

Evaluation of a System for Named Entity Recognition in a Knowledge Management Ecosystem

Philippe Tamla¹^a, Florian Freund¹^b, Matthias Hemmje¹^c and Paul Mc Kevitt²^d

¹University of Hagen, Faculty of Mathematics and Computer Science, 58097 Hagen, Germany

²Ulster University, Faculty of Arts, Humanities and Social Sciences, Londonderry BT48 7JL, North Ireland
fl

Keywords: Named Entity Recognition, Semantic Text Analysis, Natural Language Processing, Knowledge Management System, Data Annotation, User Interface.


Abstract: In this research paper, the evaluation of a new system for machine learning-based Named Entity Recognition is presented. After introducing our approach supporting two fundamental tasks for training Named Entity Recognition models using machine learning (data cleanup and data annotation), our features, prototype and evaluation methodologies are described. Also, the results of our performed quantitative and qualitative experiments validating our approach and user interface are shown. Finally, our evaluation results are discussed to derive challenges for future work.


1 INTRODUCTION


Named Entity Recognition (NER) is a method of Information Extraction that aims at recognizing Named Entities (NEs) in a Natural Language (NL) full text (Konkol, 2015). NEs are, “a word or sequence of words that [are] used to refer to something of interest in a particular application” (Croft et al., 2010). A typical NE can refer to a real-world object such as organization, person, or location. The identification and classification of NEs is known as *Named Entity Recognition and Classification (NERC)* (Nadeau and Sekine, 2007). NER, being one of the most fundamental tasks of Natural Language Processing (NLP) (Jiang, 2012), is generally applied in systems involving the semantic analysis of textual documents and has found many applications in *Information Discovery (ID)* (Dias et al., 2020) and *Information Retrieval (IR)* (Mahalakshmi, 2015). Techniques for NER include hand-coded, based on human crafted rules or dictionary lookups, and Machine Learning (ML) (Nadeau and Sekine, 2007; Palshikar, 2013). Our research particularly focuses on ML-based methods for NER as they have greatly evolved in recent years and have been widely used in combination with


IR systems for accessing and retrieving various textual documents in domains like Software Engineering (Liu et al., 2018), Social Media (Onal and Karagoz, 2015), and Medical Research (Nawroth et al., 2018). “*Machine Learning (ML) can be described as computational approaches in which a specific function is not programmed by a human being but by another program, called learner, that uses existing example data to generate programs called models, The process of creating models is called training*” (Swoboda, 2021). ML is an important method for performing NER. However, applying it generally requires domain specific knowledge in addition to software development skills (Zhang and Tsai, 2003; Swoboda, 2021).

According to (Swoboda, 2021), training a ML model for NER appears to be impossible without linguistic skills and manually provided knowledge resources. In the domain of NER, this may require providing gold labels or defining manual rules to extract NEs in the target domain (Christian, 2021). People with general software development knowledge may not have sufficient domain knowledge to provide such labels or rules. The reason is that gold standard datasets required for model training are a continuous team effort involving domain experts and information modelers (Swoboda, 2021). Analogously, domain experts may also lack the software engineering skills required for ML. However, such skills are a prerequisite for ML as writing program codes (coding) to organize one’s domain data and being able

^a <https://orcid.org/0000-0002-0786-4253>

^b <https://orcid.org/0000-0002-7344-6869>

^c <https://orcid.org/0000-0001-8293-2802>

^d <https://orcid.org/0000-0001-9715-1590>

to derive useful insights and patterns is often needed (MSV, 2020). This may be a bottleneck and a very difficult task for domain experts without software engineering background. We have recently designed and developed a system for ML-based NER using User Centered Design (UCD) methodology (Norman and Draper, 1986). This included the analysis of remaining challenges in NER, covering aspects of domain, language, and entity types, resulting in the definition of use cases, component models, and overall architecture of the system. We addressed two major tasks of NER, data cleanup and data annotation (Tamla, 2022). An initial prototype of this system, called Stanford Named Entity Recognition and Classification (SNERC), was integrated in an innovative ecosystem for knowledge management and creation called Knowledge Management - Ecosystem Portal (KM-EP) (Salman et al., 2017). The goal of SNERC was to enable the training and customization of NER models for both experts and novice users (newbies) and support knowledge management and retrieval in KM-EP.

In this paper, we present the results of various quantitative and qualitative evaluations demonstrating the effectiveness, generality and adaptability of the approach and implemented prototype. The paper is organized as follows. We review related works (section 2), summarize the implementation of SNERC, (section 3). Next, we discuss our evaluation findings (section 4) and conclude, discussing future work (section 5).

2 STATE OF THE ART IN SCIENCE AND TECHNOLOGY

2.1 Named Entity Recognition

Techniques for NER vary from hand-coded, over ML-based, to hybrid methods. But, ML-based methods are more efficient on Web content, because they include statistical models that can automatically recognize and classify named entities from very large and heterogeneous content on the Web. Existing NER techniques based on ML are often classified into Supervised Learning (SL), Unsupervised Learning, and Semi-supervised Learning. SL (Thenmalar et al., 2015) is currently the dominant technique for addressing NER and can achieve promising performance if enough high-quality training data is available. These labeled data created by domain experts are called, *gold standard* (Kang et al., 2012). Conditional Random Fields (CRF) is a SL method based on statistical algorithms (Zhang et al., 2020). It represents, “*the*

state of the art in sequence modeling and has also been very effective at NER task” (Mao et al., 2007).

Training a NER model using ML is often very challenging. It requires, besides substantial programming knowledge, dealing with different technologies and pipelines for text analysis, NLP, ML and rule-based operations (Konkol, 2015). Errors in the initial stages of the pipeline can have snowballing effects on the pipeline’s end performance. A list of remaining challenges in NER was already identified and discussed in our recent research (Tamla, 2022). These challenges cover aspects of language, domain, addressing the portability and adaptiveness of NER systems, entity types and gold-standard design and creation, ML features selection, and quality of NER preliminary steps, e.g., data cleanup, data annotation. Following these challenges, it is necessary to efficiently design, develop, and customize all necessary preliminary steps in a NER pipeline in order to reduce the effort of training new NER models whilst optimizing the performance of the whole system.

