BotCFP: A Machine Learning based Tool for COSMIC Chatbots Sizing

Rahma Becha[1,*], Asma Sellami[1,1], Nadia Bouassida[1,1], Ali Idri[2,1] and Alain Abran[3,1]

[1]Mir@cl Laboratory, University of Sfax, ISIMS, BP 242. 3021. Sfax-Tunisia

[2]Software Project Management research Team, ENSIAS, Mohammed V University in Rabat, Morocco

[3]Ecole de technologie supérieure, ETS, Montreal Université du Québec

Abstract

Chatbots are becoming more popular due to the number of functionalities they provide, the time savings and rapid responses in real time. Developing a chatbot requires defining a list of functional requirements upfront. Some of these requirements can be derived from other chatbots to discover and provide the required functionality, while the acquisition of other requirements is time-consuming and costly. Applying a standardized functional size measurement method, such as COSMIC Function Points – ISO 19761, to chatbots requirements is helpful in estimating the related project development effort and duration. This paper proposes an automated tool named BotCFP for generating the chatbots' sizes using the use-cases.csv dataset. Three Machine Learning techniques (Support Vector Machine (SVM), Random Forest (RF), and Gradient Boosting Machine (GBM)) were used to determine the chatbots' sizes based on their functional processes (FP) name using three different text vectorization methods: TF-IDF, Word2Vec, and Bag of words. The best measurement results were provided by Random Forest using TF-IDF text vectorization method, deployed and used as an API in the BotCFP tool. The proposed tool allows users (project managers and developers) to determine the chatbot size from its FPs names before starting the development process.

Keywords

Chatbots, COSMIC Sizing, Machine Learning, TF-IDF, Word2Vec, Bag of Words, API

1. Introduction

Chatbots are artificial intelligent systems based on Natural Language Processing (NLP) and Machine Learning (ML) algorithms, which can be executed in mobile devices or computer devices to simulate human conversations and automate complex tasks. They have become increasingly popular because of their advantages and usefulness in a variety of applications such as customer service, e-commerce, and education [1].

When functional requirements are available, Functional Size Measurement (FSM) methods have been successfully used to determine the software functional size, independently of the technologies used to develop the software [2]. The COSMIC Function Points (CFP) also adopted as ISO 19761 FSM method, is one of the most powerful FSM methods and is referred to as of the 2nd generation of FSM methods. It can be applied to different types of software such as real-time

---

CEUR Workshop Proceedings (CEUR-WS.org)

software, embedded software, and Management Information System (MIS) applications [3]. The COSMIC method has been applied in a variety of applications, such as in [4], [5] [6]. Over the years, several researchers have proposed an automated tool to size the functional requirements using the COSMIC FSM method, such as in [7], [8], [9], [10], [11], [12]. However, none of these have explored the application of the COSMIC FSM method to size the functionality of AI-based systems such as chatbots.

Machine Learning (ML) is a sub-field of Artificial Intelligence, which consists of several algorithms capable of learning complex tasks and build predictive models based on data samples [13]. It is applied in different application domains including healthcare [14], surveillance and security [14], weather forecasting [14], banking [15], software project management and estimation [16], and so on. A number of ML techniques (Support Vector Machine, Random Forest, and Neural Network) have been applied to size the software functionality using the Function Point Analysis (FPA), an FSM of the 1st generation [17]. However, FPA has been criticized because it has been mostly designed to size MIS applications and inadequate to size real-time software and embedded software [18].

The main objective of this study is to size the functionality of chatbots from the name of their functional processes (FP) in CFP units using three ML techniques (Support Vector Machine - SVM, Random Forest - RF, and Gradient Boosting Machine – GBM) and using the use-cases.csv dataset [19]. Thereafter, we propose an automated tool named **BotCFP** to determine the functional size of a newly added FP. This tool will help users (project managers and developers) who have to implement a chatbot within their applications by helping them to figure out what are the main functionalities that must be provided by the chatbot, and to size them early using the COSMIC ISO 19761 FSM method.

This paper is structured as follows: Section 2 presents a background about chatbots as AI-based systems, and an overview of the COSMIC Function Points method. Section 3 presents the works related to COSMIC-based tools developed to automatically generate a COSMIC functional size. Section 4 provides the process for determining the functional size of the chatbots systems using ML algorithms. Section 5 describes the deployment of the ML model and the implementation of the tool and discusses the results. Finally, section 6 summarizes the findings and suggests further research.

2. Background

This section presents the main concepts of chatbots systems and COSMIC Function Points FSM method.

