Multi-Domain Named Entity Recognition for Robotic Process Automation

Aleksander Denisiuk
University of Warmia and Mazury
Faculty of Mathematics and Computer Science, Olsztyn, Poland
denisiuk@matman.uwm.edu.pl

Maria Ganzha, Piotr Sowinski
Warsaw University of Technology, Warsaw
Poland,
maria.ganzha@pw.edu.pl

Katarzyna Wasielewska-Michniewska[†]
Marcin Paprzycki[†‡]
[†]Systems Research Institute Polish Academy of Sciences
[‡]Warsaw Management University
Warsaw, Poland
firstname.lastname@ibspan.waw.pl

Abstract

To make Robotic Process Automation more attractive, it needs to become more "intelligent". In this context, a modification of the Form-to-Rule approach, based on identifying data types of form fields, is proposed. Moreover, multi-domain named entity recognition is used, for field value identification. These techniques, used jointly, allow software robots to adapt to interface changes. Experimental results are reported and verify viability of the proposed approach.

1. Introduction

The purpose of Robotic Process Automation (RPA) is to create, so called, software robots that systematically repeat simple users' actions, such as logging into a system, receiving and processing e-mails, or transferring data between applications [1]. RPA systems can also aid businesses in data mining – for example, by extracting information from legacy systems [2]. Here, key differences between proposed solutions, are the level of human involvement, and type of source data. Naturally, solutions should limit human-in-the-loop presence by, e.g., handling multi-domain data, and be flexible with regard to possible data sources. Specifically, considering that many systems have web interfaces, the concept of web content mining (mining information from web documents, often using NLP techniques) is applicable [3]. The solution, outlined in this work, extends web content mining, since it is applicable to screenshots that can originate from any system, supporting business analytics and process mining [4]. Moreover, the proposed approach to information extraction is independent from any pre-configured structure of the sources.

Recently, it has been stipulated that RPA systems can be substantially improved, by inclusion of self-learning [5]. One of the first attempts to do this was proposed in [6]. Namely, the Form-to-Rule (F2R) approach was proposed. Overall, the F2R life-cycle consists of the following steps:

1. A human identifies *forms* in the IT system.
2. RPA robot collects process logs and deduces applicable *rules*.
3. Rules are applied to achieve business goals.
4. A human verifies (and, possibly, adjusts) *rules* that have been identified by the RPA robot.

In this context, a *form* is as a set of fields that are treated jointly. Each field is identified by a unique identifier (ID). Moreover, a *process* consists of a sequence of actions that can be translated to changes of form fields. Each change is called a *Form Action* (FA). In F2R, a *rule* consists of two parts: a *condition* and a *response*. A condition indicates whether an FA satisfies certain requirements. A response triggers an appropriate, new FA. In such a way, rules implement an RPA definition using "if ... then ..." statements [7].

The F2R approach reduces the "human-dependent" aspect of an RPA-based system, by increasing the responsibilities of the robot. In this work, the possibility of moving one more step in this direction (that is, shifting work from humans to RPA robots) is explored.

Consider the form definition step. The F2R approach assumes that a person, who defines a form, is familiar with the program's interface. Moreover, assigning IDs to individual form fields requires an interface that allows a robot to identify the fields that were changed by the user. Notice that if one were to use a different application at a given step of the process (following a different "branch"), it would be impossible to apply the previously learned rules. Thus, the RPA rules would have to be recreated from scratch.

The aim of this contribution is to show that a form can also be identified not by an ID, but by the type of the stored data. Moreover, the datatype of a field can determined by applying Named Entity Recognition (NER). Furthermore, it is sufficient to provide a screenshot as the input to the system performing rule creation, as standard OCR software can translate a screenshot into text. Afterwards, a NER subsystem can extract the field data, including corresponding data types. This information can be used in subsequent steps of the F2R approach.

The feature extractor, for the proposed NER system, was introduced in [8]. There, the key role was played by local context features of an n-gram, which turned out to be very important for recognizing the "meaning" of an entity. Here, the feature extractor has been integrated into the F2R process and progress in its development is reported. Namely, the developed system has been tested with several classifiers, i.e.: decision trees, random forest, and boosting techniques; obtaining satisfactory results (see, Section 5).

It should be stressed that the proposed system is assumed to be multi-domain, as the user works with multiple different applications, installed on the computer. However, in RPA practice, the domains of operation are likely to be *similar*, since applications process tasks within "the same office". Therefore, a use case in which a company is processing invoices from companies and schools has been considered to be reasonable, to experimentally validate the proposed approach. For this use case, three datasets have been prepared: *Companies 400*, *Companies 500* (containing company data) and *School* (containing similar data, but related to schools). Obviously, these sets have at least one common part, the address. More details about the datasets used in experiments, can be found in section 4.2.

The remaining content is organized as follows. In Section 2, a short survey of related work, from both RPA and NER domains, is presented. Next, the form detection step in the F2R process is redefined. In the new definition, a form is uniquely identified by the types of its fields. In this context, a modified NER system, based on the feature extractor presented in [8], was implemented and tested. Details on the data set and the experimental setup are given in Section 4. Obtained results are described in Section 5. Finally, a discussion of experimental results, conclusions and an outline for future work conclude the text.

2. Related works

Robotic Process Automation has recently gained a lot of attention both in the scientific community and in businesses. However, the adoption of RPA tools requires a lot of manual effort, e.g., in identification and precise definition (in some programming language specific to the RPA platform) of the to-be-automated tasks. In this context, the Form-to-Rule approach has been proposed (see, [9]) to automatically identify tasks, deduce, and apply rules. It is easy to observe that, when instantiating an RPA-based system, one of core challenges is for the robot to "understand" what data is transferred by the user from one application/form/file to another. Note that none of the existing RPA tools, analyzed in the recent survey [5], have a self-learning capability. Such ability would allow an RPA robot to gain understanding of the data it is working with, possibly without the need for human intervention.

