Information and Software Technology 123 (2020) 106310

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Deep learning model for end-to-end approximation of COSMIC functional size based on use-case names

Mirosław Ochodek [a,*], Sylwia Kopczyńska [a], Miroslaw Staron [b]

[a] Poznan University of Technology, Faculty of Computing and Telecommunications, ul. Piotrowo 2, 60-965 Poznań, Poland
[b] Chalmers | University of Gothenburg Sweden

ARTICLE INFO

Keywords:
Functional size approximation
Approximate software sizing methods
COSMIC
Deep learning
Word embeddings
Use cases

ABSTRACT

Context: COSMIC is a widely used functional size measurement (FSM) method that supports software development effort estimation. The FSM methods measure functional product size based on functional requirements. Unfortunately, when the description of the product's functionality is often abstract or incomplete, the size of the product can only be approximated since the object to be measured is not yet fully described. Also, the measurement performed by human-experts can be time-consuming, therefore, it is worth considering automating it.

Objective: Our objective is to design a new prediction model capable of approximating COSMIC-size of use cases based only on their names that is easier to train and more accurate than existing techniques.

Method: Several neural-network architectures are investigated to build a COSMIC size approximation model. The accuracy of models is evaluated in a simulation study on the dataset of 437 use cases from 27 software development projects in the Management Information Systems (MIS) domain. The accuracy of the models is compared with the Average Use-Case approximation (AUC), and two recently proposed two-step models—Average Use-Case Goal-aware Approximation (AUCG) and Bayesian Network Use-Case Goal AproxImatioN (BN-UCGAIN).

Results: The best prediction accuracy was obtained for a convolutional neural network using a word-embedding model trained on Wikipedia+Gigaworld. The accuracy of the model outperformed the baseline AUC model by ca. 20%, and the two-step models by ca. 5–7%. In the worst case, the improvement in the prediction accuracy is visible after estimating 10 use cases.

Conclusions: The proposed deep learning model can be used to automatically approximate COSMIC size of software applications for which the requirements are documented in the form of use cases (or at least in the form of use-case names). The advantage of the model is that it does not require collecting historical data other than COSMIC size and names of use cases.

1. Introduction

Functional size measurement (FSM) methods [1] have been successfully used as a proxy of development effort for over four decades since the introduction of Function Point Analysis (FPA) in 1979 [2]. The advantage of using the FSM methods for size measurement is that they are independent of the technologies used to develop the software application and base the measurement only on functional requirements. Over the years, multiple derivatives of FPA have been proposed (e.g., NESMA FPA [3], FISMA FPA [4], Mark II FP [5]). In the late 1990s, the COSMIC method (ISO/IEC 19761:2011) [6] was proposed. It is now considered to be the first second-generation FSM method. The method proposed visible improvements in the measurement process over FPA and is compliant with the fundamental measurement principles [7].

Unfortunately, in some cases, applying a standard COSMIC measurement process (or any other FSM method) could be too expensive because of incomplete/imprecise requirements or limited resources available in a project to perform upfront requirements elicitation. It is a common problem observed at early stages of software development when effort estimates have to be provided based on some initial sketches of requirements. Similarly, the FSM methods might be difficult to apply in agile projects. Such projects base their development on emergent requirements specifications [8–10], usually called product backlogs. Many authors recommend that a good product backlog should be kept DEEP (Detailed Appropriately—Emergent—Estimated—Prioritized) [11]. Essentially, it means that one should focus on specifying the most important requirements while keeping the remaining ones abstract until their time for the implementation come. However, even in agile product de-

* Corresponding author.
E-mail addresses: mochodek@cs.put.poznan.pl (M. Ochodek), sylwia.kopczynska@cs.put.poznan.pl (S. Kopczyńska), miroslaw.staron@cse.gu.se (M. Staron).

https://doi.org/10.1016/j.infsof.2020.106310
Received 19 September 2019; Received in revised form 3 March 2020; Accepted 15 March 2020
Available online 18 March 2020

velopment, it is important to make both short- and long-horizon plans [12]. Unfortunately, applying COSMIC to support the latter is difficult since it would require additional effort to elicit requirements that are too abstract for the COSMIC method. Another option is to approximate the functional product size[1] of the abstract product-backlog items instead of measuring them. Since product backlogs are frequently refined [12], it is worth considering fully automatic techniques for functional size approximation. Finally, it is worth mentioning that product-backlog items are not limited to functional requirements but they could also be non-functional requirements or other issues that are important from the perspective of product development. In this paper, we narrow the scope to functional requirements, however, it is worth mentioning here that there are studies that focus on quantifying non-functional requirements as well (e.g., [14,15]).

Use cases [16] are a widely used method for documenting functional requirements in traditional and agile projects [9,17,18]. They are a scenario-based technique of describing the interaction between end-user (actors) and the system, which leads to obtaining an important *goal* from the user's perspective. They are a flexible tool supporting iterative requirements refinement. Usually, one starts from identifying user goals and specifying use-case names (e.g., "assign a task to an employee"), and later, augment them with scenarios showing how the users obtain these goals [19]. Consequently, when use cases are used to define the scope of the project at early stages of software development most of them will have the form of use-case names (often visualized using UML use-case diagrams). Similarly, in agile projects employing use cases, many product-backlog items will be initially specified as use-case names, which at this stage are similar to high-level user stories [20].

Therefore, it is worth considering approximating COSMIC size based on use-case names to complement the measurement of well-defined requirements. There are numerous COSMIC size approximation techniques [13], however, only a few of them can be used to automatically approximate size only based on use-case names. The simplest technique is the average use-case approximation (AUC) [21,22], which ignores the names of use cases and takes into account only their size. Another technique was proposed by Hussain et al. [23] that approximate COSMIC size based on the frequency of linguistic features. However, it has been shown that this technique does not perform well when requirements are expressed in the form of use-case names or use-case scenarios [24,25]. In our previous study, we proposed a two-step technique to automatic size approximation based on use-case names [24]. In the first step, each use case is classified into one of thirteen semantic types based on its name, and in the second step, this information is used to approximate its functional size. We proposed two prediction models implementing this idea—Average Use-CaseGoal-aware Approximation (AUCG) and Bayesian Network Use-Case Goal AproxImatioN (BN-UCGAIN). Although the models proved to be superior to AUC, they have a drawback which is the necessity of collecting additional information about the types of use-case goals, the structure of their scenarios, and vocabulary to create a historical dataset to train the models. Therefore, using this approximation technique would require extra effort from an organization.

As have already said that using user-case names as the basis for applying COSMIC is difficult, we still want to consider the possibility of constructing a simpler, end-to-end prediction model that would require only the input in the form of use-case names to approximate the COSMIC size of use cases. Since the information about the names of use cases and their size is usually stored in issue tracking systems (e.g., Jira, Redmine), such model could be trained and re-trained seamlessly, without the need for human intervention.

[1] Since size measurement is often used to support *effort estimation*, it is a common practice to use the term *size approximation* instead of *size estimation* to make a clear distinction between these two terms and avoid confusing the reader [13].

The recent advancements in machine learning, and especially in the area of deep learning, revolutionized the domain of natural language processing and text analysis. Deep artificial neural networks are capable of learning meaningful representations of text data without the need for human intervention. For instance, the word-embedding models are able to capture meaningful relationships between words directly from the text. We investigate the possibility of using the power of deep neural networks to design an end-to-end prediction model capable of approximating COSMIC size based on use-case names specified for software applications in the Management Information Systems (MIS) domain. The most important contributions of this study are as follows:

- we show that it is possible to design a fully automatic, end-to-end model using neural networks to approximate COSMIC size of use cases *only* based on their names that visibly outperforms the AUC technique (the main counterpart) and is slightly more accurate than the two-step models AUCG and BN-UCGAIN,
- we analyze different neural networks architectures to learn that the best model accuracy is obtained for convolutional neural network,
- we investigate different pre-trained word embeddings to learn that using the embeddings trained on Wikipedia+Gigaworld (300d), Common Crawl 840B/42B (300d), and Stack Overflow (200d) give the best prediction accuracy.

This paper is structured as follows. In Section 2, we provide the necessary background information regarding use cases, COSMIC, and deep learning. We also discuss the related work in the area of COSMIC size approximation. Section 3 presents the research method used in this study. The results are presented in the following Sections 4 and 5. Section 4 presents the proposed prediction model while Section 5 shows the results of validating the proposed model. The implications of the study and threats to its validity are discussed in Section 6. Section 7 summarizes the findings.

2. Background and related work

In this section, we provide the most important information about use cases, the COSMIC method, and deep learning that we refer to in this paper and we also discuss the related work.

2.1. Use cases

Use cases are a scenario-based technique for describing how actors (people, devices, other systems) can obtain their goals by interacting with the system under specification. They were proposed by Ivar Jacobson in early 1990's [16], and since then, they have been widely used to document functional requirements.

Use cases can be used to document requirements at different levels of abstraction [26], and even to describe business processes [27]. In this study, we focus on user-level use cases that describe the goals that actors can achieve by interacting with the system. Use cases are also flexible when it comes to the level of details they present. For instance, in the beginning, one can start by specifying use-case names that describe the goals of actors. A set of use cases can be summarized using UML use-diagrams, as presented in Fig. 1. In this form, use cases are very similar to user stories [19,28] commonly used in agile projects. A use-case name should be formulated using a simple [verb + object] clause with an implied subject being the actor of the use case (e.g., "buy a product"). Later, use-case names are expanded to scenarios showing how the goals can be obtained by the actors and to alternative scenarios showing how to handle exceptions [29,30].

The flexibility in using use cases makes them well-suited for agile projects. Emerging requirements can be initially specified in the form of use-case names, and later, documented with scenarios. Fully documented use cases can be decomposed by slicing their scenarios and combined with user stories [31].

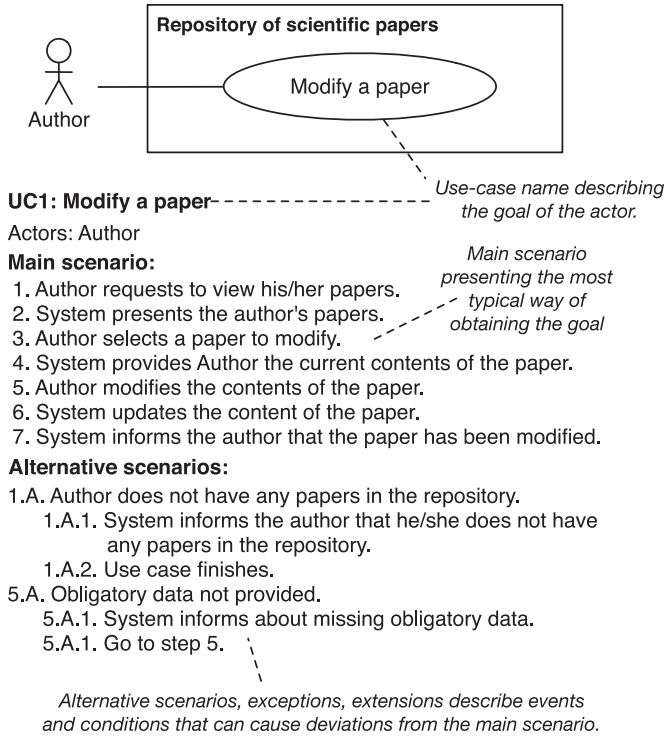

Fig. 1. An example of a use case and use-case diagram.