2.2 Knowledge Management System

A Knowledge Management System (KMS) is an information system for organizing and managing knowledge such that it can be easily accessible by potential knowledge producers and consumers (Natek et al., 2016). Systems for Knowledge Management (KM) are often introduced to support the creation, storage, retrieval, transfer, and application of Knowledge (Alavi and Leidner, 2001). The architecture of KMSs is described in (Gronau, 2002) and generally includes a Taxonomy Management System (TMS) for organizing information and documents, a content management system, and a search engine for document indexing. The European research project, Realizing and Applied Gaming Ecosystem (RAGE) is an innovative KMS and central online portal for accessing and retrieving reusable software components and other related textual documents from the Web, such as research publications, source code repositories, issues, and online discussions. RAGE is used to support software reuse in the domain of applied gaming. Applied games (AG) or serious games (SG) aim at training, educating and motivating players, instead of pure entertainment (David R. and Sandra L., 2005). The RAGE system was developed as part of KM-EP. This ecosystem uses various REST APIs to enable the integration and import of text documents from various social networks like Stack Exchange dialogs (“Hot questions”), or GitHub (“Build software better”). It also includes an own developed TMS (Vu, 2020) enabling the classifying of text documents into

taxonomies. KM-EP users can easily import various Web content like online discussions from, e.g., the Stack Overflow social site, describe them with further meta information, classify them “manually” using an integrated taxonomy management system, and then finally retrieve useful information with a faceted search that enables drilling down large set of documents. To enable semantic text analysis and automatic document classification of Web content, a new NER system based on the Stanford CoreNLP (called SNERC) was recently developed and integrated within KM-EP (Tamlal, 2022). SNERC facilitates the design, development, customization, and management of domain-specific models and can be used by both NER experts and novice users to develop new NER models.

3 IMPLEMENTATION OF SNERC

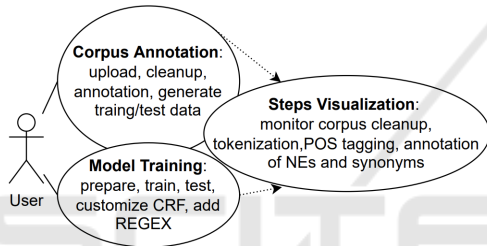


Figure 1: SNERC Use Cases for NER.

Our system, SNERC, was experimentally developed using the well-known NLP framework of Stanford university (Manning et al., 2014). This framework supports most of the common NLP tasks used for NER. We defined several use cases for designing, developing and customizing ML-based NER models as shown in Figure 1. Some GUI components of SNERC with different tabs are presented in Figure 2. Using SNERC, users can manage basic parameters and configuration steps, which we call NER model definition, that are used to select, execute, and customize various preliminary steps for training a NER model using ML. For instance, users can upload a domain corpus (data dump) and select various options for cleaning it up. They can also execute the automatic annotation of this domain corpus using the standard BIO format for token annotation (Arellano et al., 2015). For the system to annotate the corpus, users have to define the original name of each NE (like Java Script) and, if necessary, inform the system about the NE synonyms and name variations of this NE (like js, JavaScript, Javascript, Js). Also, the domain-specific NE category (or label) that should be used to annotate this NE and its synonyms, and name variations, must be defined. After the data is annotated, users can specify

how to split them to generate training and testing data automatically. Domain experts can use this feature to reduce the effort in creating their training and testing data. Also, they have the possibility to review and update the annotated data to avoid the problem of overfitting which can lead to reduced performance of the trained model (Géron, 2019). Another feature available in SNERC is the definition of global and local features. Various CRF parameters can be customized for model training. One or more gazetteers can be added to the NER pipeline to optimize the training datasets. Regular Expressions (REs) can be also introduced to the NER pipeline to detect complex NEs using handcrafted rules. All these features are available in the NER pipeline and can be executed to train a new model using ML. The execution of all NER preliminary steps before model training, e.g., data cleanup and annotation, can be monitored using logs visualization (of automatically cleaned data and annotated tokens), which helps to check and optimize the quality of the model at an early stage. The standard evaluation metrics, Precision (P), Recall (R), and F-Score (F_1), can be also visualized after each model training to check the quality of each new model.

Based on our use cases, we have defined three main components related to them as seen in Figure 3. *NER Model Definition Manager* manages all the necessary definitions and parameters for model training using ML. Its information model includes three main classes. The first two, *Named Entity Category* and *Named Entity*, hold information about the domain-specific NEs names and categories. The third class, *NER Model Definition*, is used to store data in the KM-EP database like model name, text corpus, gazette lists, and regex rules. We use the Stanford RegexNER API to construct and store complex rules, as they can easily be combined with already trained models. The *NER Model Manager* component consists of the single *NER Model* class which is used for storing trained NER models into the KM-EP filesystem so that they can be used by other systems. If a model was created using a *NER Model Definition*, users can update the generated testing and training data within the *NER Model Manager* to get better P, R and F_1 scores. Also, regular expression rules created using the Stanford Regex API can be edited and updated. It is also possible to upload a Stanford CoreNLP NER model that was trained with another system and use it in KM-EP.

The *NER Model Trainer* component is used to execute the training of a NER model using ML. This includes the automatic annotation of the domain text corpus based on the previously defined NE categories, NE names and synonyms. SNERC is also able to au-

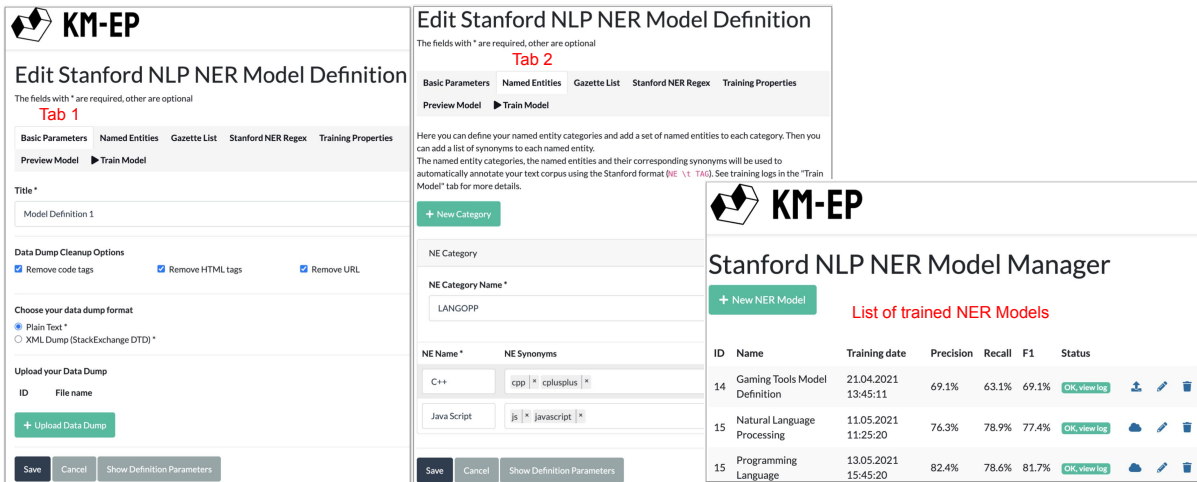


Figure 2: SNERC GUI components.