Reflecting on the problem at hand, it can be observed that Named Entity Recognition (NER) may support the implementation of self-learning. NER is a relatively "old" field of computer science, as it can be traced to the end of last century [10, 11]. It has been developed to locate and categorize important fragments (called *named entities*) within unstructured texts. Typical examples of named entities are dates, city names, money amounts, etc. NER is an important component of tasks within information retrieval [12, 13], entity linking, query answering [14, 15], machine translation [16, 17], relation extraction, and sentiment analysis. It is also noteworthy that NER methods have been adapted to work with many languages [18, 19], including Polish [20, 21, 22]. In this context, in [23] authors described and AI-based system for business documents (e.g. letters, e-mails, images) processing, for a debt collecting use case.

A typical NER pipeline was described in [10, 11]. First, if necessary, given text is turned into a digital format (e.g., using OCR-based digitization and text extraction) and pre-processed. Pre-processing may involve, among others, removal of tags and/or stop-words, stemming, etc. Here, specific pre-processing steps are application (and document) dependent. Next, the resulting text is divided into n-grams. Then, each n-gram is converted into a feature vector. Finally, feature vectors are used to train a classifier, which is tasked with categorizing n-grams appearing in a production system. It should be noted that such a NER pipeline is classifier-independent, which is the main reason for experimenting with multiple classifiers in this contribution.

Typically, NER research has been focused on designing models for a specific dataset and/or single knowledge domain. The challenge that started being addressed recently, is applying NER to multiple domains at once. Here, note that since collecting (and labeling) extensive datasets is time-consuming, it would be beneficial to have cross-domain NER models that could be transferred to a target domain by using as few training samples as possible. In other words, it would be beneficial if techniques similar to transfer learning (see, [24]) could be applied to NER systems.

The multi-domain NER problem is actively studied [25, 26, 27]. However, authors of mentioned works emphasise that a NER system trained, and relatively successful, in one domain, usually shows (very) poor performance in another. While a similar phenomenon has been observed in experiments reported in what follows, one should keep in mind that it is very natural for the RPA-based scenarios to involve humans. In other words, real-world RPA applications differ from other areas, where NER has been considered, in that human users systematically monitor/supervise/correct RPA-obtained results. Therefore, the proposed solution is realistic as one can assume that a user will correct the mistakes of the NER system (in a new domain), and the corrected data will be added to the training set. This can be seen as an example of so-called active learning [28]. As a matter of fact, experiments reported in Section 5.2 indicate that NER systems can adapt to a new domain using the active learning-inspired approach.

Interestingly, the order in which knowledge is infused into a multi-domain NER system matters. For instance, the idea to utilize NER knowledge learned from high-resource domains and then adapt it to low-resource domains (which is called cross-domain NER) is explored in [29, 30] (for an unsupervised scenario) and in [31, 32] (for a supervised one).

An important question is how to evaluate the performance of a NER pipeline. Authors of [33] introduce three experimental setups that provide a framework for evaluating the robustness of NER models. In [27] authors propose a framework and a corpora of texts to analyze and discuss the performance of state-of-the-art tools in terms of their robustness and reliability. In [26] the drawbacks of often-used datasets are identified, and a new cross-domain dataset is introduced – a fully-labeled collection of NER data, spanning five diverse domains, with specialized entity categories for different domains. Next, existing cross-domain NER models are evaluated on that data set, and explored using different levels of domain corpus and masking strategies.

Since this work is based on Polish-language documents, let us note that an example of NER application to Polish stock exchange reports is given in [25] (recognition of persons and companies). A cross-domain evaluation on a small corpus of police reports was also presented, however the results are significantly worse.

3. Redefinition of the form identification

Proceeding toward the description of the proposed solution, let us briefly summarize the main concepts introduced in [6]. The original article should be referred to for more details and formal definitions. The basic notion, in the F2R process, is a form. It is defined as a set of form fields and is identified by its name. The form name uniquely defines the set of fields. Here, the *business process log* consists of a series of Form Actions (FA), i.e. changes of the form field values.

An RPA agent learns rules on the basis of process logs. Here, the rules are in the form `if` *condition* `then` *response*, where the *condition* is an FA that can be automatically verified, and the *response* is an FA that can be automatically performed.

In this work, a modification of the notion of a form is proposed. Namely, the form can be identified by the set of its fields. More precisely, by the set of types of its fields. Even though the notion of a form is to be modified, the remaining part of the self-learning RPA process coincides with that introduced in [6].

To formally redefine the F2R process, the following definitions are modifications of Definition 1 from [6]. Here, the notation from this reference is preserved. Specifically, for an arbitrary set X we denote by $\mathcal{P}(X)$ the *power set* of X, i.e. $\mathcal{P}(X) = \{ X' | X' \subset X \}$. A multiset that allows its elements to occur multiple times is written in brackets, e.g. $[a^2, b]$. The set of all possible multisets for X is denoted by $\mathcal{B}(X)$.