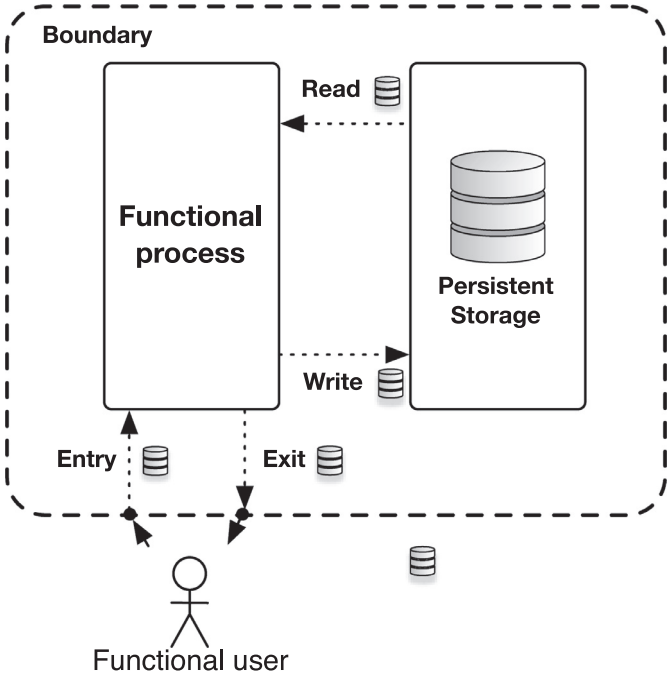

Fig. 2. Data movements in COSMIC.

2.2. The COSMIC method

COSMIC is a widely used second-generation FSM method registered as the international standard ISO/IEC 19761:2003 [6] and compliant with ISO/IEC 14143-1:2007 (R2019) [1]. The method has been developed by the Common Software Measurement International Consortium (COSMIC).

The Basic Functional Components (BFCs) considered in the COSMIC method are data movements (Entry, Exit, Read, and Write). The generic flow of data groups in the COSMIC method is presented in Fig. 2. Entries and Exits move data groups through boundaries which are "conceptual interface between the software being measured and its functional users" [32]. Data movements are identified and counted within the scope of so-called *functional processes*. A functional process is a set of data movements, representing an elementary part of the Functional User Requirements (FUR). Each functional process is unique within the FUR and can be defined independently of any other functional process. Every functional process has a triggering Entry and a set of data movements that are needed to meet its FUR for all the possible responses to its triggering Entry [32].

The unit of measure is called COSMIC Function Point (CFP). A single CFP corresponds to a movement of data attributes belonging to a single data group. The functional size is calculated as the total number of data movements within all the functional processes in the scope of measurement.

2.3. COSMIC and use cases

The COSMIC method analyzes functional requirements at the level of functional processes and does not require any particular form of document requirements, as long as they could be mapped to the constructs used by the method. The mapping between COSMIC and use-case models has been discussed by Marín et al. [33]. The authors seem unanimous in their opinions that the availability of detailed use-case scenarios is necessary to apply the COSMIC method based on use cases [34–37]. It is required that use-case scenarios provide detailed information about all the data movements, including local reads and writes.

2.4. COSMIC in agile projects

Recently published case studies on using COSMIC in agile projects have shown promising results, often reporting superiority of the COSMIC method over the human-judgment-based methods frequently used in agile projects.

The three case studies conducted by Salmanoglu et al. [38] showed that COSMIC size can be effectively used to estimate effort in agile projects. They reported that COSMIC-based estimates were more precise than those provided using Story Points [39].

Commeyne et al. [40] compared the performance of estimation models built using Story Points and COSMIC. They reported that the use of COSMIC leads to effort estimation models with much smaller variances. Another reported benefit of using COSMIC was that it allowed objective comparison of productivity across tasks within a Scrum environment.

Both studies observed that it is possible to apply COSMIC in agile projects as long as the requirements include enough details. Desharnais et al. [41] proposed a systematic approach to measuring COSMIC size based on user stories. One of the steps is the identification of functional processes based on the information provided in a user story. One may need to collect additional information about the requirements to perform the measurement. However, as shown by another study by Desharnais et al. [42], the analysis of user stories for the purpose of applying COSMIC might help to improve the quality of requirements. Recently, Ecar et al. [43] proposed a new template for writing user stories that is supposed to be more expressive in terms of COSMIC size estimation.

2.5. The COSMIC size approximation techniques

Functional size approximation techniques can be classified as *early* or *rapid* [44]. A technique is considered early if it allows for approximating functional size before the Functional User Requirements are specified at the level of granularity accepted by the FSM method. Rapid techniques allow approximating the size more efficiently, i.e., by reducing the time or cost required to perform a standard measurement (usually, at the cost of compromising the accuracy of measurement).

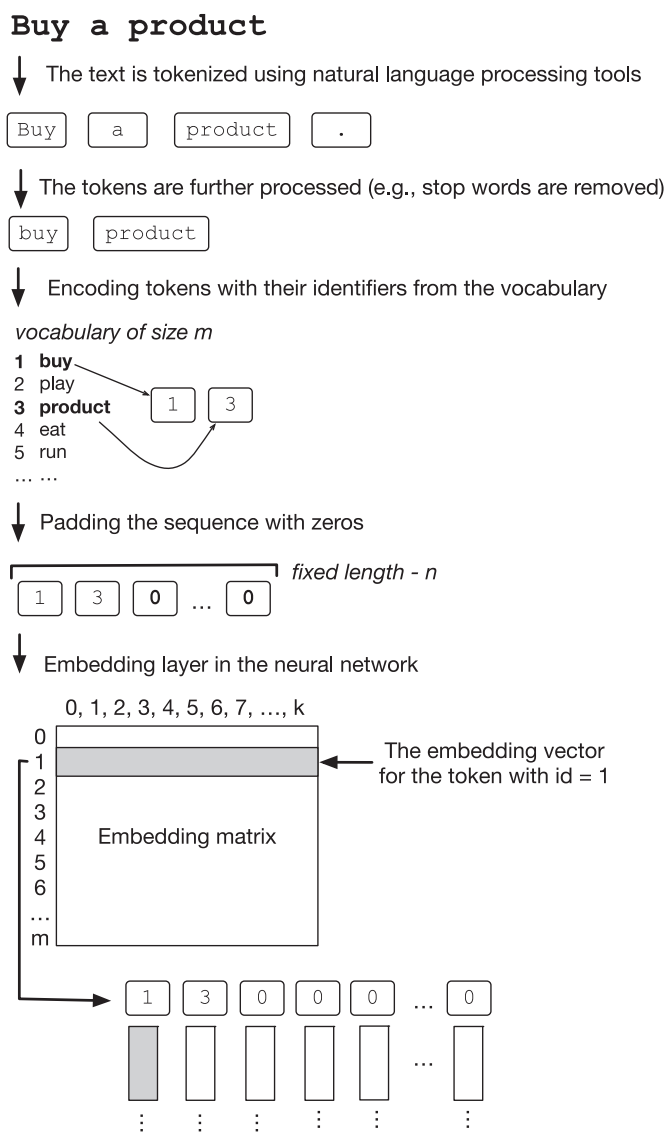

Fig. 3. Transforming text to vectors using word embeddings and embedding matrix.

A widely used technique for functional size approximation that is both early and rapid is to derive a scaling factor as an average size of requirements. From the perspective of this study, the most interesting example of this technique is the average use-case approximation (AUC) technique [21] because it can be easily applied to approximate functional size based on use-case diagrams [22]. AUC can be easily used in agile projects using emergent requirements specification as it only requires information about the number of use cases to be included in an iteration and historical data about the COSMIC size of use cases. Therefore, we treat AUC as the main counterpart for the approximation technique proposed in this study.

In our previous study [24], we investigated the relationship between the types of use-case goals and functional size of use cases. We proposed a taxonomy of thirteen use-case goals (e.g., Create, Retrieve, Update, Delete, Dynamic Retrieve, Complex Internal Activity). We proposed two size approximation models called Average Use-Case Goal-aware Approximation (AUCG) and Bayesian Network Use-Case Goal AproxImatioN (BN-UCGAIN) that approximate size in two steps. Firstly, the type of use case is determined based on its name. A hybrid classifier that combines manually-designed rules and decision tree is used to classify use cases. Use-case names are processed using natural language processing (NLP) tools, including manually composed sets of synonyms (synset) based on the WordNet lexical database [45]. In the second step, the use-case size is approximated based on the type of its goal. Therefore, in order to use these prediction models, a company needs to collect information about the vocabulary used in use cases (and compose/update the synsets), types of use-case goals, and in the case of BN-UCGAIN, the information about the types of transaction in use-case scenarios [46,47].

Another early and rapid technique that could be applied to approximate COSMIC size of textual requirements (e.g., use cases) was proposed by Hussain et al. [23]. They used natural language processing (NLP) tools to analyze the frequency of syntactic linguistic features in functional requirements (the number of words, frequency of nouns phrases, the number of keywords, etc.). They used this information to train a tree-based classifier to categorize requirements into complexity classes established based on historical data. Although promising results were obtained when the technique was used to approximate size of unstructured requirements, it seems to perform visibly worse when applied to predict the size of use cases based on their names (it was outperformed by AUC) [24] or scenarios [25]. Consequently, we did not consider this method as a baseline for comparison in this study.

The above mentioned COSMIC size approximation techniques seem to be the most similar to the technique proposed in this study. However, there are numerous studies on COSMIC rapid and early approximation techniques that are worth mentioning here. For instance, Bagriyanik and Karahoca [48] proposed a rapid approximation technique that is capable of automatically approximate functional size based on domain-specific requirements ontology. The method seems oriented towards later phases of software development and maintenance because it requires detailed information about requirements, services, and conceptual data models. Valdés et al. [49,50] proposed the Estimation of Projects in a Context of Uncertainty model (EPCU), which is a fuzzy logic-based model that enables COSMIC size approximation. De Vito and Ferrucci [51] proposed a quick and early technique for approximating COSMIC functional size based on use-case models. They proposed to analyze the flow of data groups in use-case preconditions and scenarios. Mushtaq and Wahid [52] designed a technique for approximating COSMIC size of mobile applications based on UML models. Recently, Lavazza and Morasca [53] evaluated the Average Functional Process (AFP), Equal Size Bands (ESB), and two new approaches for defining bands in the Fixed Size Classification technique to observe that the methods using bands can provide accurate estimates. They also observed that AFP generally provides accurate estimates, but in some cases, its prediction errors are too large to be acceptable.