2.1. Chatbots

Chatbots are artificial intelligence systems defined as "a computer program designed to simulate conversations with human users, especially over the internet" [1]. They can simulate human conversation using Natural Language Processing (NLP) to understand human language, then generate the relevant response, which can be text or speech. They can be referred to as a smart bot, interactive bots, digital assistants or artificial intelligence entities. They can be used

in different domains such as education, e-commerce, and business [1]. The chatbots can be classified into the following four categories (see figure 1):
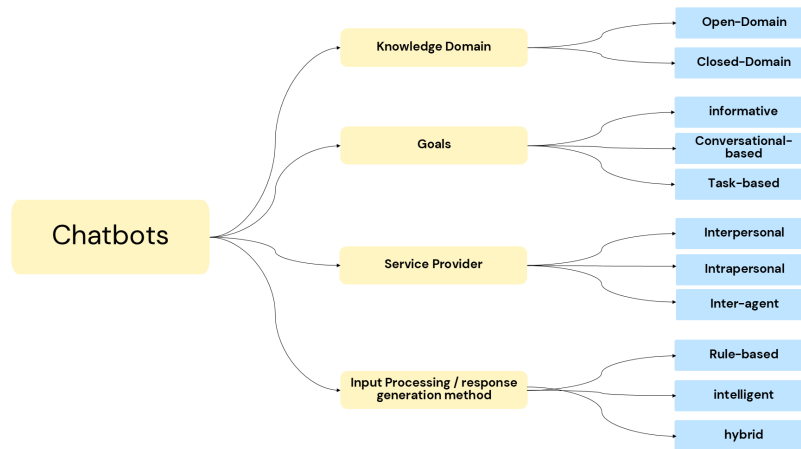

Figure 1: Chatbots classification

- **Knowledge domain:** this category of chatbots focuses on the information that the chatbot can provide or the volume of data it is trained on. They can be open-domain or closed-domain [1].
- **Goals:** this category of chatbots focuses on the main goal they have to accomplish, which can be task-based, informative or conversational-based [1].
- **Service provider:** considers the sentimental proximity of the chatbot to the user. It is also dependent upon the task the chatbot is performing. They can be interpersonal, intrapersonal, and inter-agent [1].
- **Input processing/response generation method:** this category of chatbots focuses on how the input is processed and how the appropriate response is generated. There are three techniques used in this category: rule-based, intelligent, and hybrid [20].

2.2. COSMIC FSM method

The COSMIC Function Points method is the first second-generation FSM method based on the functional user requirements, as specified in ISO 14143-1 [2]. It can be applied to any type of software such as real-time applications, business software applications, and other scientific and engineering software types. The COSMIC method is independent of the technical development and implementation decisions [3]. The application of this FSM method involves the following three phases.

1. **The measurement strategy phase:** in this phase the key parameters of the measurement process will be defined, including the purpose of the measurement, the scope, the functional users of the software and the contextual diagrams [3].
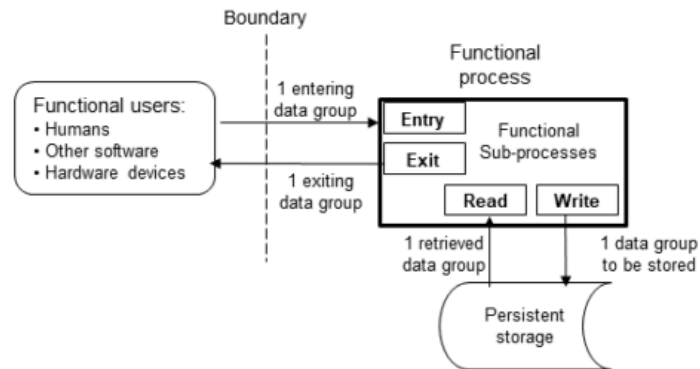
Figure 2: COSMIC Data Movements types [3]

2. **The mapping phase:** in this phase each functional requirement will be mapped into a set of FP. Each FP consists of a set of data movements. There are four data movements types: Entry, eXit, Read, and Write (See figure 2) [3].

3. **The measurement phase**: in this phase each data movement of one data group will be assigned a size of 1 COSMIC Function Point (1 CFP), which is the measurement unit in COSMIC functional size measurement: it corresponds to one data movement of one data group. Then, the functional size of the software will be the sum of all detected data movements of the four types [3].

3. Related work

This section presents the works related to the automated tools developed to determine the functional size using the COSMIC FSM method.

The JavaCFP tool in [7] determines the COSMIC functional size of the developed java source code using Java Swing. JavaCFP is useful for monitoring the completeness of the implemented requirements against the specified requirements, detect any deviation when new functionality has to be implemented, and generate progress reports.

The CFP4J tool in [8] automates the COSMIC FSM for Java web applications using the Spring Web MVC framework. Their contribution involves establishing mapping rules from code and providing a software library to automate the COSMIC functional size of Java web applications using the Spring MVC framework.

The ScopeMaster® commercial tool in [9] measures the COSMIC functional size from textual requirements in English. The tool conducts a series of separate and combined analyses on the textual requirements to identify candidate objects of interest, candidate functional users, candidate data movements, and potential defects.

The tool in [10] generates the COSMIC functional size of C programming language code

source. The authors established a mapping rule from regular expressions and provide a tool using python programming language.