Definition 1 (Form field) *Let $\mathcal{U}_{\mathrm{tp}}$ be the universe of all form field types, $\mathcal{U}_{\mathrm{Val}}$ denote the universe of possible form fields values. A form field is a pair $(t, V) \in \mathcal{U}_{\mathrm{tp}} \times \mathcal{P}(\mathcal{U}_{\mathrm{Val}})$, where $t \in \mathcal{U}_{\mathrm{tp}}$ is the type of the field, $V \subset \mathcal{U}_{\mathrm{Val}}$ is a set of field values. Denoted by $\mathcal{U}_{\mathrm{fld}}$ is the set of all possible form fields in the system, for which*

$$\forall F = (t, V), F' = (t', V') \in \mathcal{U}_{\mathrm{fld}} \quad (t = t' \Rightarrow V = V'),$$

is enforced, i.e. the type of a form field uniquely defines the set of values associated with it.

The type of a form field is understood in the sense of named entities, and consists of pre-defined categories, such as person name, organization, location, monetary value, etc.

Definition 2 (Form) *Let $\mathcal{U}_{\mathrm{frm}}$ be the universe of all form field types. A form is defined as a multiset of form fields $f \in \mathcal{B}(\mathcal{U}_{\mathrm{tp}})$. Denoted by $\mathcal{U}_{\mathrm{frm}}$ is the set of all possible forms in the system.*

The difference between the proposed definition of a form, and the definition from [6] is that the original definition required an ID of a form to identify it. In the proposed definition, the form is completely identified by the set of it fields, i.e. by the set of its field types.

The evident advantage of the proposed approach is the resulting stability, with respect to interface changes. Once a rule is learned, it can be applied, even after change of the applications used in the process (as long as the filled form types remain unchanged). On other hand, it can be assumed that an RPA robot can identify the type of a form field in an application. This is particularly so, since in what follows we illustrate (on the level of a *proof of concept*) that this can be successfully achieved by the application of NER techniques.

4. Experimental setup

As stated, the core of the proposed approach is to use multi-domain NER in the F2R pipeline. Therefore, let us now describe the setup of the experiments that were used to validate the utility of such an approach.

4.1. NER pipeline for the RPA systems

In the experiments, we adopted the same NER pipeline (figure 1) as the one that was used in [8].

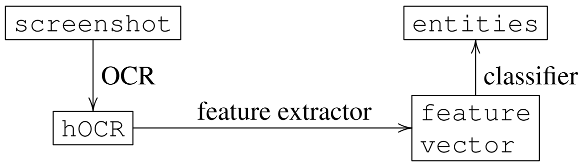

$$\begin{tikzcd}
\boxed{\texttt{screenshot}} \arrow[d, "\text{OCR}"] & \boxed{\texttt{entities}} \\
\boxed{\texttt{hOCR}} \arrow[r, "\text{feature extractor}"] & \boxed{\texttt{feature vector}} \arrow[u, "\text{classifier}"']
\end{tikzcd}$$

Figure 1. NER pipeline

In general, it is assumed that any OCR software can be used within the first step of the pipeline. In the reported experiments, the Tesseract software [34] with Polish language recognition interface was used as the OCR program. It turned out that the Tesseract option `-psm 4` delivered the best output results, with respect to the text segmentation.

The same feature extractor, which has been used in [8] has been applied. There, the most of commonly used NER features [10, 11] have been extracted, as well as the *local context* (see, [8] for all details).

Three classifiers were tested in the proposed NER pipeline: (1) decision tree, (2) AdaBoost, and (3) random forest. In each case, the standard implementation from the Scikit-learn library [35] has been used. Hence, the documentation of Scikit-learn should be consulted for details concerning the classifiers.

4.2. Data preparation

The second core contribution of this work is the exploration of the performance of the proposed NER pipeline in the multi-domain context. For this purpose the following datasets were prepared, and used in the experiments:

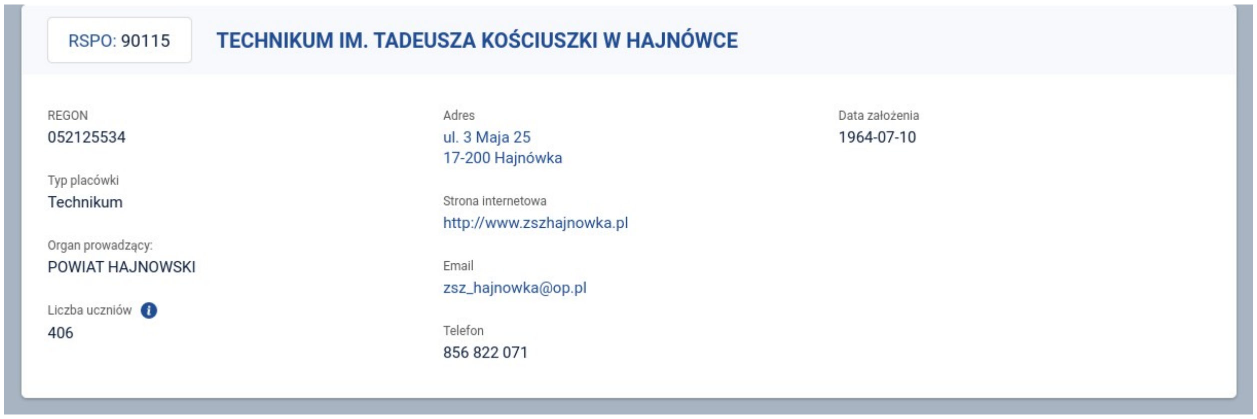
- **Companies 400** – extracted from the official database of business organizations registered in Poland, maintained by Statistics Poland; 21 screenshots.
- **Companies 500** – extracted from an alternative database of Polish business organizations maintained by the "Foundation ePaństwo"; 28 screenshots.
- **Companies 0** – data from arbitrarily chosen company websites; 56 screenshots.
- **Schools** – school information in the original layout; 92 examples.
- **Schools flipped** – flipped school information (columns to rows); 75 screenshots.
- **Schools shuffled** – school information with randomly shuffled field order; 84 examples.