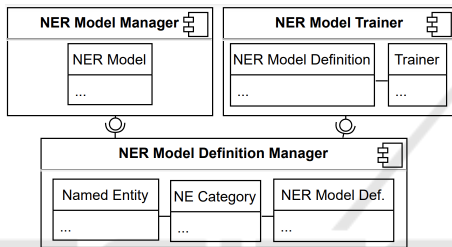


Figure 3: SNERC components for NER.

automatically split the annotated text corpus into testing and training data. However, the testing data, needs to be reviewed by a human expert and uploaded again to avoid overfitting, and thus a realistic calculation of P, R and F₁ scores. When this is done, the *NER Model Trainer* component is used to execute the task for training a NER model using jobs and Stanford CoreNLP. Its information model defines two classes. The first class, *NER Model Definition*, holds the same basic information provided in the *NER Model Definition Manager* component which is needed for model training. The second class, *Trainer*, is used to control the training process whether the model is in the prepared or trained state. It is also used to store the reference to each trained model in the KM-EP database. Further detail on all components of SNERC can be found in (Tamla, 2022).

4 EVALUATION OF SNERC

To validate our research, we need to prove the feasibility, usability, usefulness and efficiency of our approach and implemented SNERC prototype. Hence, this work aims at evaluating various aspects of our solution using qualitative and quantitative methods.

There are different evaluation methodologies used for software applications (Kitchenham, 1996). A well-known method to measure the efficiency or effectivity of a system is a *Controlled Experiment*, a qualitative evaluation methodology relying on hypothesis testing (Helander, 2014). Controlled experiments have clear and precise goals where all participants have a fix set of tasks to complete. The evaluation results are then assessed to check whether or not the previously defined research goals were reached. Participants in a controlled experiment generally have various profiles and can thus be clustered into different groups of users based on their background or expertise. The well-known metrics P, R, and F₁ (Powers, 2020) can also be used to evaluate the results, if the tasks require the participants to choose items out of a set of possible solutions. When a prototype solution is available, it becomes easy to compare the results of each group of users. Let *s*₁ be the set of correctly chosen items (hits), *s*₂ the set of prototype solution items that were not chosen (misses), and *s*₃ the set of incorrectly chosen items (false alarms). The standard metrics P, R, F₁ are calculated as follows:

$$P = \frac{s_1}{s_1 \cup s_2}, R = \frac{s_1}{s_1 \cup s_3}, F_1 = 2 \times \frac{P \times R}{P + R} \quad (1)$$

The initial evaluation (section 4.2) is a controlled experiment for assessing the efficiency of our approach. It evaluates our features supporting two fundamental tasks of NER, “data cleanup” and “data annotation”. Our two user groups including NER experts and novice users (newbies) are taking part of this evaluation. We use prototype solutions and the standard metrics (P, R, and F₁) to assess our supported features while comparing the results of our user groups.

In the second evaluation (section 4.3), a walk-through approach (including surveys and questionnaires) (Finstad, 2010; Lund, 2001; Thielsch and Stegemöller, 2014) is used as the evaluation methodology to validate the GUI features of our prototype. Following a predefined tutorial, NER experts and newbies are asked to train a new NER model. Our goal is to evaluate and check the feasibility, usability and usefulness of SNERC GUI components in supporting the tasks of NER in a real-world environment.

4.1 Evaluation Setup and Pretesting

The experiments presented were guided by a tutorial document. This document consisted of sections for assessing our NER approach and implemented prototype. Each of these sections included an introduction with the structure and goal of the experiment, examples highlighting our approaches and features, and a doing phase with tasks to be completed by the participants. Before our participants commenced the experiments, they were asked to complete an initial survey questionnaire, which helped us to collect information about their background and experience in programming languages, ML, NER. Based on this information, we were able to classify our participants into the categories NER experts and newbies.

We created an evaluation document, including tutorial, guidelines, and description of tasks, and sent it to a NER expert to perform a pretesting of our approaches and provide feedback. This NER expert completed his Ph.D. research in the domain of emerging Named Entity (eNE) and has more than 5 years experience in the development of ML projects. The result of the pretesting did not lead to significant improvements in the evaluation guidelines and tutorial document. After the review of the evaluation document, we sent its final version to all the participants together with credentials, usernames and passwords, for accessing the KM-EP portal.

For our evaluations, we focus on two groups of users having different backgrounds and educations: Newbies (group 1) and Experts (group 2). Newbies are normal users, who might or might not have knowledge about NER. Experts are defined as skillful people having experience in ML-based NER. Our participants were chosen from the domain of software engineering and data science, which also includes ML experts. Our group of newbies consists of 4 software engineers, who have a master’s degree in computer science. These users do not have experience in NER and ML. The group of experts consists of 4 users (2 Ph.D.s and 2 Ph.D. candidates), all having at least 3 years experience in using and developing ML and

NLP systems, including NER. The job of all these users was to perform all evaluations.

4.2 Qualitative Evaluation of the NER Approach

The concept of this evaluation is to let newbies and experts do the same, completing various preliminary tasks for training a NER model, then compare the results to validate if newbies can achieve similar or even better results than experts.

4.2.1 Evaluation Setup

To execute this evaluation, a document, can be downloaded from Google Drive¹, including tutorial, guidelines, and description of tasks was presented to the participants. This document also described the domain corpus and labels we used to execute the experiment. Finally, the document included questions for gathering direct feedback from all participants about our approach supporting the NER preliminary steps of, “data cleanup” and “data annotation”. The domain corpus that is used in this experiment contains a list of discussion posts about programming languages found in the Stack Overflow social network. Such discussion posts generally contain various types of NEs, related to official names of programming languages (like “Java Script”), their synonyms (like “js”), and name variations (like “javascript”, “JavaScript”, “Javascrpt”) as it was initially analyzed in our previous preliminary study (Tamla et al., 2019).

