1, 3 | | 1, 3.1, 3.3 |
| TV_CH | - | - | 2, 3.3 | 2, 3.3 | 3,3 | - | 3,3 | 2 | 3.1, 3.3 | 3.1, 3.3 | 1, 2, 3 | 3.1, 3.2 | |

Fig. 4. Augmented confusion matrix for RASA NLU – PRETRAINED_EMBEDDINGS_SPACY with diacritics

the slot labeling fails to identify both the location slot and the temperature level slot, in utterances in which the level slot is followed by the location slot, at the end of the sentence.

*2) Wit.ai:* For *Wit.ai*, the results obtained on the datasets without and with diacritics are presented in Table IV below. We can see that the performance of the slot filling models decreases significantly as we introduce the diacritics in our dataset (with an absolute performance gap of 42.59% in the last scenario). For intent detection the performance is slightly better in some scenarios, with an improvement of 0.51% to 4.87%.

By analyzing the potential causes for this drop in performance for the slot filling case we found that some of the errors are generated by the fact that *Wit.ai* stores the dataset using the same encoding used in our dataset (i.e. UTF–8), but the models are trained using a different encoding. This makes the models provided by *Wit.ai* unsuitable for datasets with diacritics.

TABLE IV
WIT.AI - WITH DIACRITICS (*dia*) AND WITHOUT (*no dia*)

| | 0 | 1 | 2 | 3.1 | 3.2 | 3.3 |
|---|---|---|---|---|---|---|
| Intent F1 (*no dia*) | 100 | 39.74 | 38.66 | 29.48 | 29.74 | 27.17 |
| Intent F1 (*dia*) | 96.66 | 44.61 | 31.33 | 28.71 | 30.25 | 29.23 |
| Slot F1 (*no dia*) | 99.10 | 86.97 | 76.17 | 73.01 | 75.85 | 79.18 |
| Slot F1 (*dia*) | 79.17 | 70.77 | 37.14 | 31.57 | 36.59 | 36.59 |

## V. CONCLUSIONS

This paper aims to provide an empirical analysis of the effect of diacritics on the performance of existing intent detection and slot filling approaches, focusing on a specific application scenario and data: Romanian home assistant. We extend on previous work which analyzed several data and language-specific complexities that are prone to appear in such a scenario [1], and focus on benchmarking two of the most prominent existing tools: the *RASA* framework [2] and *Facebook's Wit.ai* API [3]. The results indicate that diacritics introduce a new level on learning complexity and that their presence amplifies other sources of errors, especially the incapacity of word embeddings to separate between antonym verbs (initially discovered in [1]). Also, slot filling seems to be affected to a larger extent by the usage of diacritics, since new types of errors have emerged during the current analysis.

## ACKNOWLEDGMENT

The work presented in this paper was supported by grant no. 72PCCDI/01.03.2018, *ROBIN - Robots and Society: Cognitive Systems for Personal Robots and Autonomous Vehicles*.

## REFERENCES

[1] A. Stoica, T. Kadar, C. Lemnaru, R. Potolea, and M. Dînşoreanu, "The impact of data challenges on intent detection and slot filling for the home assistant scenario," in *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 2019, pp. 41–47.

[2] T. Bocklisch, J. Faulkner, N. Pawlowski, and A. Nichol, "Rasa: Open source language understanding and dialogue management," *CoRR*, vol. abs/1712.05181, 2017. [Online]. Available: http://arxiv.org/abs/1712.05181

[3] "Wit.ai," Natural Language for Developers. [Online]. Available: https://wit.ai/
[4] "DialogFlow." [Online]. Available: https://dialogflow.com/
[5] J. Williams, E. Kamal, M. Ashour, H. Amr, J. Miller, and G. Zweig, "Fast and easy language understanding for dialog systems with microsoft language understanding intelligent service (luis)," 01 2015, pp. 159–161.
[6] "IBM Watson." [Online]. Available: https://www.ibm.com/watson
[7] "Rasa," Rasa NLU: Language Understanding for Chatbots and AI assistants. [Online]. Available: https://rasa.com/docs/rasa/nlu/about/
[8] "Snips," Snips Python library to extract meaning from text. [Online]. Available: https://github.com/snipsco/snips-nlu
[9] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril *et al.*, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *arXiv preprint arXiv:1805.10190*, 2018.
[10] "SpaCy," Industrial-Strength Natural Language Processing. [Online]. Available: https://spacy.io/
[11] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.
[12] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext. zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.
[13] V. Barbu Mititelu, R. Ion, R. Simionescu, E. Irimia, and C.-A. Perez, "The romanian treebank annotated according to universal dependencies," in *Proceedings of the tenth international conference on natural language processing (hrtal2016)*, 2016.