The first three datasets were partially used in previous work [8], while the remaining have been added to represent the multi-domain settings. Specifically, for schools, an entirely new dataset has been prepared. The source of used data for schools is publicly available at `https://rspo.gov.pl/`, "Polish registry of schools" (*Rejestr Szkół i Placówek Oświatowych*).

For the *Schools* dataset, the syntactic *Schools flipped* dataset, as a dual one, was generated. The *Companies 0* and *Schools shuffled* datasets were introduced to avoid model overfitting, and they were used only for model training (not for testing).

Contrary to the *Companies* datasets, the process of the *School* dataset generation was automated. First, for each school to be included in the dataset, a screenshot of its entry in the registry had to be obtained. To make the process faster, a browser extension was developed. It automatically scrolls the page and identifies each visible entry and saves the screenshot. Additional datasets, i.e. *Schools shuffle* and *Schools flipped*, were generated from the original *Schools* dataset, using HTML DOM manipulation. An example screenshot from the *Schools* dataset is presented in Figure 2.

Collected screenshots were fed to the OCR software. To avoid having to manually annotate the OCR-processed screenshots (specifying the selected


RSPO: 90115

TECHNIKUM IM. TADEUSZA KOŚCIUSZKI W HAJNÓWCE

REGON
052125534

Typ placówki
Technikum

Organ prowadzący:
POWIAT HAJNOWSKI

Liczba uczniów
406

Adres
ul. 3 Maja 25
17-200 Hajnówka

Strona internetowa
http://www.zszhajnowka.pl

Email
zsz_hajnowka@op.pl

Telefon
856 822 071

Data założenia
1964-07-10

Figure 2. Example screenshot from the School dataset (original layout)

fields), an automatic approach was used instead. A CSV file, containing all data about the schools was retrieved. Next, using a Python script leveraging fuzzy string matching, the text fragments present in the OCR-processed screenshot were matched with those from the ground truth CSV file. Finally, for each screenshot, annotations were generated. Hence, the entire dataset generation process was fully automatic.

Taking into account the nature of the collected data (information about named institutions), all datasets have common entities: `addressLocality`, `streetAddress`, `postalCode`, `name`, `average`. Moreover, the *Companies 400* dataset contains additional `taxID` entity. The *Companies 500* dataset, additionally, includes two entities: `taxID` and `addressRegion`. Finally, the *Schools* dataset defines four new entities: `email`, `regon`, `foundingDate` and `telephone`.

Note that all considered entities are defined in the Schema.org vocabulary, except for the `regon` entity, which is an official statistical identifier of business entities registered in Poland.

The *School* dataset, produced as a result of this work, is published on Zenodo and GitHub under an open license. It is available at `https://zenodo.org/record/6091666`.

5. Experimental results and their analysis

Overall, two groups of tests have been performed. Firstly, the standard 20% cross-validation test was done (section 5.1). Here, 80% of data has been used for model training and the remaining 20% for testing. The second group of tests, described in section 5.2, models the situation when the application interface changes. Here, the system is trained on a part of data and then tested on data that was not used in the training process. Obviously, some entities will have a poor recognition score. This being the case, the experiment was extended by adding to the training set from one to ten screenshots from the test dataset. This corresponds to the scenario in which the user corrects the NER system (active learning) and "rebuilds" the classifier. As expected, the results of experiments demonstrate an increasing trend in the recognition rate.

As a measure of correctness of entity recognition the, commonly found in NER-related literature, F1 measure [10] has been applied.

As noted, three different classifiers were tested: decision tree, random forest, and AdaBoost. In the case of decision trees, each experiment was repeated 20 times, to overcome the inherent instability present in these models. For this classifier, the average score is reported.

Note that in some tests classifiers delivered false positive results for entities that were not a part of the particular dataset. For instance, `AddressRegion` was incorrectly recognized in one screenshot of the *Companies 500* series. However, such cases were very rare, and did not have a significant impact on the experiment statistics. Therefore, for the sake of clarity, they were not included in figures 3–12.

5.1. Cross-validation test

The first test is a standard 20%-fold cross-validation test. The data was randomly split into two groups: one for training (80%), with the remaining part kept for testing. The second group contained only data from *Companies 400*, *Companies 500* and *School* datasets. Specifically, ten screenshots from *Companies 400* and *Companies 500* datasets, and 20 from the *School* datasets have been randomly selected for testing.

In figure 3, the first column for each entity represents the F1 recognition score for the decision tree, the second one for AdaBoost, while the third for the random forest classifier.

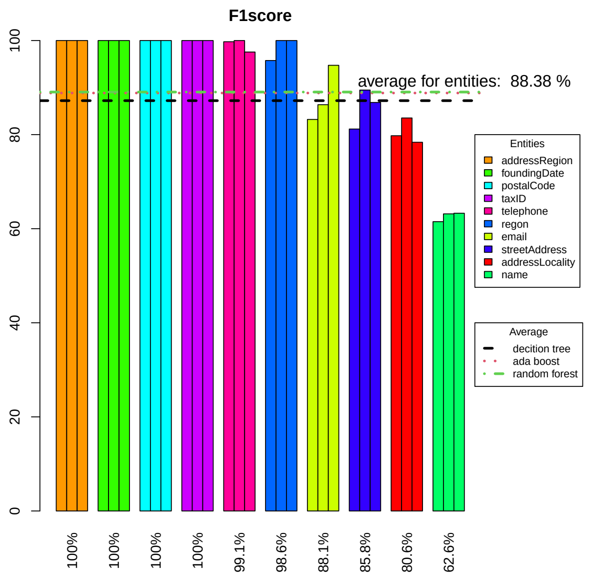


Figure 3. Results of the 20%-fold cross-validation test. Average F1 scores for the three classifiers are presented

It is easy to observe that *simple* entities, such as `addressRegion`, `postalCode`, `taxID`, and `foundingDate`, are stably recognized, with 100% accuracy. The remaining entities (but one) also have satisfactory, high (above 80%), recognition rate. As could have been expected, random forest and AdaBoost deliver a slightly better recognition rate than the decision tree.