Table 1 shows a set of documents from our selected domain corpus. Column 1 shows some sentences of each discussion post. We can identify HTML and code snippets in some of these discussions. Column 2 shows the NEs identified in the documents together with their synonyms and name variations used by online users (like Java Script, Javascript, JS). Column 3 shows a reference ID to each Stack Overflow post. As we are dealing with programming languages, our selected domain labels, or NE category names, are defined based on common programming paradigms. For the sake of simplicity, we only consider five programming paradigms with a respective label, as they are very popular in programming: Declarative (*LANGDECL*), Functional (*LANGFUNC*), Imperative (*LANGIMP*), Object Oriented (*LANGOOP*) and Procedural (*LANGPROC*) programming languages.

¹https://drive.google.com/file/d/1FCKuX.GY1Xm_Rxe8IkAR1oN15aDXcRSh

Table 1: Subset of Documents about Serious Games (SG)-related Posts in Stack Overflow.

Document	Identified NEs and synonyms	Stack ID
What is the difference between String and string in C#?	C#	#7074
How can I decrypt an “encrypted string with java” in c sharp?	C Sharp	#22742097
Is Java “pass-by-reference” or “pass-by-value”?	Java	#40480
javascript code to change the background color on clicking more buttons	javascript	#67365586
Check out <code>Unobtrusive JavaScript</code> and Progressive enhancement (both Wikipedia).	JavaScript	#134845
Are there any coding standards for JavaScript? <code><code>...if ... else ...</code></code>	JavaScript	#211795
Parse an HTML string with JS	JS	#10585029
javascript code to change the background color on clicking more buttons	javascript	#67365586
Finding duplicate values in a SQL table	SQL	#2594829
Learning COBOL Without Access to Mainframe	COBOL	#4433165

4.2.2 Procedure

In the first task of this experiment (Task 1.1), users are asked to apply a set of options to clean up our selected domain corpus consisting of Stack Overflow discussion posts about programming languages. SNERC supports a variety of options for cleaning up data from a Web document, which might not be relevant for training a model in the target domain. For instance, if the initial domain corpus is a HTML document, users may choose to remove all the HTML markups, images, code snippets, and links, whilst keeping only the textual information containing the relevant NEs in the target domain. The cleaned-up document can then be easily annotated to create the gold standard for model training. After explaining our cleanup features and providing concrete examples how to apply them on the example documents, a doing-phase including multiple clean up tasks was presented to the users to complete them. Users were asked to choose features to clean up documents about programming languages, which we also collected from Stack Overflow. As we wanted to compare the qualitative performance of newbies with the performance of experts, both groups performed the same task of “data cleanup”. A prototype solution for this task was also defined (see Table 2). Based on these prototype solutions and the answers provided by the participants, we could compare all the results of newbies and experts using the standard metrics P, R, and F₁.

The second task of this experiment (Task 1.2) was related to “data annotation”, which is another fundamental step for model training in NER. Our approach provides features to automatically annotate a domain corpus. It relies on *Computational Natural Language Learning (CoNLL)* (Sang and De Meulder, 2003), a widely used data format for generating testing and

training data (Pinto et al., 2016). CoNLL uses the standard BIO format for token annotation in NER. BIO (short for Beginning, Inside, Outside) is the de-facto standard format for tagging tokens in many NLP frameworks (Arellano et al., 2015). After explaining with appropriate examples, the concept of token annotation using BIO, users were asked to annotate a set of documents including different types of NEs, such as, single-word NEs (like “Java”, “HTML”), multi-word NEs (like “C sharp”, “Java Script”), and extended NEs (like “C sharp 5.0”). We should note that some of the documents included the official names of programming languages, with their synonyms and name variations used in the crowd (like “csharp”, “c#” or “c sharp”). Thus, users had to provide the appropriate BIO annotations to label them. For comparing the results of newbies and experts, a prototype solution (see Table 3) was also defined, which enabled us to compare all the results using the standard metrics P, R, and F₁ as in the first task.

Finally, we introduced the following three open questions to collect direct feedback about our approach supporting “data cleanup” and “data annotation” in NER:

- Are the offered annotation features for the creation of training and testing data sufficient?
- Do you use a data format which is different from the provided CoNLL 2003 format? If yes, which one?
- What are you missing?

4.2.3 Evaluation Results

Table 4 shows the calculated metrics P, R, and F₁ for each user completing Task 1.1. All experts performed with a Precision of 100% and a Recall greater than

Table 2: Prototype Solution for Task 1.1.

Document	Remove Code Tags	Remove HTML	Remove URL
For-each over an array in <code>JavaScript</code>	No	Yes	Yes
Are there any coding standards for JavaScript? <code><code>...if ... else ...</code></code>	Yes	No	No
Parse an HTML string with <code><bold>JS</bold></code>	No	Yes	No
Remove the <code><style="text-color:#ff0000">?</style></code> at the end of this C sharp code <code><code>...player.run();?</code></code> to solve your compilation error?	Yes	Yes	No

Table 3: Prototype Solution for Task 1.2.

Token	NE Category Name	NE type
This	O	
limitation	O	
found	O	
in	O	
your	O	
Java	B-LANGFUNC	multi-word
Script	I-LANGFUNC	
code	O	
is	O	
also	O	
found	O	
in	O	
Java	B-LANGOOP	single-word
and	O	
C	B-LANGOOP	extended, synonym of c#
sharp	I-LANGOOP	
5.0	I-LANGOOP	
,	O	
C#	B-LANGOOP	extended
5.0	I-LANGOOP	
and	O	
the	O	
latest	O	
COBOL	B-LANGPROC	single-word
Version	O	

66%, while all newbies obtained a Recall of 100% and a Precision greater than 85%. The best result is from participant Newbie3 with a Precision of 100% and a Recall of 100%, which means that he or she performed even better than all the experts in this task.

The average P, R, and F₁ scores for each user group is displayed in Table 5. It shows that while newbies have performed with the highest Recall of 100%, experts have performed with the highest Precision of 100%. Considering the F₁ score, we can say that newbies (having a F₁ of 92.58%) have better performed compared to experts, having a F₁ of 88.18%, in this task.