2.6. Deep learning for text analysis

Deep learning is a specific subfield of machine learning that emphasizes learning successive *layers of increasingly meaningful representations* of the input data [54]. These layered representations are nearly always learned using artificial neural networks (ANN) models, structured in layers stacked on top of each other. The data transformation implemented by a layer is parameterized by its weights (also called the parameters of a layer). While training a model, we want to find a set of values for the weights of all layers in a network, such that the model will correctly map the inputs to their associated ground truth. The model and its layer can have parameters whose values have to be explicitly defined before the training. Such parameters are called *hyperparameters* (e.g., the number of weights in a layer). There are numerous types of layers available. In the following paragraphs, we very briefly describe a few of them that are used in this study.

Text can be treated as a sequence of tokens. In order to be processed by an ANN model, it has to be vectorized. A commonly used approach to vectorize texts is to use word embeddings. In word embeddings, each word is represented as a dense, low-dimensional floating-point vector.

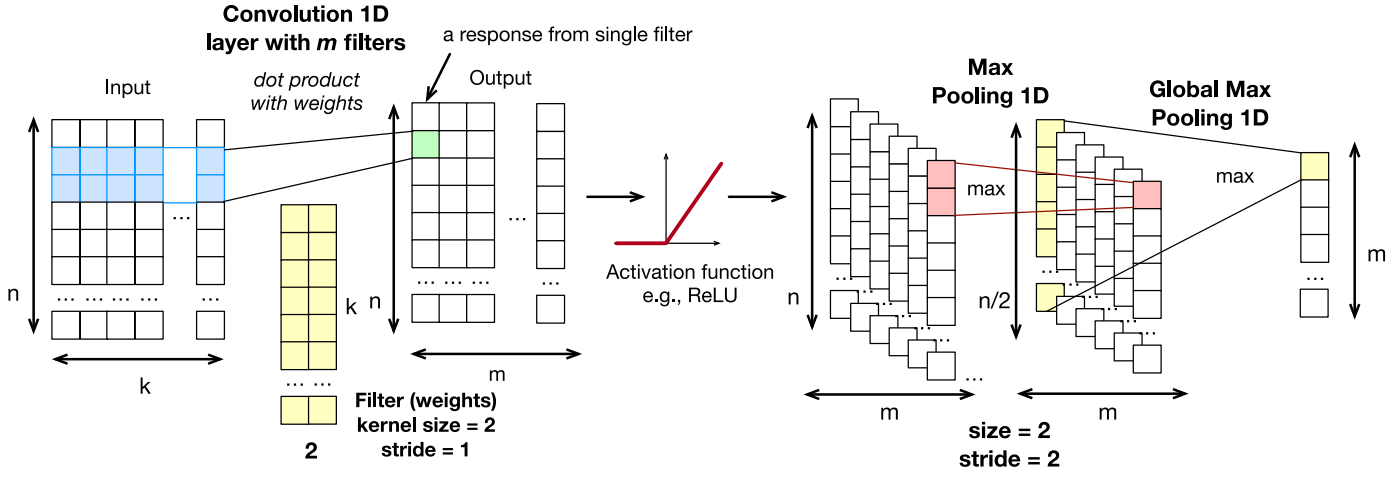

Fig. 4. An example of combining convolutional and different types of pooling layers.

Such vectors are learned from data. They can be learned independently of the task being solved or as a part of it. An important benefit of using word embeddings is that the geometric relationships between word vectors should reflect the semantic relationships between these words. For instance, there is a well-known example of the relationships between the words 'king', 'man', 'woman', and 'queen'. When we perform the following sequence of operations on the vectors representing these words: 'king' − 'man' + 'woman', we obtain a vector that is nearly identical to the one representing the word 'queen' [55]. In this study we used word-embedding models trained with word2vec [56] and GloVe [57].

The process of transforming a use-case name to vectors is presented in Fig. 3. The input text is tokenized, transformed into a sequence of indices of tokens in the vocabulary, and finally padded with zeros, so all sequences have the same, fixed size (n). The sequence is further processed by the embedding layer in a neural-network model. In this layer, token identifiers are exchanged with the corresponding vectors of size (k). Therefore, the input for a single use-case name would have the form of a 2D tensor ($n \times k$).

In convolutional neural networks (CNNs), a vectorized input is processed by a stack of 1D convolutional layers (Conv1D), usually, combined with pooling layers. Fig. 4 shows how a single convolutional layer processes its input. Each layer has a number of filters that are convolved over the patches of the input. The responses from the filters are transformed using a non-linear activation function, e.g., Rectified Linear Unit (ReLU). Pooling layers are used to reduce the number of parameters and computation in the network and help controlling overfitting by progressively reducing the spatial size of the network. The pooling layer applies a given function on a patch of the input (e.g., mean or maximum) and replaces the patch with the calculated value. As presented in Fig. 4, we can have over-time pooling (Pooling1D) and global pooling (GlobalPooling1D) layers depending on the axes they operate on. Another regularization layer used to control overfitting is called Dropout. The idea is to randomly exclude some of the weights from being modified in a given iteration of the training procedure.

Recurrent neural networks (RNNs) are another family of models that are often used to process text. A characteristic feature of RNNs is that they augment the input in time t with some information derived from the previous inputs. Therefore, they have a kind of built-in memory. Probably, the two most frequently used types of RNNs are Gated Recurrent Units (GRUs) [58] and Long Short-Term Memory (LSTM) [59]. Both have similar capabilities, however, in the context of natural language processing, the GRU-based models seem to be more accurate than the LSTM-based models [60].

We can combine CNN and RNN layers to create a CNN-RNN network. Usually, CNN layers are applied first and then their output is processed by recurrent layers.

3. Research method

We decided to follow the Design Science Research (DSR) methodology [61,62], which is a problem-solving paradigm that focuses on creating and evaluating artifacts and solutions for practical purposes. In particular, we focus on two of the DSR-engineering-cycle steps—designing and validating the treatment.

The replication package for this study is publicly available on GitHub.[2]

3.1. Research questions

We defined the following research questions that need to be answered to design and validate the new size-approximation model:

- RQ1: Which artificial-neural-network architecture provides the best accuracy while approximating the COSMIC size of use cases based only on their names?
- RQ2: Which of the available pre-trained word embeddings allow to obtain the best accuracy while approximating the COSMIC size of use cases based on their names?
- RQ3: Is the proposed prediction model more accurate than AUC?
- RQ4: Is the proposed prediction model more accurate than two-step models (AUCG and BN-UCGAIN)?

The first two questions (RQ1, RQ2) are design-related. The answers will allow us making decisions about the design of the prediction model. We need to investigate different ANN architectures to find the most promising one. Also, since there are many pre-trained word-embedding models that capture semantic dependencies between the meaning of words, we need to find those that provide the best results for the considered problem.

The remaining research questions (RQ3, RQ4) regard the validation of the proposed treatment. To be useful, the proposed prediction model need to be more accurate than AUC and at least as accurate as two-step prediction models (AUCG and BN-UCGAIN) since its advantage is that it should require less effort needed to collect the training data than the two-step models (there is no need to collect data about use-case types, vocabulary, or structure of their scenarios) and to maintain the end-to-end model (i.e., re-training the model when new historical data are available).

[2] Replication package — https://github.com/mochodek/deep-cosmic-use-cases.

Table 1
Application domain and basic description of the projects under study. *Type: N—new development; C—customization of an existing product; E—enhancement project; Origin: I—project developed by a software development company; U—project developed at the university by staff or students for internal use or external customer; UC: no. of user-level use cases; CFP: the product functional size delivered by the project; Effort: recorded effort.*

ID	Developer	Type	Origin	UC	CFP	Effort [h]	Product description
P01	D1.1	N	U	42	693	3593	Java, Oracle DBMS, Hibernate, GWT-based custom framework, JSON, OSGi, Backend application for the university admission system (Rich Internet Application).
P02	D1.1	E	U	17	211	1681	Java, Oracle DBMS, Apache Struts 1.2, Java Swing, Web-based frontend for the university admission system. Some parts were re-used from the previous prototype version.
P03	D1.2	N	U	36	342	606	PHP, PHPLiteMVC, PostgreSQL DBMS, C#, Web-application and daemon for managing life-cycle of smart cards with an exemplary client application.
P04	D1.2	E	U	23	159	417	PHP, PHPLiteMVC, PostgreSQL DBMS, C#, Two client applications for the system P16 (web-based application for managing life-cycle of smart cards).
P05	D1.3	E	U	19	151	1623	Java, GWT, PostgreSQL DBMS, Hibernate, Web-application for managing e-protocols for students grades
P06	D1.3	N	U	26	161	1344	C#, ASP.Net, MS SQL DBMS, Web-based Customer Relationship Management (CRM) system.
P07	D1.3	N	U	13	113	1260	Java, SmartGWT, PostgreSQL DBMS, Hibernate, Web-application for monitoring assignments of organizational duties.
P08	D1.3	E	U	8	78	1230	PHP, Moodle, PostgreSQL DBMS, Web-application, a module to Moodle LMS enabling surveying students and alumni, which integrates with the University e-services.
P09	D1.3	E	U	13	138	1220	Java, Oracle DBMS, Hibernate, GWT-based custom framework, JSON, OSGi, University admission system for foreign students.
P10	D1.3	N	U	9	70	1216	Java, Ninja Framework, PostgreSQL DBMS, Hibernate, Web-application for collecting and raporting bibliometric information.
P11	D1.3	N	U	10	81	1030	Java, PostgreSQL DBMS, Hibernate, Web-application for planning educational duties assignments.
P12	D1.3	N	U	11	45	992	Java, Java Servlets, Spring, Android, SQLite, MySQL, Mobile application for federation of libraries.
P13	D1.3	N	U	12	91	775	Java, Spring, PostgreSQL DBMS, Hibernate, Web-application for collecting information about a faculty.
P14	D1.3	N	U	13	85	755	Java, Spring, PostgreSQL DBMS, Hibernate, Web-application for storing and evaluating studies programmes.
P15	D1.3	N	U	14	86	720	Java, Java Servlets, JSP, Java Swing, SOAP, MySQL DBMS, Web and standalone application for managing members of the organization.
P16	D1.3	N	U	7	49	695	Ruby, PostgreSQL DBMS, Web-application that collects information about alumni from web pages.
P17	D1.3	N	U	6	54	514	Java, Hibernate, Axis, PHP, C+, Bibliometric Information System. A system for collecting information regarding publications and citations. (Limited GUI)
P18	D1.3	N	U	18	116	397	Java, Apache Struts 2, Hibernate, Web-based system supporting the assignment of B.Sc and M.Sc. theses projects.
P19	D2	N	I	31	399	3037	C#, ASP.Net, MS SQL DBMS, custom web-framework and Web-based e-commerce solution.
P20	D3	N	I	11	125	1173	Python, Django, PostgreSQL, Web application for collecting and tracking projects metrics.
P21	D3	N	I	19	154	742	Python, Plone, Zope, Web application developed based on existing CMS solution.
P22	D4	N	I	25	142	512	PHP, PostreSQL DBMS, Yii web-framework, ExGWT (administration panel), An e-learning web platform.
P23	D4	N	I	9	59	64	PHP, PostreSQL DBMS, Yii web-framework, Web application for handling customers' orders.
P24	D5	N	I	10	45	277	Java, PHP, MySQL DBMS, Eclipse Rich Client Platform (RCP), Web-based repository of invoices with additional standalone client application.
P25	D6	C	I	10	85	492	Python, Plone, Zope, Content Management System (CMS).
P26	D7	N	I	15	136	614	Delphi, Firebird DBMS, Bank system integrating payments into virtual accounts in one real account.
P27	D7	N	I	11	117	1917	Delphi, Firebird DBMS, Integration of two sub-systems within the ERP scale system.