The relatively low F1 score for the *name* entity (about 60%) can be explained by the fact that the *name* entity has no rigid internal structure, while the length of it in the datasets varies from one to seventeen words. Moreover, sometimes it is placed across two lines. Obviously, situations like these need to be take into account during requirements analysis, when a real-world RPA system is to be instantiated. However, this is out of the scope of this contribution.

5.2. Modeling change of interface

In this series of experiments, the case of an interface change has been investigated. Assume that the user performs the same process as above, at one of the stages they decide to use a different application. Here, the NER system that has learned one interface, should recognize entities in the other interface. In general, this problem can be conceptualized as a case of a multi-domain NER. As mentioned, this problem is still investigated and, in

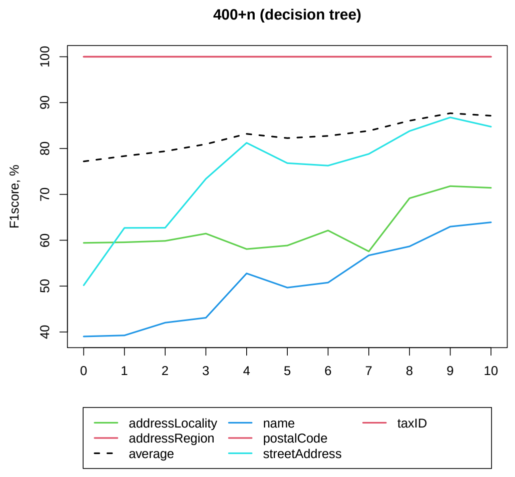


Figure 4. Adaptive test for 400 series data, decision tree classifier

the most general case, NER systems often show poor performance when the domain changes.

However, in the case of RPA systems, an assumption can be made that the two domains are not too distant from each other, and this assumption is, implicitly, guiding the performed experiments. Specifically, the following modification for real-world RPA applications is proposed. Assume that the user corrects the system, after spotting a mistake. Then the system re-learns the F2R rule. For the purpose of this work, this process is called *adaptation*.

Considered interface change was modelled in the following way. The *Companies 400* and *Companies 500* datasets have been considered as interfaces to different applications. For the *School* dataset a new domain has been created. Namely, for some *School* screenshots rows and columns have been swapped with each other. This new dataset was treated as a new interface.

The NER system has been trained on all data except the selected interface set (different interface), and then tested on this data. To model the adaptation process, one to ten randomly chosen screenshots from the different interface have been moved from the test dataset to the train dataset.

In the first test, *Companies 500* and *Companies 0*, *Schools* and *Schools shuffled* dataset were used for training. The *Companies 400* dataset was used for testing. The experimental results are shown in figures 4–6.

As can be seen, “simple” entities are stably recognized at the rate of 100%. More complicated entities, such as `name`, `addressLocality` or

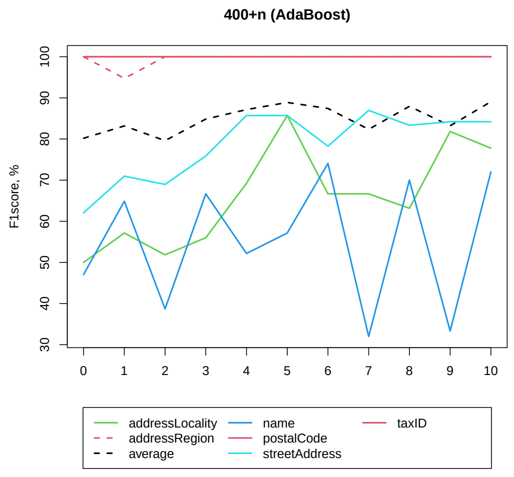


Figure 5. Adaptive test for 400 series data, AdaBoost classifier

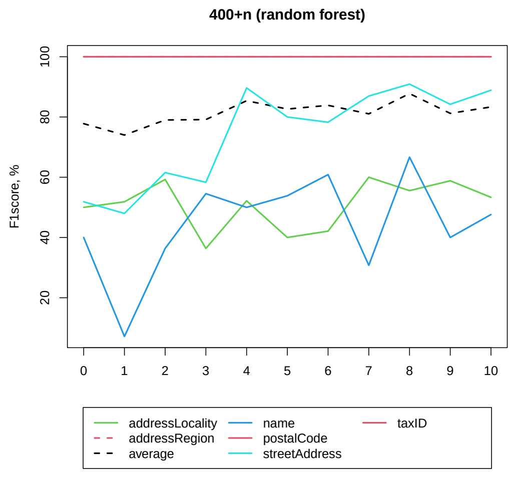


Figure 6. Adaptive test for 400 series data, random forest classifier

`streetAddress`, have a smaller recognition rate, but have a clear upward trend (result of active learning). Note also that the decision tree classifier demonstrates the most stable growth of F1 score.

In the second set of tests, *Companies 400*, *Companies 0*, *Schools* and *Schools shuffled* datasets were used for training, while *Companies 500* was used for testing. The results are shown in figures 7–9.