Table 4: Precision, Recall and F₁ of Task 1.1.

Participant	Precision (%)	Recall (%)	F ₁ (%)
Newbie1	85.71	100.00	92.31
Newbie2	75.00	100.00	85.71
Newbie3	100.00	100.00	100.00
Newbie4	85.71	100.00	92.31
Expert1	100.00	83.33	90.91
Expert2	100.00	66.67	80.00
Expert3	100.00	83.33	90.91
Expert4	100.00	83.33	90.91

Table 5: Average Precision, Recall and F₁ for Task 1.1 for each User Group.

	Precision (%)	Recall (%)	F ₁ (%)
Newbies	86.61	100.00	92.58
Experts	100.00	79.17	88.18

Table 6 displays the scores for each user completing Task 1.2. It shows that 3 experts and 1 newbie obtained the highest Precision and Recall of 100%. Only one expert had a lower Precision of 72.73% and a lower Recall of 88.89%. Newbie2 and Newbie4 performed poorly which had a negative effect on the overall score of newbies compared to experts.

Table 6: Precision, Recall and F₁ for Task 1.2.

Participant	Precision (%)	Recall (%)	F ₁ (%)
Newbie1	100.00	100.00	100.00
Newbie2	83.33	55.56	66.67
Newbie3	100.00	77.78	87.50
Newbie4	88.89	88.89	88.89
Expert1	72.73	88.89	80.00
Expert2	100.00	100.00	100.00
Expert3	100.00	100.00	100.00
Expert4	100.00	100.00	100.00

The average scores for each user group are displayed in Table 7. We can see that experts have performed with a 10% higher F₁, concluding that experts are better in this task.

Table 7: Average Precision, Recall and F₁ for Task 1.2 for each User Group.

	Precision (%)	Recall (%)	F ₁ (%)
Newbies	93.06	80.56	85.77
Experts	93.18	97.22	95.00

The answers provided in the open questions revealed that our approach supporting “data cleanup” and “data annotation” is sufficient for most of our participants. However, two experts found that the CoNLL data format and BIO labels used for token annotation in NER is not enough. From their perspective, supporting other data formats such as JSON, XML, or making use of a standard library for NLP related tasks (like spaCy²) could be helpful to make our application more flexible. The next section evaluates our SNERC GUI components enabling NER model training in KM-EP.

4.3 Quantitative Evaluation of SNERC

This experiment based on walkthrough methodology aims at validating the feasibility, usability, and efficiency of our implemented SNERC prototype by gathering feedback on its GUI components, features and functionality. SNERC is a sub-module of KM-EP and its features supporting NER were already summarized in section 3. Also, the GUI components of SNERC were already shown in Figure 2. In this experiment, we want to validate if our participants can use our system to train new NER models in their respective domains by collecting various quantitative factors of our prototype, such as how good the implemented components are. The stakeholders of KM-EP are different user groups and communities who will be affected by and will be using the services and possibilities of the system developed and provided during the project. As the participants of the first evaluation were selected from the software engineering, ML, and data science fields, those are also valid stakeholders of KM-EP and have hence been selected for this evaluation. KM-EP users, including NER experts or newbies, may want to train a new NER model to extract NEs in their particular domains. To complete this experiment, a document, which can be downloaded from Google Drive³, including tutorial, guidelines, and description of tasks was prepared and provided to the participants.

²<https://www.spacy.io>

³https://drive.google.com/file/d/1FCkuX_GY1Xm-Rxe8IkAR1oN15aDXcRSh

4.3.1 Evaluation Setup

A walkthrough requires setting up the environment and preparing various data for the evaluation. To walk through our NER features, data was cloned and provided to all the participants’ KM-EP instances. We created 10 user accounts (including usernames, passwords), 8 for the participants and 2 for the admins managing all the evaluation data of this walkthrough. The usernames and passwords were sent to the participants via E-Mail with a link to download the document tutorial including instructions to execute this walkthrough. As in the first evaluation, our evaluation document included instructions about our SNERC features used for training a new NER model. It also introduced the participants to the domain corpus used to execute this walkthrough. Finally, this document included questions for gathering information about our features supporting NER.

4.3.2 Procedure

Our evaluation includes a combination of questions from standardized questionnaires UMUX (Finstad, 2010), USE (Lund, 2001) and Münsteraner Fragebogen zur Evaluation – Zusatzmodul Diskussion (Thielsch and Stegemöller, 2014) and open questions targeting the functionality of SNERC. Our survey largely covered questions to be answered on a 7-point-likert-scale from strongly disagree (1) to strongly agree (7). Also, we added three open questions at the end of the survey to collect information concerning the improvement of the prototype and the quality of the tutorial. Overall, our questions were divided into four categories: Usability, Usefulness, User Interface and NER features of SNERC. Data on the evaluation categories was collected as follows:

Usability: We used the UMUX questionnaire (Finstad, 2010) for a general assessment of the usability with 4 items as shown in Table 8:

Table 8: Question about Usability.

Id	Text
Usab1	This tool’s capabilities meet my requirements.
Usab2	Using this tool is a frustrating experience.
Usab3	This tool is easy to use.
Usab4	I have to spend too much time correcting things with this tool.

Usefulness: For evaluating the usefulness of SNERC, one scale with 8 items from USE (Lund, 2001) was

adopted. Responses were provided on the same 7-point rating scale as for the UMUX (Table 9).

Table 9: Question about Usefulness.

Id	Text
Usef1	It helps me be more effective.
Usef2	It helps me be more productive.
Usef3	It is useful.
Usef4	It gives me more control over the activities in my work.
Usef5	It makes the things I want to accomplish easier to get done.
Usef6	It saves me time when I use it.
Usef7	It meets my needs.
Usef8	It does everything I would expect it to do.

User Interface: This category aims at gathering information about how fast SNERC works and how the user interface feels. We included questions concerning the buttons, icons, images, and texts in the questionnaire. To ensure consistency with the other scales/categories, we included 5 items affecting the user interface to be answered on a 7-point rating scale (Table 10).

Table 10: Questions about User Interface.

Id	Text
Ui1	All SNERC-components work fast.
Ui2	The user interface feels good.
Ui3	Buttons, images, and texts are in the right position.
Ui4	Enough information and explanations are presented.
Ui5	The images and icons look good.