3.2. Dataset

The study is based on the analysis of 437 use cases[3] from 27 software development projects. It is the combination of the datasets used in the previous studies [24,25,46]. Brief characteristics of the projects and descriptions of the developed products are presented in Table 1.

The projects were developed by six software development companies (D2-D7) and Poznan University of Technology—PUT (D1). The projects owned by PUT were developed by a team established to develop a system for handling student admission process at the University (D1.1); Department of Software Development—a unit at PUT, hiring professional software developers to deliver software for internal purposes of the University (D.1.2); Software Development Studio (SDS) [63] (D1.3); or as a B.Sc. project for the internal use at the University (D1.4).

The measurement was performed based on use-case models and supplementary material available for the projects, e.g., data models, user interface (UI) designs, application screens, and working application [24]. Fig. 5 presents the distributions of COSMIC size of use cases in the dataset. The average size of use case in the dataset is ca. 9 CFP (median = 7, SD = 6), however, there are five use cases with size larger than 30 CFP containing scenarios for multiple related goals (e.g., CRUD-like use cases—create, retrieve, update, and delete). The average size of the use case does not differ visibly between the projects (SD = 2.7).

We used the spaCy library[4] to process the use-case names. We removed stop words, whitespace characters, and some symbols (e.g., 'and', 'or', '.', '?'). The resulting dataset is available in the replication package of our study.

3.3. Designing the prediction model: Finding the architecture of the artificial-neural-network

The space of possible ANN architectures and their hyperparameters is too large to be exhaustively searched in a reasonable time. Therefore, we decided to use a funnel-like search procedure to systematically

[3] The dataset consisted of 438 use cases, however, we rejected one of the use cases from the project P26 because its scenarios did not contain any user-performed actions.

[4] spaCy — https://spacy.io.

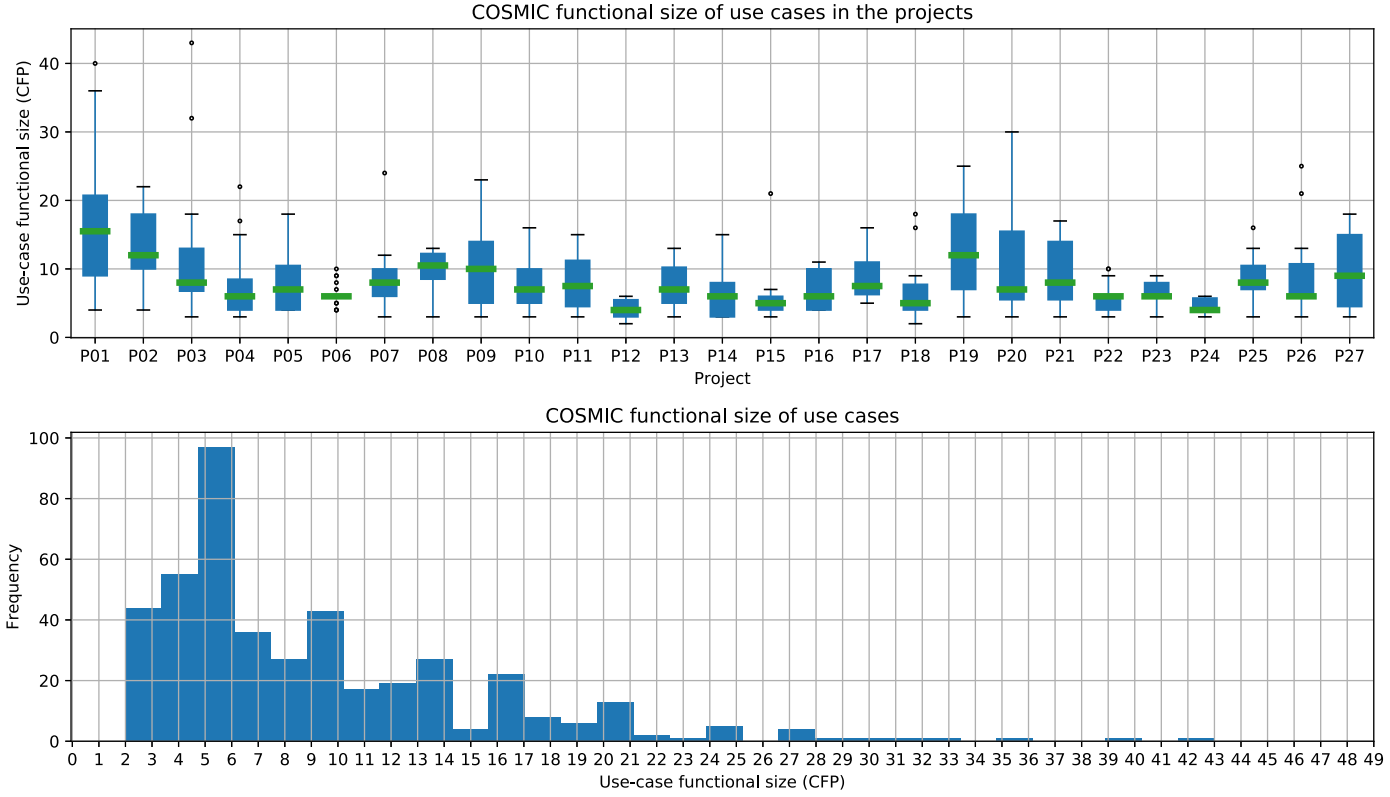

Fig. 5. COSMIC functional size of use cases per project (top) and in the dataset (bottom).

search through its sub-space and make the key architectural decisions regarding the model. We used Keras [64] to implement the ANN-based prediction models and Talos[5] to tune the hyperparameters of the models.

We based all of the tests on random train-test splits (33% observations left for testing) and used mean absolute error (MAE) as the prediction quality criterion. Eq. (1) presents the formula for calculating absolute error (AE) or absolute residual (AR); y_j is the actual size of use case j, while $\hat{y_j}$ is its predicted size.

$$AE_j = |y_j - \hat{y_j}| \tag{1}$$

We used train-test split for model selection and cross-validation to validate its accuracy to reduce the risk of biasing the results by tuning and validating the model using the same settings [65].

In the first step of the analysis, we wanted to decide on the type of layers used in our prediction model, i.e., to choose between convolutional and recurrent layers (or use both of them). Taking into account the size of the available dataset (the necessity of keeping the number of parameters small to prevent the models from overfitting to the data) we decided to start by comparing five ANN architectures (k equals to 1 and 2):

- Convolutional Neural Network (CNN) — Embedding → (Conv1D → AveragePooling1D) × k → Dropout → Dense
- Recurrent Neural Network (RNN) — Embedding → GRU × k → Dropout → Dense
- Convolutional + Recurrent Neural Network (CRNN) — Embedding → Conv1D → AveragePooling1D → GRU → Dropout → Dense

We decided to use the average-over-time pooling (size and stride = 2) instead of max-over-time pooling at this stage since it provides smoother filtering of the data, more reasonable for the regression problem. We set the numbers of filters and recurrent units to 4, 8, 16, and 32; batch sizes to 64, 128, and 256; the number of epochs to 100, 300, and 500; and dropout levels to 0.0, 0.2, 0.5, and 0.8. All of the considered configurations used the ReLU activation function in the hidden layers and linear function in the last, dense layer. We used the kernels of size 3 in the convolutional layers. We employed the Adam optimizer [66] with the learning rate set to 0.001. Finally, we used the word embeddings dedicated to the Software Engineering domain that was trained on Stack Overflow using word2vec [67]. More information about the hyperparameters can be found in the study replication package.

In the following steps, we performed an iterative tuning of the models, each time observing the effect of changing some of the models hyperparameters on the accuracy of the models.

Finally, we tried different pre-trained word embeddings models: word2vec models trained on Stack Overflow (200d) [67] and Google News (300d) [56]; and GloVe models trained on Wikipedia+Gigaword (300d, 200d, 100d, and 50d), Twitter (200d, 100d, 50d, and 25d), and Common Crawl (300d) [57].

3.4. Validating the proposed model

We based the framework for validating the accuracy of prediction models on the guidelines by Shepperd and MacDonell [68].

In order to assess the accuracy of prediction model P_i based on a set of n predictions, we calculated mean absolute error (MAE), median absolute error (MdAE), and standardized accuracy measure (SA). SA_{P_i} is calculated according to Eq. (2). It shows how much more (or less) accurate the model P_i is compared to random guessing.

$$SA_{P_i} = \left(1 - \frac{MAE_{P_i}}{\overline{MAE}_{P_0}}\right) \times 100\% \tag{2}$$

[5] Talos — https://github.com/autonomio/talos.

$\overline{MAE}_{P_0}$ is the MAE of random guessing. We determined it using the exact algorithm proposed by Langdon et al. [69], which allows random guesses to alight on the correct answer.

We used the above criteria and 10 runs of 10-fold cross-validation to evaluate each of the proposed size-approximation models P_i.

We used the following set of criteria comparing the accuracy of the proposed prediction models and the baseline models (AUC, AUCG, and BN-UCGAIN):

- the difference between SA calculated for the models (ΔSA);
- the results of one-tailed Wilcoxon signed-rank test for Absolute Errors (AE)—we decided to use a non-parametric test after performing a series of Shapiro-Wilk tests that show strong evidence against the normality assumption;
- Cliff's δ [70] — a non-parametric effect size measure that estimates the probability that a value selected from one of the groups is greater than a value selected from the other group, minus the reverse probability. The measure ranges between +1 and −1. The extreme values indicate the absence of overlap between the two samples whereas zero suggests equivalence of samples distributions [71]. We used the thresholds by Kitchenham et al. [72] to interpret the values of Cliff's δ (small: $|\delta|$=0.112, medium: $|\delta|$=0.276, and large: $|\delta|$=0.428).
- Number Needed to Treat (NNT) — a measure calculated as δ^{-1} [73]. In the context of this study, NNT can be interpreted as the number of use cases which size would have to be approximated using a given prediction model to expect that the size of one more of them was approximated with a higher (or lower) accuracy than if the same use case had been estimated with the use of a baseline approximation method.

ΔSA and NNT complement each other. The former metric shows the "on-average" improvement in the prediction error one can expect, while the latter tells us how consistent the improvement is in time.