Similarly to the previous experiments, entities `name`, `addressLocality`, and `streetAddress`, have a lower recognition rate with respect to other entities. However, a clear and stable upward trend can also be observed.

In the final tests *Companies 400*, *Companies 500*, *Companies 0*, *Schools*, and *Schools shuffled* datasets were used for training. The *Schools flipped* dataset was used for testing. Experimental results are shown in figures 10–12.

In this case, although it is still visible, the increasing trend is not as pronounced as in the previous experiments. One can observe a particular problem with the entity `name`, which is very poorly recognized. In this case, the same explanation as above, related to the long and unstructured text within this entity, applies. The random forest classifier encounters a similar problem (for the same reasons) in the case of the `streetAddress` entity.

6. Discussion

The presented tests confirm the main thesis: a modification of the F2R approach, involving the NER technology, is promising. The results of the 20% cross-validation test, applied to the proposed approach, show that an RPA system can be defined in terms of form (data) type, rather than form fields ID and the corresponding API, extending the robot-responsibility aspects of an RPA pipeline.

Moreover, the suggested approach covers the situations when the application changes in a specific step of the process. Experiments, described in section 5.2 demonstrate that, for *simple* entities, the NER system copes with this issue very well. As what concerns other entities, the proposed *adaptation* process (based on the ideas from active learning), can help in most situations.

Let us also discuss some benefits of the proposed approach in comparison to other ways of addressing the problem. Firstly, the proposed solution seems to be the first attempt to join two fields of, broadly understood, data mining: NER and process mining. The need of such a conjunction arises from the effort to create an RPA system with a self-learning ability. Recall, that none of existing RPA system has such a feature [5].

The initial attempt at implementing an RPA system with self-learning, through form-to-action techniques, was reported in [6]. The approach presented in this contribution improves the form definition step, by changing form fields definition. Instead of IDs, using the field's datatype is proposed. This allows to formulate rules without the access to the internal structure of the program's interface. Specifically, once learned, the rule is relatively stable, with respect to interface modifications or even a change of application. Moreover, use of active learning-based techniques

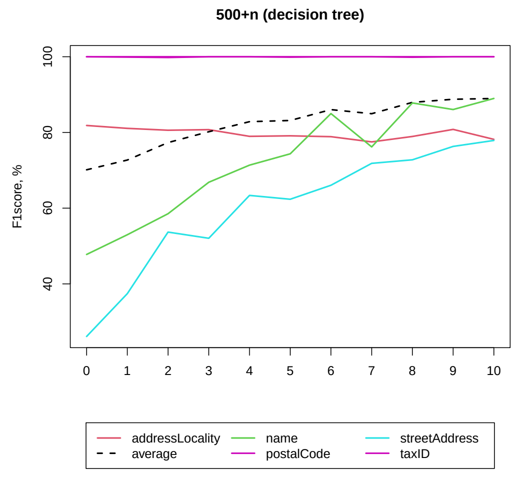


Figure 7. Adaptive test for 500 series data, decision tree classifier

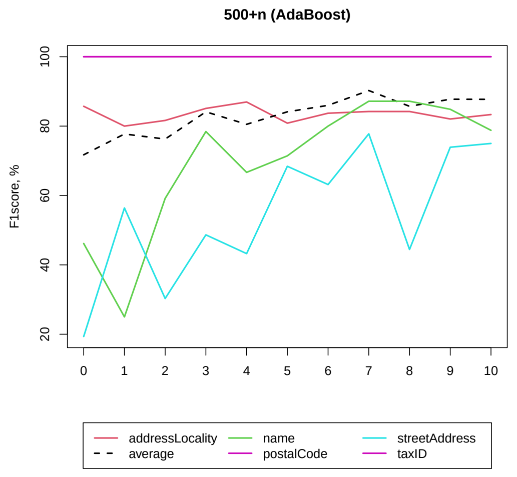


Figure 8. Adaptive test for 500 series data, AdaBoost classifier

further stabilizes the process, as long as the domains remain reasonably closely-related.

The form field datatype, in turn, is discovered by using NER techniques. Here, the significant difference from the standard approaches (see surveys [5, 11]) is the format of input data. Instead of text, the NER system uses an application screenshot, which is processed by an OCR tool. Note that this format of input data forces changes in the standard NER algorithm. In particular, importance of features used in recognition changes. Therefore, a new *local context* feature was introduced. This feature appears to play an important role in the recognition of the context (see [8] for more details).

7. Conclusion and Future Work

In this work, a modification of the F2R approach to self-learning RPA system has been proposed. This modification is based on identifying data type of form fields. The multi-domain NER technique is used for field value recognition. Experiments show that this is a promising direction of the RPA development.

Based on conducted research, it is clear that further automation can be achieved by applying semi-supervised, or unsupervised learning. Hence the next step will be joining the proposed NER system with the actual F2R pipeline, developed in [6].

Note also, that typical RPA systems, by design, are soft real-time applications. Hence, learning and recognition time is important. Tree-based classifiers, especially such as random forest, or boosting methods, are known for large computation times. Hence, it worth to explore other, not tree-based, classifiers.

Finally, as a side effect of our work, a large dataset of Polish language screenshots, with corresponding hOCR and metadata files has been developed. This dataset can be used by other investigators in the field of multi-domain NER.

Acknowledgement

Work of A. Denisiuk, M. Ganzha, P. Sowiński and K. Wasielewska-Michniewska has been realized, in part, within the scope of the “Conducting R&D works for the intelligent OfficeAgent solution” project, which is co-financed by the European Regional Development Fund under the Regional Operational Program of the Mazowieckie Voivodeship 2014-2020.