NER Features of SNERC: To evaluate the NER components of SNERC, 5 items answered on a 7-point-rating scale dealt with the flexibility and visualization of the NER features available in SNERC (Table 11).

Besides the questions with answers on a 7-point rating scale, 3 open questions were asked concerning improvements of SNERC and the tutorial (Table 12). The answers were interpreted separately.

To get familiar with SNERC, specific tasks were prepared, to make sure that all the participants work with SNERC consistently. Completing this walk-through experiment took approximately around 1.5 hours, up to 2 hours for unexperienced users, including the time to go through the provided tutorial and complete the survey.

Table 11: Questions about NE Features of SNERC.

Id	Text
NERf1	I spent a lot of time testing the NER Model Definition Manager.
NERf2	The NER Model Definition Manager guided me through the process of training a custom NER model.
NERf3	I need more flexibility for customizing the training pipeline.
NERf4	The CoreNLP based pipeline is a sufficient basis.
NERf5	The "Preview" feature is helpful to get short round trips, while customizing parameters.

Table 12: Questions about Improvement.

Id	Text
Imp1	Which functions or aspects are lacking in the current solution in your opinion?
Imp2	Do you have ideas for improvements or alterations of SNERC?
Imp3	Do you think the support materials (manuals, tutorials, etc.) are sufficient? If not, what is missing?

4.3.3 Evaluation Results

The results of this chapter will be used to demonstrate whether our prototype can be used to enable NER in a real-world scenario. Also, these results can be used as a starting point for further R&D work.

Results related to Demographics: 8 users (including 4 experts and 4 newbies) completed both experiments. All respondents hold at least a master degree and had worked at least for 5 years as professional software engineers or data scientists. One expert has worked for 10 years as a data scientist and another one has completed a Ph.D. in the domain of eNE. All 4 experts have at least 3 years of experience in developing and applying ML-NER systems.

Results Related to Usability: All participants had to fill out a questionnaire regarding the usability of SNERC, which included 4 items (Usab1 to Usab4). The responses on these items are presented in Table 13. Figure 4 depicts the average values of all participants providing answers to the usability questions. It shows that in average 62.5% of the participants disagreed to have a frustrating experience with SNERC (Usab2), while 50% agreed that SNERC is easy to use (Usab3). However, we can also observe that there is quite a high spread in the answers for

Table 13: Responses (%) for the Usability Questionnaire.

Scale	Usab1 (%)	Usab2 (%)	Usab3 (%)	Usab4 (%)
Strongly agree	12.5	0.0	0.0	0.0
agree	25.0	0.0	25.0	0.0
somewhat agree	25.0	0.0	50.0	25.0
neither agree nor disagree	12.5	12.5	12.5	12.5
somewhat disagree	12.5	12.5	0.0	50.0
disagree	12.5	62.50	12.5	12.5
strongly disagree	0.0	12.5	0.0	12.5

Usab4, which might reflect the demographics of the participants as they have different background knowledge about methods and tools for NER and ML.

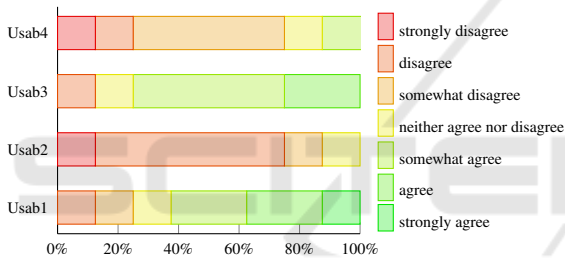


Figure 4: Average Values of the Answers of the Usability Questionnaire.

Figure 5 shows a bar plot with the usability answers divided by our user groups (newbies and experts). We can see that SNERC is usable and easy to use for both user groups as they selected on average “agree” for the two statements “This tool’s capabilities meet my requirements” (Usab1) and “This tool is easy to use” (Usab3). From this diagram we can also see that newbies seem to need more time to get familiar with this tool than NER experts. We conclude this from their answers provided in Usab2 and Usab4.

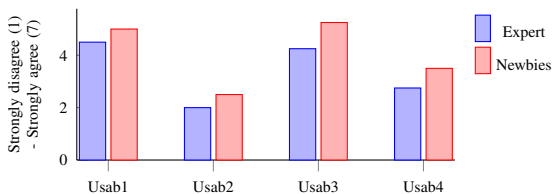


Figure 5: Average Values of the Usability Answers divided by User Groups.

Results Related to Usefulness: All participants had to fill out a questionnaire regarding the usefulness of SNERC, which included 8 items (Usef1 to Usef8). The responses of these items are presented in Table 14. Figure 6 shows the answers to the questions related to the usefulness in a bar plot. We can see that most answers are positive which indicates that our tool is useful to enable NER. There is one user who gave a negative feedback (disagree), but unfortunately no additional comment was provided. Hence, it is not clear which feature the user was missing.

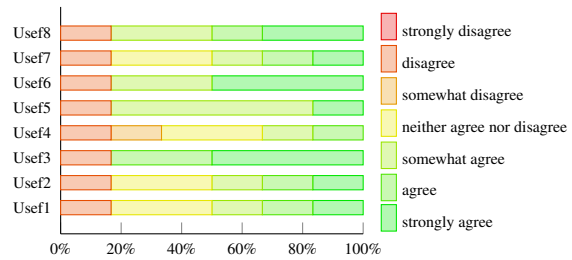


Figure 6: Average Values of the Answers of the Usefulness Questionnaire.

Figure 7 shows the average answers for our two user groups to the usefulness questions. The most positive answers are for the question “It is useful” (Usef3), 5.75 for experts and 5.5 for newbies, and “It saves me time when I use it” (Usef6), 5.75 for experts and 5.25 for newbies. As we can see, there is no significant difference between the two user groups. We can only see slightly smaller scores for the group of newbies (especially Usef1, Usef2, Usef7) which might be related to their lack of experience in the domain of NER. This result shows that our tool is useful for both, NER experts and newbies.

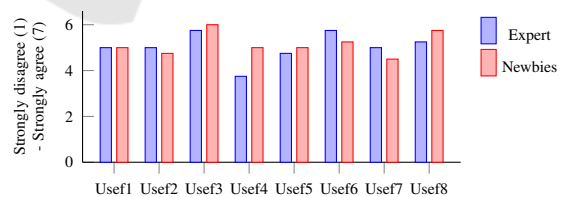


Figure 7: Average Values of the Usefulness Answers divided by User Groups.