4. The COSMIC size approximation model

The information about each of the steps of the hyperparameters optimization procedure (see Section 3.3) are presented in Fig. 6. It shows the goal of each step of the process, the combinations of hyperparameters that were used to train the family of models, and the lowest MAE obtained during the iteration.

In the first iteration, we compared the accuracy of five ANN architectures (one- and two-layer CNN, one- and two-layer RNN, and CNN-RNN). After generating and studying 2016 prediction models, we observed that the most accurate were the 2-layer CNN models. The lowest MAE was equal to 3.68. The RNN models performed visibly worse (the lowest MAEs were equal to 4.02 and 3.94). Also, the CNN-RNN models made predictions with higher error rates than the CNN models (the lowest MAE was equal to 3.93). We believe that the worse performance of the recurrent networks could be caused by the fact that use-case names are formulated as short sentences (with a fixed structure), thus, even networks consisting of a single convolutional layer is sufficient to cover the whole context and require fewer model parameters to be trained than recurrent layers. However, we cannot say that the observed superiority of CNNs is not caused by the size of the dataset or some specific properties of the use cases it contains. Also, we did not study all the possible combinations of hyperparameters. It is also worth emphasizing that MAE was quite similar for all the compared models and the process of hyperparameters tuning did not allow us to find a model which accuracy would stand out from the other ones by a large margin (the difference in MAE between the worst from the best prediction models found in the first iteration of the tuning procedure and the best one was smaller than 0.5 CFP).

Taking the above into account, we formulate the following answer to RQ1:

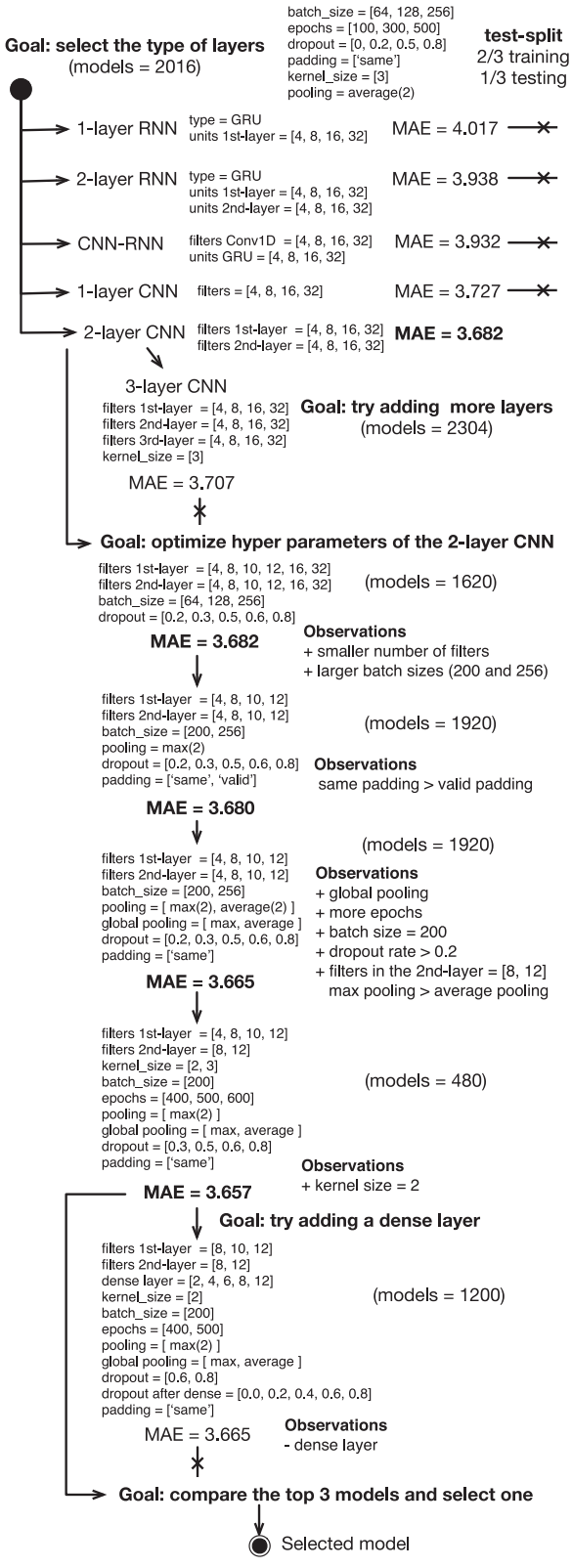

Fig. 6. The steps of the model-selection procedure.

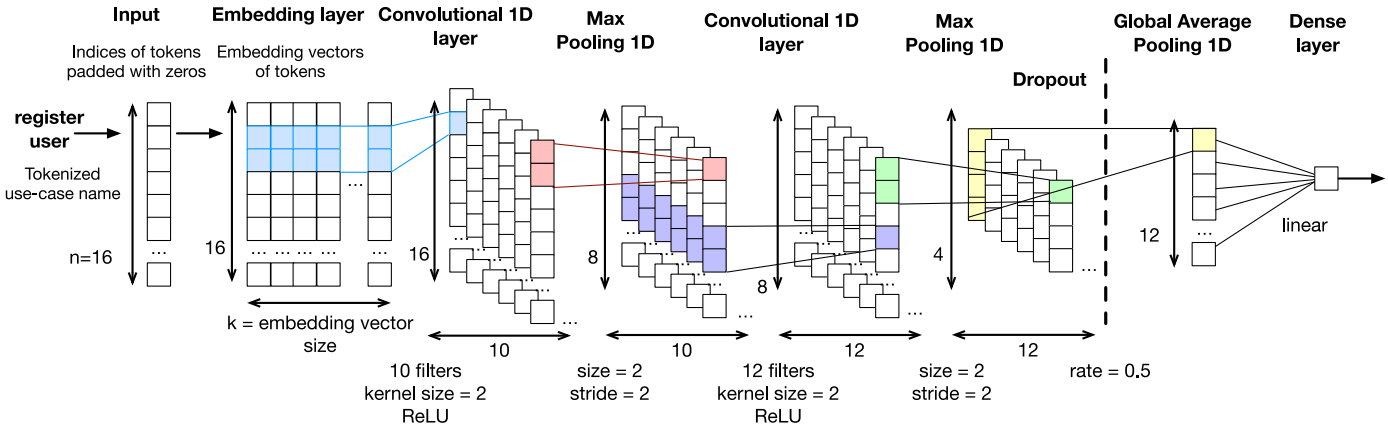

Fig. 7. The architecture of the selected deep-learning COSMIC-size approximation model (DEEP-COSMIC-UC).

Finding 1. Convolutional Neural Network architecture allows achieving higher prediction accuracy for the problem of approximating COSMIC size based on use-case names than the Recurrent Neural Network and Convolutional-Recurrent Neural Network architectures.

In the second iteration, we increased the number of convolutional layers to three. However, we observed that for the considered problem and dataset, the 3-layer architecture provided predictions with worse accuracy than the 2-layer one. Consequently, we decided to focus on optimizing the hyperparameters of the latter. We also considered adding an additional dense layer to the model, however, it did not improve its performance.

In the following iterations, we continue optimizing the models, each time focusing on certain hyperparameters of the models and taking into account the observations made in the previous iterations. As a result of the second phase, we selected three most-promising models and studied their sensitivity to the sampling by repeating the train-test procedure ten times.

The architecture of the selected deep use-case COSMIC-size approximation model (DEEP-COSMIC-UC) is presented in Fig. 7. The input sequence of tokens identifiers is transformed into a sequence of word-embedding vectors in the embedding layer. The output is processed by two convolutional layers, each proceeded with max-over-time pooling. Finally, a dropout and global average pooling is used. These two layers provide regularization for the model and mitigate the risk of its overfitting to the training data. Finally, a dense layer with a linear activation function is used to output the use-case COSMIC functional size.

We used the word embeddings trained on Stack Overflow [67] during the process of selecting the architecture of our prediction model. Since the word-embedding model was trained on the corpus of texts related to software development, its vocabulary and relationship between the terms should be relevant in the context of use cases. However, we observed that the vocabulary used in the names of use cases often goes beyond the computer-science terminology and contains references to the terms used in the business domain of the system being specified. Therefore, we decided to study other state-of-the-practice pre-trained word embeddings to see how they affect the accuracy of the prediction models (see Section 3.3).

We trained the DEEP-COSMIC-UC model presented in Fig. 7 using each of the considered word-embedding models (see Section 3.3). We repeated the train-test splits ten times for each of the models and compared MAEs. The results are presented in Table 2. The word embeddings with the highest dimensionality of vectors seem to allow achieving the best accuracy. The top-three word-embedding models (source-wise) were Common Crawl, Stack Overflow, and Wikipedia+Gigaworld.

In Fig. 8, we visualize how different word embeddings capture the similarities between words appearing in the use-case names in the dataset. We use t-distributed stochastic neighbor embedding (t-SNE) to reduce the dimensionality of word-embedding vectors and use colors to highlight the verbs that represent actions characteristic for a given type of use-case goal while other words are presented as gray dots. We used the classification of verbs from [24], and since that study showed that there is a relationship between the types of use-case goals and COSMIC size, the ability to capture this kind of similarities between the verbs by a word-embedding model could improve the accuracy of the size approximation models.

Table 2
MAE for prediction models using different word-embedding models.

Word embeddings	min. MAE	mean MAE	SD
Common Crawl 840B (300d)	3.66	3.74	0.05
Common Crawl 42B (300d)	3.68	3.73	0.05
Stack Overflow (200d)	3.70	3.74	0.03
Wikipedia+Gigaworld (300d)	3.70	3.77	0.05
Wikipedia+Gigaworld (200d)	3.73	3.83	0.06
Google News (300d)	3.75	3.83	0.06
Wikipedia+Gigaworld (100d)	3.77	3.90	0.08
Twitter (200d)	3.77	3.85	0.06
Twitter (100d)	3.89	4.01	0.08
Wikipedia+Gigaworld (50d)	3.93	4.03	0.06
Twitter (50d)	4.05	4.16	0.05
Twitter (25d)	4.21	4.23	0.02

We assume that a good word-embedding model should cluster verbs that are characteristic for a given type of use case and provide a visible separation between the clusters. By analyzing the plots in Fig. 8, we can see that the Common-Crawl, Wikipedia+Gigaworld, and Stack Overflow word-embedding models provide a very good grouping of use-case-goal-related verbs. However, it seems that the model trained on Stack Overflow, in addition to separating words by meaning, separates them by their part-of-speech, making all the verbs close to each other. The worst clustering could be observed for the Twitter-based word embeddings. We can see that not-related verbs are often much closer to each other than the related ones.