References

[1] W. M. P. van der Aalst, M. Bichler, and A. Heinzl, “Robotic process automation,” *Business & Information Systems Engineering*, vol. 60, no. 4, pp. 269–272, 2018.

[2] Y. Ketkar and S. Gawade, “Effectiveness of robotic process automation for data mining using uipath,” in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pp. 864–867, IEEE, 2021.

[3] M. Pujar and D. Mundada, “A systematic review web content mining tools and its applications,” *International Journal of Advanced Computer Science and Applications*, vol. 12, 01 2021.

[4] P. Zerbino, A. Stefanini, and D. Aloini, “Process science in action: A literature review on process mining in business management,” *Technological Forecasting and Social Change*, vol. 172, p. 121021, 2021.

[5] S. Agostinelli, A. Marrella, and M. Mecella, “Research challenges for intelligent robotic process automation,”

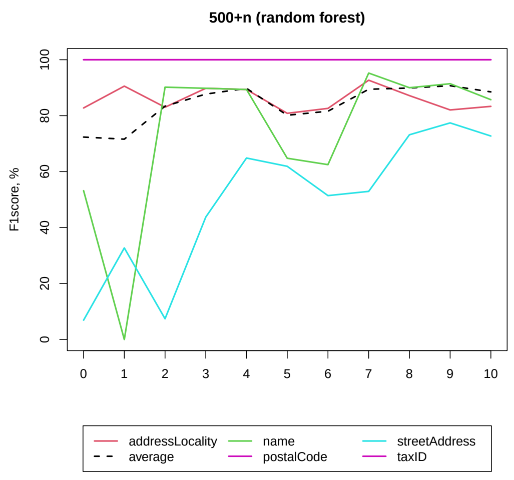


Figure 9. Adaptive test for 500 series data, random forest classifier

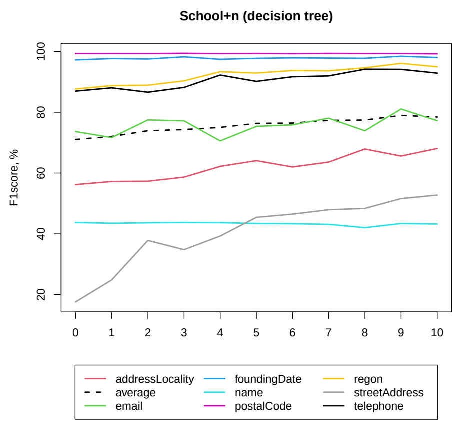


Figure 10. Adaptive test for the School series data, decision tree classifier

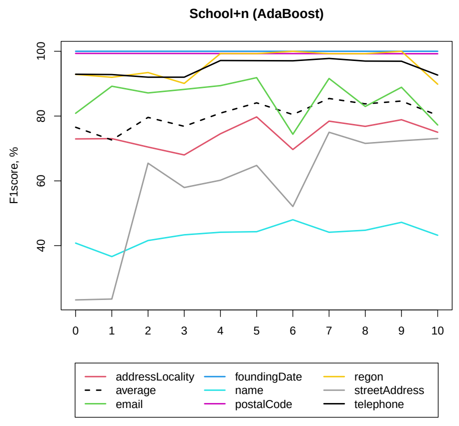


Figure 11. Adaptive test for the School series data, AdaBoost classifier

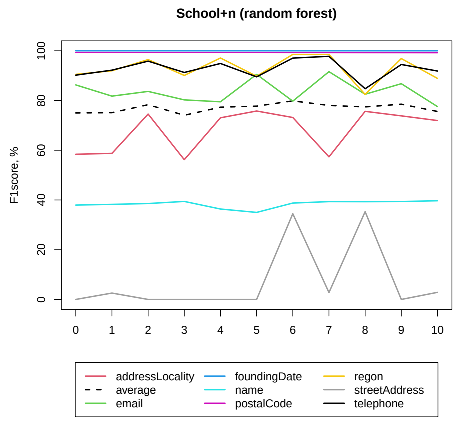


Figure 12. Adaptive test for the School series data, random forest classifier

in *International Conference on Business Process Management*, pp. 12–18, Springer, 2019.

[6] J. Gao, S. J. van Zelst, X. Lu, and W. M. P. van der Aalst, "Automated robotic process automation: A self-learning approach," in *On the Move to Meaningful Internet Systems: OTM 2019 Conferences: Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21–25, 2019, Proceedings*, (Berlin, Heidelberg), p. 95–112, Springer-Verlag, 2019.

[7] C. Tornbohm and R. Dunie, "Market guide for robotic process automation software," *Gartner. com*, 2017.

[8] A. Denisiuk, M. Ganzha, K. Wasielewska-Michniewska, and M. Paprzycki, "Feature extraction for polish language named entities recognition in intelligent office assistant," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, Hawaii International Conference on System Sciences, 2022.

[9] J. Gao, S. J. van Zelst, X. Lu, and W. M. P. van der Aalst, "Automated robotic process automation: A self-learning approach," in *On the Move to Meaningful Internet Systems: OTM 2019 Conferences: Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21–25, 2019, Proceedings*, (Berlin, Heidelberg), p. 95–112, Springer-Verlag, 2019.

[10] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, pp. 3–26, Jan. 2007.

[11] B. Mohit, "Named entity recognition," in *Natural Language Processing of Semitic Languages* (I. Zitouni, ed.), pp. 221–245, Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[12] M. A. Khalid, V. Jijkoun, and M. de Rijke, "The impact of named entity normalization on information retrieval for question answering," in *Advances in Information Retrieval* (C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, and R. W. White, eds.), (Berlin, Heidelberg), pp. 705–710, Springer Berlin Heidelberg, 2008.