Results Related to User Interface: Our participants had to fill out a questionnaire regarding the quality of the user interface of SNERC. This questionnaire included five items (Ui1 to Ui5). The responses of these items are presented in Table 15.

Figure 8 depicts the average values of all participants’ answers to the user interface questionnaire. It shows that all participants gave only positive answers to these questions as they agree that the user interface

Table 14: Responses (%) for the Usefulness Questionnaire.

Scale	Usef1 (%)	Usef2 (%)	Usef3 (%)	Usef4 (%)	Usef5 (%)	Usef6 (%)	Usef7 (%)	Usef8 (%)
strongly agree	12.5	12.5	37.5	0.0	25.0	37.5	12.5	25.0
agree	37.5	25.0	50.0	25.0	0.0	12.5	12.5	37.5
somewhat agree	12.5	25.0	0.0	25.0	62.5	37.5	37.5	25.0
neither agree nor disagree	25.0	25.0	0.0	25.0	0.0	0.0	25.0	0.0
somewhat disagree	0.0	0.0	0.0	12.5	0.0	0.0	0.0	0.0
disagree	12.5	12.5	12.5	12.5	12.5	12.5	12.5	12.5
strongly disagree	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 15: Responses (%) for the User Interface Questionnaire.

Scale	Ui1 (%)	Ui2 (%)	Ui3 (%)	Ui4 (%)	Ui5 (%)
strongly agree	37.5	12.5	25.0	0.0	37.5
agree	37.5	62.5	37.5	37.5	37.5
somewhat agree	12.5	25.0	12.5	62.5	25.0
neither agree nor disagree	12.5	0.0	25.0	0.0	0.0
somewhat disagree	0.0	0.0	0.0	0.0	0.0
disagree	0.0	0.0	0.0	0.0	0.0
strongly disagree	0.0	0.0	0.0	0.0	0.0

of SNERC feels good (Ui2), has enough information are provided (Ui4), and works fast (Ui1).

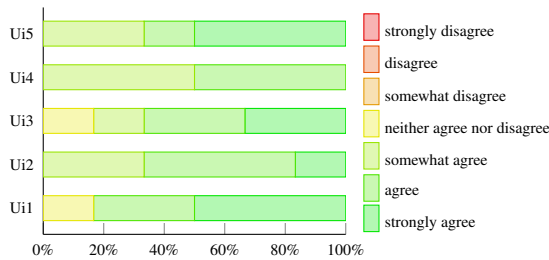


Figure 8: Average Values of the Answers of the User Interface Questionnaire.

Figure 9 shows the average answers divided by user groups. We can see no significant differences between newbies and experts which indicate that they are all satisfied with our user interface.

Results Related to NER Features of SNERC: Our participants had to fill out a questionnaire regarding the NER features of SNERC, which included 5 items (NERf1 to NERf5). The responses of these items are presented in Table 16.

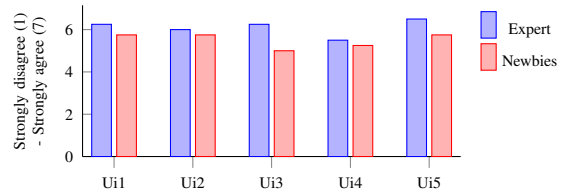


Figure 9: Average Values of the User Interface Answers divided by User Groups.

Figure 10 depicts the average values of all participants' answers for the NER features category. It shows 37.5% of the participants agree that the “NER Model Definition Manager” of SNERC guided them through the process of training a custom NER model (NERf2) which indicates that our tool is a valid tool to complete this task. Also, 50% of the participants are happy with the preview feature (NERf5) as it helps them to check the results and quality of their trained model after customizing various parameters. While SNERC seems to be very useful to support NER, half of the participants are neutral about its flexibility and customization capability. The same portion of participants somewhat agree that CoreNLP-based pipeline is a sufficient basis. These answers are also reflected in Figure 11 where no significant differences are shown between NER experts and newbies.

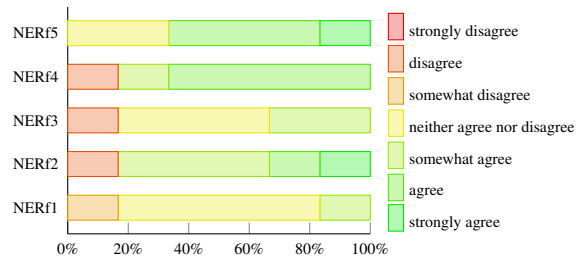


Figure 10: Average Values of the Answers of the NER Features Questionnaire.

Results Related to Improvement: Participants were asked to fill out a questionnaire concerning possible shortcomings (Imp1), ideas for improving SNERC

Table 16: Responses (%) for the NER Features Questionnaire.

Scale	NERf1 (%)	NERf2 (%)	NERf3 (%)	NERf4 (%)	NERf5 (%)
strongly agree	0.00	12.5	0.00	0.00	0.00
agree	0.00	37.5	0.00	37.5	50.0
somewhat agree	12.5	25.0	37.5	50.0	12.5
neither agree nor disagree	50.0	0.00	50.0	0.00	25.0
somewhat disagree	37.5	0.00	0.00	0.00	0.00
disagree	0.00	12.5	12.5	12.5	0.00
strongly disagree	0.00	0.00	0.00	0.00	0.00

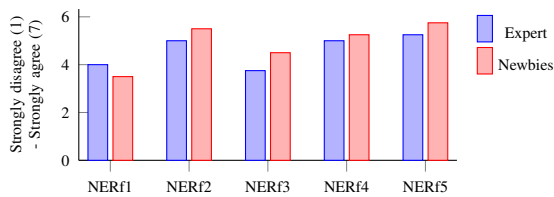


Figure 11: Average Values of the Answers of the NER Features Questionnaire divided by User Groups.

(Imp2), and quality of support materials (including the provided tutorial) (Imp3). For item Imp3 (quality of support materials), all participants stated that the support materials and provided tutorials were satisfactory and sufficient.

5 CONCLUSION

This paper discusses the results of quantitative and qualitative experiments validating our approach for ML-based NER in an innovative Knowledge Management System for Applied Gaming. Two groups of users including NER experts and novice users (newbies) participated in this research project.