Taking into account the results of the simulation study and visual examination of similarities between the verbs, we give the following answer to the question RQ2:

Finding 2. The Common Crawl 840B/42B (300d; GloVe), Wikipedia+Gigaworld (300d; Glove), and Stack Overflow (200d; word2vec) word-embeddings models are better suited for representing use-case names for COSMIC-size approximation based on their names than the Wikipedia+Gigaworld (200d, 100d, 50d; GloVe), Google News (300d; word2vec), or Twitter (200d, 100d, 50d, 25d; GloVe) word-embeddings models.

At the same time, we want to emphasize that none of the word-embedding models visibly outperformed the other models. The general

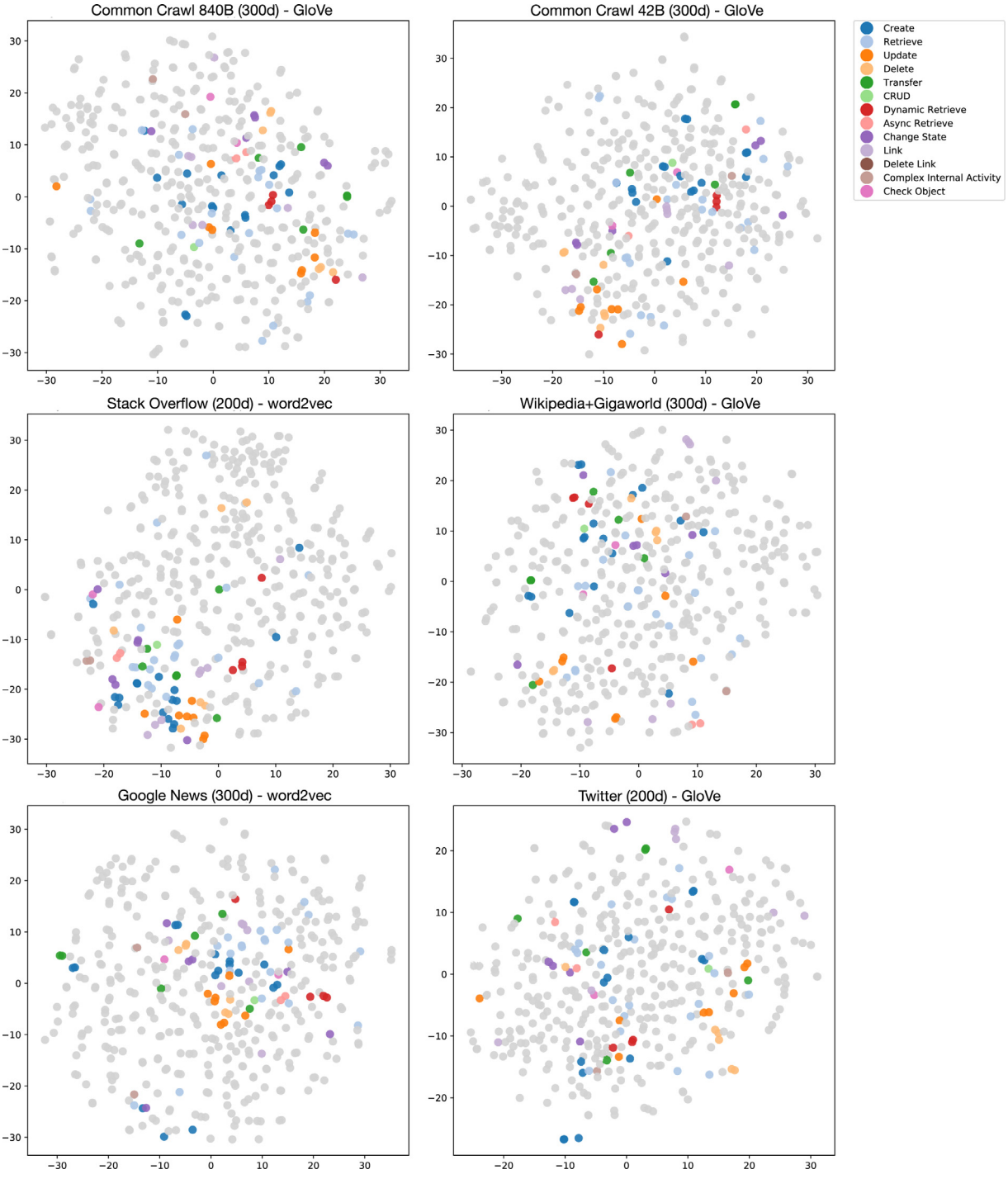

Fig. 8. Distances between the verbs appearing in use-case names that relate to the same types of use-case goals (colored dots) and other words (gray dots) (we use the classification of words from [24]).

observation is that the models using high dimensional vectors (300d) achieved higher accuracy than those using smaller vectors.

5. Validation of DEEP-COSMIC-UC

As we expected, all the prediction models outperformed random guessing (SA = 26–45%).

As it stands from Table 3, the variants of DEEP-COSMIC-UC using different word embeddings were the most accurate in predicting the size of use cases. The lowest prediction error was observed for the variant of the model trained using the Wikipedia+Gigaworld (300d) word em-

Table 3
Prediction accuracy of the prediction models.

Prediction model	SA%	MAE	MdAE
AUC	25.64	4.732	3.794
AUCG	38.45	3.917	2.894
BN-UCGAIN	40.83	3.765	2.765
DEEP-COSMIC-UC:			
– Common Crawl 840B	45.02	3.499	2.284
– Common Crawl 42B	44.70	3.519	2.265
– Stack Overflow	43.56	3.592	2.337
– **Wikipedia+Gigaworld**	**45.33**	**3.479**	**2.204**

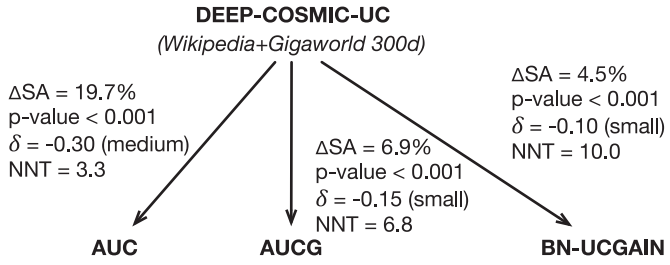

Fig. 9. Direct comparison between the prediction accuracy of DEEP-COSMIC-UC and AUC, AUCG, and BN-UCGAIN.

beddings. However, the differences between the accuracy of the DEEP-COSMIC-UC variants were very small ($\Delta SA < 2\%$).

Fig. 9 shows a direct comparison between the accuracy of the best variant of DEEP-COSMIC-UC and the accuracy of the baseline models: AUC, AUCG, and BN-UCGAIN.

DEEP-COSMIC-UC visibly outperformed its main counterpart AUC by ca. 20% (ΔSA). The observed Cliff's δ can be interpreted as 'medium' effect size and translates to NNT of ca. 3 use cases that need to be estimated to predict the size of at least one of them with a lower prediction error than in the case of using AUC. Therefore, we formulate the following answer to the research question RQ3:

Finding 3. DEEP-COSMIC-UC outperforms the average use-case size approximation method (AUC) in terms of its accuracy in predicting use-case COSMIC size based on use-case names.

Also, we observed that DEEP-COSMIC-UC was more accurate than AUCG and BN-UCGAIN. However, the observed improvements in prediction accuracy were smaller than in the case of AUC. ΔSA ranged between ca. 7% and 5%, respectively for AUCG and BN-UCGAIN. The observed Cliff's δ could be interpreted as indicating 'small' effect size. NNTs were 7 and 10 use cases to be estimated to see an improvement in the accuracy for at least one of them when using DEEP-COSMIC-UC instead of AUCG and BN-UCGAIN. Based on these results, we formulate the following answer to RQ4:

Finding 4. DEEP-COSMIC-UC is slightly more accurate in predicting COSMIC-size of use cases based on their names than the AUCG and BN-UCGAIN models.

While analyzing the distributions of the absolute errors (AE) presented in Fig. 10, we can see that all of them are positively skewed (with some outlying observations). We can see two visible kernels in the error distributions for AUC, AUCG, and BN-UCGAIN, which are less visible in the corresponding distributions for DEEP-COSMIC-UC, making the prediction more consistent for the non-outlying use cases (MdAE equals to 2.2). We analyzed the most extreme outliers and observed that they were mainly CRUD-like use cases (describing multiple goals). In our opinion, it would be difficult to predict their complexity only based on their names since they did not stand out from the names of other multi-goal use cases. For instance, the four use cases for which the highest prediction errors were observed (MAE in 10 runs of cross-validation between 23 and 32) have the following names: "Add a card profile / edit a card profile", "Accept the candidates", "Modify candidate's payments", and "Assign candidates to exam rooms."

6. Discussion

6.1. Implications for practitioners

The proposed DEEP-COSMIC-UC prediction model can be used to approximate COSMIC-size at early stages of software development when only the names of use cases are defined (e.g., in the form of UML use-case diagram) or in agile projects with emergent requirements specifications.

The costs and benefits of adopting and using the model can differ between contexts and organizations. We believe that the two most common scenarios would be to either (S1) use it "as it is" or (S2) to (re-)train/tune it with company-specific data. The first scenario applies to the context of a company that either does not collect historical data about use cases and COSMIC size or it assumes that the use cases authored by its employees are similar to those in our dataset (both size- and writing style-wise).

The potential costs of adopting the model would include the cost of performing the following activities (depending on the scenario):

1. Preparing a training dataset including use-case names and their COSMIC size (S2).
2. Re-training the prediction model (S2).
3. Preparing the input for the model—the list of names of use cases to be measured (S1 and S2).
4. Reporting the approximate COSMIC size (S1 and S2).

The cost of performing the activities (1), (3), and (4) depends on the project-management tools used by a company and the expected level of their integration with DEEP-COSMIC-UC. We could roughly estimate that performing each of these activities can take from a few minutes (e.g., querying task management system for data) to a few man-hours or man-days (e.g., writing a script that would export data from a task management system and transform it to the format required by the model, run DEEP-COSMIC-UC, and return its output back to the task-management system). The cost of re-training the model (2) can differ depending on the size of the training dataset and available hardware. However, we believe that even when using commodity hardware it should not take more than a man-day to perform that task using the scripts provided in the study replication package.

On the benefit side of using DEEP-COSMIC-UC, our study shows that when the model is used instead of the AUC technique (its main counterpart) one should see a nearly instant improvement in the accuracy of size predictions ($NNT \approx 3$). The observed improvement measured in MAE was around 1.25 CFP per use case. Taking into account that the average size of a use case in the dataset is ca. 9 CFP and the average team product delivery rate is ca. 9 h / CFP, the expected on-average reduction in the effort-estimation error at the level of individual use case is ca. 11 h.

The improvement in the accuracy over the AUCG/BN-UCGAIN models is less noticeable and would allow estimating effort required to implement individual use cases with the error lower by (on average) ca. 4 and 2.5 h, respectively. However, the main advantage of DEEP-COSMIC-UC over AUCG/BN-UCGAIN is that it requires only the names of use cases to be collected (and their COSMIC size) to train the model, while the two latter models also require collecting the information about the type of use-case goals (AUCG and BN-UCGAIN) and the information about the types of use-case transactions in scenarios (BN-UCGAIN).