[13] R. More, J. Patil, A. Palaskar, and A. Pawde, "Removing named entities to find precedent legal cases," in *Working Notes of FIRE 2019 - Forum for Information Retrieval Evaluation, Kolkata, India, December 12-15, 2019* (P. Mehta, P. Rosso, P. Majumder, and M. Mitra, eds.), vol. 2517 of *CEUR Workshop Proceedings*, pp. 13–18, CEUR-WS.org, 2019.

[14] E. Noguera, A. Toral, F. Llopis, and R. Muńoz, "Reducing question answering input data using named entity recognition," in *Proceedings of the 8th International Conference on Text, Speech and Dialogue*, TSD'05, (Berlin, Heidelberg), p. 428–434, Springer-Verlag, 2005.

[15] H. Kilicoglu, A. B. Abacha, Y. Mrabet, K. Roberts, L. Rodriguez, S. Shooshan, and D. Demner-Fushman, "Annotating named entities in consumer health questions," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)* (N. C. C. Chair), K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, eds.), (Paris, France), European Language Resources Association (ELRA), may 2016.

[16] B. Babych and A. Hartley, "Improving machine translation quality with automatic named entity recognition," in *Proceedings of the 7th International EAMT Workshop on MT and Other Language Technology Tools, Improving MT through Other Language Technology Tools: Resources and Tools for Building MT*, EAMT '03, (USA), p. 1–8, Association for Computational Linguistics, 2003.

[17] R. Sellami, F. Deffaf, F. Sadat, and L. H. Belguith, "Improved statistical machine translation by cross-linguistic projection of named entities recognition and translation," *Computación y Sistemas*, vol. 19, no. 4, 2015.

[18] R. Al-Rfou, V. Kulkarni, B. Perozzi, and S. Skiena, "POLYGLOT-NER: Massive multilingual named entity recognition," *arXiv preprint arXiv:1410.3791v1*, 2014.

[19] S. Chen, Y. Pei, Z. K. 2, and W. Silamu, "Low-resource named entity recognition via the pre-training model," *Symmetry*, vol. 13, no. 786, pp. 3–26, 2021.

[20] A. Pohl, "Knowledge-based named entity recognition in polish," in *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems* (M. Ganzha, L. Maciaszek, and M. Paprzycki, eds.), pp. 145–151, IEEE, 2013.

[21] K. Wróbel and A. Smywinski-Pohl, "Kner: Named entity recognition for polish," in *Proceedings of the PolEval 2018 Workshop*, pp. 101–108, Institute of Computer Science, Polish Academy of Sciences, 2018.

[22] M. Marcińczuk and A. Wawer, "Named entity recognition for polish," *Poznan Studies in Contemporary Linguistics*, vol. 55, no. 2, pp. 239–269, 2019.

[23] A. Wróblewska, T. Stanisławek, B. Prus-Zajączkowski, and Łukasz Garncarek, "Robotic process automation of unstructured data with machine learning," in *Position Papers of the 2018 Federated Conference on Computer Science and Information Systems* (M. Ganzha, L. Maciaszek, and M. Paprzycki, eds.), vol. 16 of *Annals of Computer Science and Information Systems*, pp. 9–16, PTI, 2018.

[24] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

[25] M. Marcińczuk and M. Piasecki, "Named Entity Recognition in the Domain of Polish Stock Exchange Reports," in *Intelligent Information Systems* (M. A. Kłopotek, M. Marciniak, A. Mykowiecka, W. Penczek, and S. T. Wierzchoń, eds.), pp. 127–140, Siedlce: Publishing House of University of Podlasie, 2010.

[26] Z. Liu, Y. Xu, T. Yu, W. Dai, Z. Ji, S. Cahyawijaya, A. Madotto, and P. Fung, "Crossner: Evaluating cross-domain named entity recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 13452–13460, May 2021.

[27] Z. S. Abdallah, M. Carman, and G. Haffari, "Multi-domain evaluation framework for named entity recognition tools," *Computer Speech & Language*, vol. 43, pp. 34–55, 2017.

[28] H. Trittenbach, A. Englhardt, and K. Böhm, "An overview and a benchmark of active learning for outlier detection with one-class classifiers," *Expert Systems with Applications*, vol. 168, p. 114372, 2021.

[29] Z. Liu, G. I. Winata, and P. Fung, "Zero-resource cross-domain named entity recognition," in *Proceedings of the 5th Workshop on Representation Learning for NLP*, (Online), pp. 1–6, Association for Computational Linguistics, July 2020.

[30] C. Jia, X. Liang, and Y. Zhang, "Cross-domain NER using cross-domain language modeling," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 2464–2474, Association for Computational Linguistics, July 2019.

[31] B. Y. Lin and W. Lu, "Neural adaptation layers for cross-domain named entity recognition," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 2012–2022, Association for Computational Linguistics, Oct.-Nov. 2018.

[32] Z. Wang, Y. Qu, L. Chen, J. Shen, W. Zhang, S. Zhang, Y. Gao, G. Gu, K. Chen, and Y. Yu, "Label-aware double transfer learning for cross-specialty medical named entity recognition," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 1–15, Association for Computational Linguistics, June 2018.

[33] J. Wang, M. Kulkarni, and D. Preotiuc-Pietro, "Multi-domain named entity recognition with genre-aware and agnostic inference," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 8476–8488, Association for Computational Linguistics, July 2020.

[34] Google, "Tesseract, version 4.4.1," 2019.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.