Our initial evaluation is based on controlled experiments and aimed at comparing results of the participants performing two fundamental tasks of NER model training, namely “data cleanup” and “data annotation”. The results of this evaluation revealed that both groups performed nearly identical in both tasks while designing and developing a new NER model in their domain. Our second evaluation relied on a walkthrough experiment and used questionnaires to collect answers about the *usability, usefulness, user interface* and *NER features of SNERC*. It was found that SNERC, which is currently based on the CoreNLP framework, is a valid tool for NER. It provides a rich set of features with various GUI components to design, develop, train, and monitor NER models while supporting various preliminary steps of ML-based model training. Our evaluation results

also revealed that supporting other data formats such as JSON, XML or making use of a standard library for NLP related tasks (like spaCy) could be helpful to make our application more flexible. Future work would consider developing a more agnostic approach, to enable using and combining various NLP frameworks easily adapting to new requirements, such as multiple data formats for data annotation and model training tasks.

REFERENCES

- Alavi, M. and Leidner, D. E. (2001). Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS quarterly*, pages 107–136.
- Arellano, A., Zontek-Carney, E., and Austin, M. A. (2015). Frameworks for natural language processing of textual requirements. *International Journal On Advances in Systems and Measurements*, 8:230–240.
- Christian, N. (2021). *Supporting Information Retrieval for Emerging Knowledge and Argumentation*. Phd.
- Croft, W. B., Metzler, D., and Strohman, T. (2010). *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading.
- David R. and Sandra L. (2005). *Serious games: Games that educate, train, and inform*. ACM.
- Dias, M., Boné, J., Ferreira, J. C., Ribeiro, R., and Maia, R. (2020). Named entity recognition for sensitive data discovery in portuguese. *Applied Sciences*, 10(7):2303.
- Finstad, K. (2010). The Usability Metric for User Experience. *Interacting with Computers*, 22(5):323–327.
- Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O’Reilly Media.
- Gronau, N. (2002). The knowledge café—a knowledge management system and its application to hospitality and tourism. *Journal of Quality Assurance in Hospitality & Tourism*, 3(3-4):75–88.
- Helander, M. G. (2014). *Handbook of human-computer interaction*. Elsevier.

- Jiang, J. (2012). Information extraction from text. In *Mining text data*, pages 11–41. Springer.
- Kang, N., van Mulligen, E. M., and Kors, J. A. (2012). Training text chunkers on a silver standard corpus: can silver replace gold? *BMC bioinformatics*, 13(1):1–6.
- Kitchenham, B. A. (1996). Evaluating software engineering methods and tool part 1: The evaluation context and evaluation methods. *ACM SIGSOFT Software Engineering Notes*, 21(1):11–14.
- Konkol, I. (2015). *Named entity recognition*. Phd.
- Liu, M., Peng, X., Jiang, Q., Marcus, A., Yang, J., and Zhao, W., editors (2018). *Searching StackOverflow Questions with Multi-Faceted Categorization*. ACM.
- Lund, A. M. (2001). Measuring usability with the use questionnaire. *Usability interface*, 8(2):3–6.
- Mahalakshmi, G. S. (2015). Content-based information retrieval by named entity recognition and verb semantic role labelling. *Journal of universal computer science*, 21(13):1830.
- Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.
- Mao, X., Xu, W., Dong, Y., He, S., and Wang, H. (2007). Using non-local features to improve named entity recognition recall. In *Proceedings of the Korean Society for Language and Information Conference*, pages 303–310. Korean Society for Language and Information.
- MSV, J. (2020). Why do developers find it hard to learn machine learning?
- Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30:3–26.
- Natek, S., Zwilling, M., et al. (2016). Knowledge management systems support seci model of knowledge-creating process. In *Joint International Conference*, pages 1123–1131.
- Nawroth, C., Engel, F., Eljasik-Swoboda, T., and Hemmje, M. L. (2018). Towards enabling emerging named entity recognition as a clinical information and argumentation support. In *DATA*, pages 47–55.
- Norman, D. A. and Draper, S. W. (1986). *User Centered System Design; New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc.
- Onal, K. D. and Karagoz, P. (2015). Named entity recognition from scratch on social media. In *Proceedings of 6th International Workshop on Mining Ubiquitous and Social Environments (MUSE), co-located with the ECML PKDD*, volume 104.
- Palshikar, G. K. (2013). Techniques for named entity recognition: a survey. In *Bioinformatics: Concepts, Methodologies, Tools, and Applications*, pages 400–426. IGI Global.
- Pinto, A., Gonalo Oliveira, H., and Oliveira Alves, A. (2016). Comparing the performance of different nlp toolkits in formal and social media text. In *5th Symposium on Languages, Applications and Technologies (SLATE'16)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
- Powers, D. M. (2020). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.
- Salman, M., Mertens, J., Vu, B., Fuchs, M., Heutelbeck, D., and Hemmje, M. (2017). Social network-based knowledge, content, and software asset management supporting collaborative and co-creative innovation. In *Collaborative European Research Conference: CERC 2017*, pages 16–25.
- Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.
- Swoboda, Eljasik, T. (2021). *Bootstrapping Explainable Text Categorization in Emergent Knowledge-Domains*. Phd.
- Tamla, P. (2022). *Supporting Access to Textual Resources Using Named Entity Recognition and Document Classification*. PhD thesis.
- Tamla, P., Böhme, T., Hemmje, M., and Fuchs, M., editors (2019). *Named Entity Recognition supporting Serious Games Development in Stack Overflow Social Content*. International Journal of Games Based Learning.
- Thenmalar, S., Balaji, J., and Geetha, T. (2015). Semi-supervised bootstrapping approach for named entity recognition. *arXiv:1511.06833*.
- Thielsch, M. and Stegemöller, I. (2014). Münsteraner fragebogen zur evaluation-zusatzmodul diskussion. In *Zusammenstellung sozialwissenschaftlicher Items und Skalen*.
- Vu, B. (2020). *A Taxonomy Management System Supporting Crowd-based Taxonomy Generation, Evolution, and Management*. PhD thesis, University of Hagen, Germany.
- Zhang, D. and Tsai, J. J. (2003). Machine learning and software engineering. *Software Quality Journal*, 11(2):87–119.
- Zhang, M., Geng, G., and Chen, J. (2020). Semi-supervised bidirectional long short-term memory and conditional random fields model for named-entity recognition using embeddings from language models representations. *Entropy*, 22(2):252.