Finally, although this study shows that use-case names provide sufficient information to accurately approximate the size of most of the use cases, there are some use cases which complexity cannot be revealed only based on their names. This observation is convergent with the observation made by Lavazza and Morasca [53], who reported similar issues when using Average Functional Process (AEP) to approximate COSMIC size at the level of functional processes. Therefore, we recommend combining the automatic methods with human judgment when approximating the size of use cases based on their names to increase the chances of identifying extreme outliers.

6.2. Implications for researchers

Our previous study [24] showed that there is a relationship between the types of use-case goals and functional size. The existence of similar relationships at the level of functional process was reported by Trudel et al. [74] who introduced so-called FSM patterns (e.g., Basic Create, Basic Update, Basic Delete, Composite CRUDL-3DG). This study confirms these observations and shows that it is possible to approximate the COS-

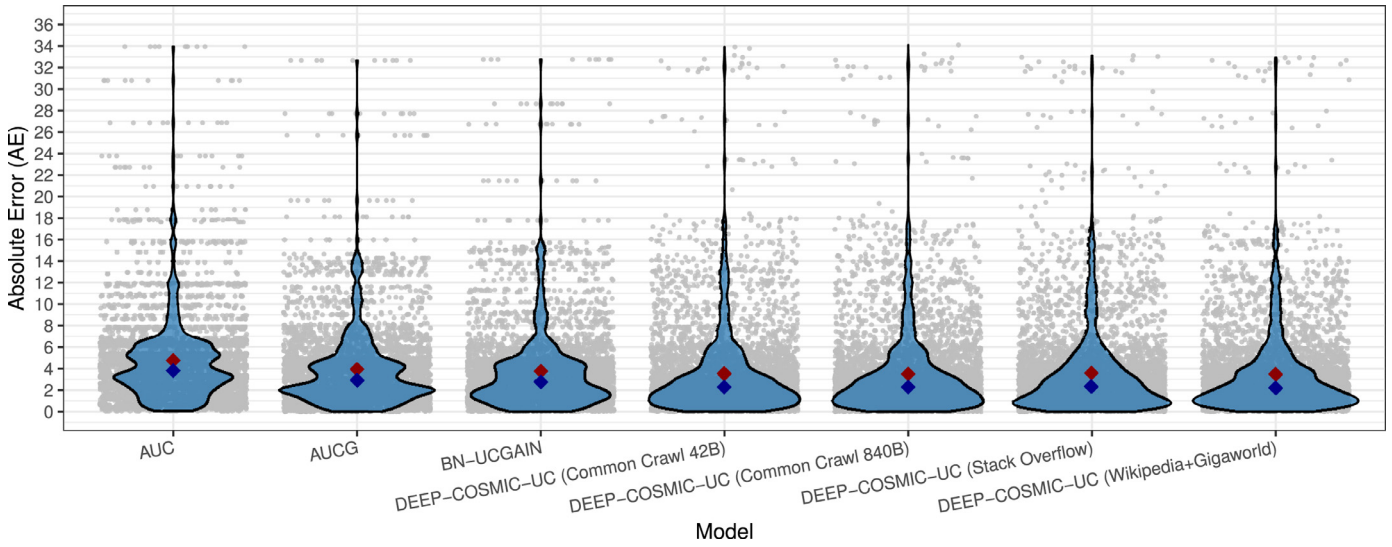

Fig. 10. Violin plots presenting the distributions of Absolute Error of the COSMIC-size approximation for different models in 10 runs of 10-fold cross-validation (mean—red diamond, median—blue diamond). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

MIC size of use cases using word-embedding models without the need of determining the goals of use cases.

Also, we have published the dataset and the source code used in this study. They are available on GitHub allowing other researchers to re-use them in their studies.

6.3. Threats to validity and study limitations

There are several threats to validity and limitations of this study that should be taken into account while interpreting its results. We discuss these threats based on the guidelines provided by Wohlin et al. [75] and the framework dedicated to simulation studies proposed by de França and Travassos [76].

6.3.1. Construct validity

Unfortunately, there are no formal standards on how to author use cases. Use cases can differ visibly among organizations, also when it comes to their level of granularity. That could cause difficulties in measurement and necessity of calibrating use-case based approximation models [13,77,78]. The user-level use cases in the considered sample of projects seemed convergent with the state-of-the-art books by Cockburn [26] and Adolph et al. [29]. Also, although there could be many different approaches to document use-case scenarios, there should not be visible differences in the way people document the names of use cases, as long as they describe the goals of use cases.

When applying the COSMIC methods, measurement specialists have to interpret the rules in the context of a given application. Consequently, depending on the skills of the specialist and quality of available requirements (e.g., their clarity and completeness), we may observe some measurement error being introduced because of the need of making subjective interpretation of the rules in the context. In the dataset we use in this study, the size of use cases was measured by an experienced FSM specialist having more than ten years of experience in applying FSM methods (measuring the size of applications for government and industrial projects, conducting workshops, and providing consulting services), also being certified in IFPUG FPA and COSMIC. This reduces the risk of incorrect application of the COSMIC method.

6.3.2. Internal validity

Since we used a dataset of projects from the previous studies, we automatically inherited some of the threats to validity related to its structure, i.e., the need of translating some of the use-case titles to English, or the heterogeneity of the projects. However, as it was discussed in [24], these threats should not have a visible impact on the results since the style of authoring use case was very similar between the projects.

6.3.3. Conclusion validity

The limited size of the dataset could affect the process of selecting the architecture of DEEP-COSMIC-UC since it could favor the models with a smaller number of parameters. Taking that into account, we decided to formulate our conclusions regarding the architecture of the model in the form of “findings” rather than strong conclusions.

6.3.4. External validity

As we have already mentioned, the way use cases are documented can differ visibly between and within organizations [13,77,78]. This could affect the generalizability of the model trained in our study. However, as long as use-case names express the goals of use cases, it should be possible to re-train the model to be used in a particular context.

There is a risk that tuning and validating a prediction model using the same data could bias the results [65]. Unfortunately, splitting the dataset used in our study in two subsets be used exclusively for tuning the model and validating its accuracy would cause an even bigger threat to the generalizability of our findings since the dataset contains only 437 instances of use cases. Therefore, we decided to mitigate that risk by using two different methods of splitting the dataset while performing each of the tasks. We used train-test split while tuning the models and cross-validation when validating their accuracy. The proportion of the data used for training and testing was also different when performing these two tasks (66% of instances were used for training when tuning the models and 90% when validating their accuracy).

Finally, the dataset used to validate the proposed prediction model contains applications from the Management Information System (MIS) domain. Therefore, we cannot safely generalize our findings to other domains, e.g., telecom, computer games.

7. Conclusions

We proposed a new COSMIC size approximation technique that can approximate COSMIC size based only on use-case names. The new technique uses a prediction model, called DEEP-COSMIC-UC, that has the form of a multi-layer convolutional neural network. By using pre-trained word embeddings, we were able to a model that takes raw text representing use-case name as an input and approximate the COSMIC size of

the use case. Consequently, it could be adopted by organizations that collect historical data about use cases and their COSMIC size without the need for revisiting and manually labeling the collected data.

We compared the accuracy of DEEP-COSMIC-UC with average use-case approximation technique (AUC), which is the main counterpart of the proposed model when it comes to the type of the input they accept, and two previously proposed models—Average Use-Case Goal-aware Approximation (AUCG) and Bayesian Network Use-Case Goal AproxImatioN (BN-UCGAIN). AUCG and BN-UCGAIN are two-step models that firstly predict the type of use-case goal based on the use-case name, and in the second step use this information to predict its size. Unfortunately, one needs to collect the information about use-case goals and structure of use-case scenarios (what is not a common practice) to train these models.

We validated the accuracy of the proposed model based on the dataset of 437 use cases. The results of 10 runs of 10-fold cross-validation showed that DEEP-COSMIC-UC outperforms AUC in terms of prediction accuracy ($\Delta SA = 19.7\%$, Cliff's δ for absolute error = -0.30). An organization that wants to replace AUC with DEEP-COSMIC-UC could see the increase in the accuracy of predictions nearly instantly after switching to the new model (NNT = 3.3). The proposed model was also slightly more accurate than the two-steps model, showing 4.5–6.9% improvement over AUCG and BN-UCGAIN, respectively.

The results of our study show that word-embedding models allow capturing relationship between the meaning of the words that could help in predicting functional size based on use-case names.

As future research, it would be worth investigating the possibility of training a specialized word-embedding model based on texts related to Requirements Engineering and investigate if using such a model to encode use-case names could improve the accuracy of the proposed prediction model. Also, it would be worth investigating the possibility of approximating functional size measured using other FSM methods (e.g., IFPUG FPA) or proposing similar models that could be used to approximate the size of user stories.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Mirosław Ochodek: Conceptualization, Methodology, Software, Validation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Sylwia Kopczyńska:** Methodology, Validation, Writing - review & editing. **Miroslaw Staron:** Methodology, Validation, Writing - review & editing.

Acknowledgments

This research has been partially supported by the statutory funds of Poznan University of Technology.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.infsof.2020.106310.

References

[1] ISO/IEC, ISO/IEC 14143-1:2007(R2019): Information technology — Software measurement — Functional size measurement — Part 1: Definition of concepts, 2007.
[2] A. Albrecht, Measuring application development productivity, Proc. Joint SHARE/GUIDE/IBM Appl. Dev. Symp. (1979) 83–92.
[3] ISO/IEC, ISO/IEC 24570:2005: Software engineering — NESMA functional size measurement method version 2.1 — Definitions and counting guidelines for the application of Function Point Analysis, 2005.
[4] ISO/IEC, ISO/IEC 29881:2010: Information technology — Systems and software engineering — FiSMA 1.1 functional size measurement method, 2010.
[5] C. Symons, Software Sizing and Estimating: Mk II FPA (Function Point Analysis), Wiley-Interscience, 1991.
[6] ISO/IEC, ISO/IEC 19761:2011: Software engineering – COSMIC: a functional size measurement method, 2003.
[7] A. Abran, Software Metrics and Software Metrology, John Wiley & Sons, 2010.
[8] I. Inayat, S.S. Salim, S. Marczak, M. Daneva, S. Shamshirband, A systematic literature review on agile requirements engineering practices and challenges, Comput. Human Behav. 51 (2015).
[9] S. Wagner, D. Méndez-Fernández, M. Kalinowski, M. Felderer, Agile requirements engineering in practice: status quo and critical problems, CLEI Electron. J. 21 (1) (2018) 15.
[10] M. Ochodek, S. Kopczyńska, Perceived importance of agile requirements engineering practices–a survey, J. Syst. Softw. 143 (2018) 29–43.
[11] M. Cohn, Succeeding with Agile: Software Development Using Scrum, Pearson Education, 2010.
[12] M. Cohn, Agile Estimating and Planning, Pearson Education, 2005.
[13] COSMIC, COSMIC Guideline for Early or Rapid Functional Size Measurement using approximation approaches, edited by F. Vogelezang, 2015. doi:10.13140/RG.2.1.4195.0567.
[14] L. Buglione, The next frontier: measuring and evaluating non-functional productivity, Metr. Views, IFPUG Newslett. 6 (2) (2012) 11–14.
[15] L. Buglione, A. Abran, Improving the user story agile technique using the invest criteria, in: 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, IEEE, 2013, pp. 49–53.
[16] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard, Object-Oriented Software Engineering: A Use Case Driven Approach, Addison Wesley Longman, Inc, 1992.
[17] S. Tiwari, A. Gupta, A systematic literature review of use case specifications research, Inf. Softw. Technol. 67 (2015) 128–158.
[18] E.-M. Schön, J. Thomaschewski, M.J. Escalona, Agile requirements engineering: a systematic literature review, Comput. Standards Interf. 49 (2017) 79–91.
[19] K. Wiegers, B. Joy, Software Requirements, third, Microsoft Press, 2013.
[20] Y. Wautelet, S. Heng, D. Hintea, M. Kolp, S. Poelmans, Bridging user story sets with the use case model, in: International Conference on Conceptual Modeling, Springer, 2016, pp. 127–138.
[21] COSMIC, The COSMIC Functional Size Measurement Method version 3.0 Advanced and Related topics, edited by A. Lesterhuis and C. Symons, 2007.
[22] A. Živkovič, I. Rozman, M. Heričko, Automated software size estimation based on Function Points using UML models, Inf. Softw. Technol. 47 (13) (2005) 881–890.
[23] I. Hussain, L. Kosseim, O. Ormandjieva, Approximation of COSMIC functional size to support early effort estimation in Agile, Data Knowl. Eng. 85 (2013) 2–14.
[24] M. Ochodek, Functional size approximation based on use-case names, Inf. Softw. Technol. 80 (2016) 73–88.
[25] M. Ochodek, Approximation of cosmic functional size of scenario-based requirements in agile based on syntactic linguistic features replication study, in: 2016 joint conference of the international workshop on software measurement and the international conference on software process and product measurement (IWSM-MENSURA), IEEE, 2016, pp. 201–211.
[26] A. Cockburn, Writing Effective Use Cases, Addison-Wesley Boston, 2001.
[27] J. Nawrocki, T. Nedza, M. Ochodek, Ł. Olek, Describing business processes with use cases, in: W. Abramowicz (Ed.), Proceedings of the Business Information Systems Conference, Lecture Notes in Informatics, P-85, Koellen Druck+Verlag, 2006, pp. 13–27.
[28] J. Nawrocki, M. Ochodek, J. Jurkiewicz, S. Kopczyńska, B. Alchimowicz, Agile requirements engineering: A research perspective, in: International Conference on Current Trends in Theory and Practice of Informatics, Springer, 2014, pp. 40–51.
[29] S. Adolph, P. Bramble, A. Cockburn, A. Pols, Patterns for Effective Use Cases, Addison-Wesley, 2002.
[30] J. Jurkiewicz, J. Nawrocki, M. Ochodek, T. Głowacki, HAZOP-based identification of events in use cases, Empir. Softw. Eng. 20 (1) (2015) 82–109.
[31] I. Jacobson, I. Spence, B. Kerr, Use-case 2.0., Commun. ACM 59 (5) (2016) 61–69.
[32] COSMIC, The COSMIC Functional Size Measurement Method v4.0.1, Measurement Manual, 2015.
[33] B. Marín, G. Giachetti, O. Pastor, Measurement of Functional Size in Conceptual Models: a Survey of Measurement Procedures Based on COSMIC, in: Software Process and Product Measurement, Springer, 2008, pp. 170–183.
[34] P. Habela, E. Głowacki, T. Serafinski, K. Subieta, COSMIC Function Points: Theory and Advanced Practices, CRC Press.
[35] M. Jenner, COSMIC-FFP and UML: Estimation of the Size of a System Specified in UML–Problems of Granularity, in: Proc. the Fourth European Conference on Software Measurement and ICT Control, 2001, pp. 173–184.
[36] V. Bévo, G. Lévesque, A. Abran, Application de la methode FFP a partir d'une specification selon la notation UML: Compte rendu des premiers essais d'application et questions, 9th International Workshop Software Measurement, Lac Supérieur, Canada, 1999.
[37] A. Sellami, H. Ben-Abdallah, Functional size of use case diagrams: a fine-grain measurement, in: Proceedings of ICSEA'09, IEEE, 2009, pp. 282–288, doi:10.1109/ICSEA.2009.96.
[38] M. Salmanoglu, T. Hacaloglu, O. Demirors, Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points, in: Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, ACM, 2017, pp. 41–49.

[39] M. Cohn, User Stories Applied: For Agile Software Development, Addison-Wesley Professional, 2004.
[40] C. Commeyne, A. Abran, R. Djouab, Effort estimation with story points and COSMIC function points-an industry case study, Softw. Meas. News 21 (1) (2016) 25–36.
[41] J.-M. Desharnais, L. Buglione, B. Kocatürk, Using the COSMIC method to estimate agile user stories, in: Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement, ACM, 2011, pp. 68–73.
[42] J.-M. Desharnais, B. Kocaturk, A. Abran, Using the COSMIC method to evaluate the quality of the documentation of agile user stories, in: 2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement, IEEE, 2011, pp. 269–272.
[43] M. Ecar, F. Kepler, J.P.S. da Silva, COSMIC user story standard, in: International Conference on Agile Software Development, Springer, 2018, pp. 3–18.
[44] F. Vogelezang, T. Prins, S.N. BV, Approximate size measurement with the COSMIC method factors of influence, Proc. SMEF'07 (2007) 167–178.
[45] G.A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.
[46] M. Ochodek, J. Nawrocki, K. Kwarciak, Simplifying effort estimation based on use case points, Inf. Softw. Technol. 53 (3) (2011) 200–213, doi:10.1016/j.infsof.2010.10.005.
[47] M. Ochodek, B. Alchimowicz, J. Jurkiewicz, J. Nawrocki, Improving the reliability of transaction identification in use cases, Inf. Softw. Technol. 53 (8) (2011) 885–897.
[48] S. Bagriyanik, A. Karahoca, Automated COSMIC Function Point measurement using a requirements engineering ontology, Inf. Softw. Technol. 72 (2016) 189–203, doi:10.1016/j.infsof.2015.12.011.
[49] F. Valdés, A. Abran, Industry case studies of estimation models based on fuzzy sets, in: Proceedings of IWSM-MENSURA 2007, 2007, pp. 5–9.
[50] F. Valdés-Souto, Analyzing the performance of two cosmic approximation sizing techniques at the functional process level, Sci. Comput. Program. 135 (2017) 105–121.
[51] G. De Vito, F. Ferrucci, Approximate COSMIC Size: The Quick/Early Method, in: Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on, IEEE, 2014, pp. 69–76.
[52] Z. Mushtaq, A. Wahid, Revised approach for the prediction of functional size of mobile application, Appl. Comput. Inform. (2019).
[53] L. Lavazza, S. Morasca, Empirical evaluation and proposals for bands-based cosmic early estimation methods, Inf. Softw. Technol. 109 (2019) 108–125.
[54] F. Chollet, Deep Learning with Python, Manning Publications Co., 2018.
[55] T. Mikolov, W.-t. Yih, G. Zweig, Linguistic regularities in continuous space word representations, in: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2013, pp. 746–751.
[56] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.
[57] J. Pennington, R. Socher, C.D. Manning, Glove: global vectors for word representation, in: Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.
[58] K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches (2014). arXiv: 1409.1259.
[59] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[60] W. Yin, K. Kann, M. Yu, H. Schütze, Comparative study of cnn and rnn for natural language processing (2017). arXiv: 1702.01923.
[61] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, MIS quarterly 28 (1) (2004) 75–105.
[62] R. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Springer, 2014.
[63] S. Kopczyńska, J. Nawrocki, M. Ochodek, Software development studio—Bringing industrial environment to a classroom, in: Proceedings of EduRex 2012, IEEE, 2012, pp. 13–16, doi:10.1109/EduRex.2012.6225698.
[64] F. Chollet, et al., Keras, 2015, (https://keras.io).
[65] R.B. Rao, G. Fung, R. Rosales, On the dangers of cross-validation. an experimental evaluation, in: Proceedings of the 2008 SIAM international conference on data mining, SIAM, 2008, pp. 588–596.
[66] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization (2014). arXiv: 1412.6980.
[67] V. Efstathiou, C. Chatzilenas, D. Spinellis, Word embeddings for the software engineering domain, in: Proceedings of the 15th International Conference on Mining Software Repositories, ACM, 2018, pp. 38–41.
[68] M. Shepperd, S. MacDonell, Evaluating prediction systems in software project estimation, Inf. Softw. Technol. 54 (8) (2012) 820–827, doi:10.1016/j.infsof.2011.12.008.
[69] W.B. Langdon, J. Dolado, F. Sarro, M. Harman, Exact mean absolute error of baseline predictor, MARP0, Inf. Softw. Technol. 73 (2016) 16–18, doi:10.1016/j.infsof.2016.01.003.
[70] N. Cliff, Dominance statistics: ordinal analyses to answer ordinal questions., Psychol. Bull. 114 (3) (1993) 494.
[71] M.R. Hess, J.D. Kromrey, Robust confidence intervals for effect sizes: acomparative study of Cohen's d and Cliff's delta under non-normality and heterogeneous variances, Annual Meeting of the American Educational Research Association, San Diego, 2004.
[72] B. Kitchenham, L. Madeyski, D. Budgen, J. Keung, P. Brereton, S. Charters, S. Gibbs, A. Pohthong, Robust statistical methods for empirical software engineering, Empir. Softw. Eng. 22 (2) (2017).
[73] H.C. Kraemer, D.J. Kupfer, Size of treatment effects and their importance to clinical research and practice, Biol. Psychiatry 59 (11) (2006) 990–996.
[74] S. Trudel, J.-M. Desharnais, J. Cloutier, Functional size measurement patterns: A proposed approach, in: 2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), IEEE, 2016, pp. 23–34.
[75] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer Science & Business Media, 2012.
[76] B. Bernard Nicolau de França, G. Horta Travassos, Simulation based studies in software engineering: amatter of validity, CLEI Electron. J. 18 (1) (2015) 5.
[77] F. Vogelezang, C. Symons, A. Lesterhuis, R. Meli, M. Daneva, Approximate COSMIC Functional Size–Guideline for Approximate COSMIC Functional Size Measurement, in: Proceedings of IWSM-MENSURA 2013, IEEE, 2013, pp. 27–32.
[78] V. Del Bianco, L. Lavazza, G. Liu, S. Morasca, A.Z. Abualkishik, Model-based early and rapid estimation of COSMIC functional size–An experimental evaluation, Inf. Softw. Technol. 56 (10) (2014) 1253–1267.