The tool in [11] determines the COSMIC functional size of UML activity diagrams and component diagrams by defining mapping rules. The proposed tool could be used not only for size measurement, but also for verifying the consistency between activity and component diagrams.

An automated functional size measurement tool of embedded systems documented in [12] can accept XML documents from three different notations such as UML, SysML, and Petri net. The authors translated the XML documents to the COSMIC functional size measurement by using mapping rules with the case study of a rice cooker system.

The J-UML COSMIC tool in [21] measures the COSMIC functional size of UML diagrams (e.g., use cases models, package diagrams, class diagrams, sequence diagrams, activity diagrams and component diagrams). The authors established mapping rules between the UML artifacts and the COSMIC concepts, and next verified the accuracy of the developed tool using two case studies (e.g., "Web Advice Module" and "Course registration system").

None of the COSMIC FSM tool proposals attempted to automatically size AI-based systems.

4. Sizing chatbots using ML techniques

As described in section 1, our main objective is to determine the COSMIC functional size of a FP based on its name. This section presents the process we followed to build the ML models to determine the COSMIC functional size of a chatbot Functional Process (FP) based on its name. This sizing process includes the following steps - see Figure 3.
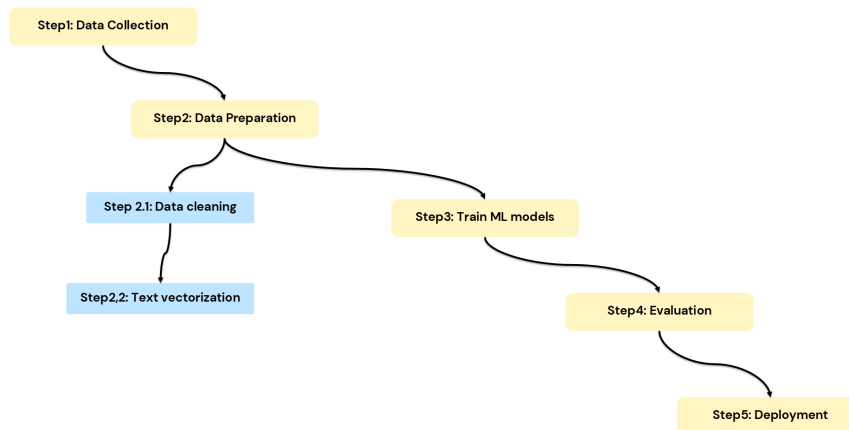

Figure 3: COSMIC sizing using ML

1. **Data collection:** In this step, we collect the dataset relevant to our problem - see section 4.1.

2. **Data preparation:** This step aims to prepare the data in the appropriate format for the ML algorithms, and it consists of two sub-steps: data cleaning and text vectorization - see section 4.2.
3. **Train ML models:** In this step, we conduct experiments to build three sizing models based on three ML algorithms for regression problems (Support Vector Machine, Random Forest, and Gradient Boosting Machine). This is essential for getting the chatbot size from the name of its FPs in terms of CFP units - see section 4.3.
4. **Evaluation:** to evaluate the selected ML algorithms we use the Mean Absolute Error (MAE), the Mean Squared Error (MSE), and the Magnitude Relative Error (MRE) - see section 4.4.
5. **Deployment:** the most relevant algorithm results is chosen for the deployment phase: the selected algorithm is used as an API (Application Programming Interface) for the tool implementation - see Section 5.

4.1. Data collection

The data collection step is the first step in every ML project. It is considered as the foundation and the baseline of the ML project. It can help in understanding the problem and the required outcome to develop high quality ML models.

The dataset used in this step is use-cases.csv which includes 437 uses cases derived from 27 MIS applications [19]. This dataset was selected because we believe that use-cases represent FPs as they describe the behavior of the software system [22]. Table 1 presents some of the use-cases within this dataset, as well as their corresponding COSMIC functional sizes in CFP units as determined by Ochodek et al [19].

Table 1
Some of the use-cases sizing included in the use-cases.csv dataset

Use Cases	CFP
Add candidate	4
Remove account	6
Login	3
Add course	4
Register new user	6
Add group	4
Change group	6
Delete group	4
View list employees	6

4.2. Data preparation

This step aims to clean and prepare the dataset in order to apply the ML algorithms. This step is subdivided into two sub-steps: data cleaning and text vectorization.

4.2.1. Data Cleaning

Data cleaning is crucial in building ML algorithms. We used Natural Language Processing to clean up the dataset through four sub-steps: (i) convert any upper-case character of the FP name to lower-case, automatically (ii) tokenize the FP (split the FP into a set of small units called tokens) (iii) remove stop words (e.g., "a", "this", "as", etc.) (iv) lemmatize the FP, that is finding the root word by considering the vocabulary (e.g., good, better, or best is lemmatized into good) [23].

4.2.2. Text vectorization

After cleaning and splitting the use cases into a set of tokens, it is important to convert them into an appropriate format for the ML algorithms. Since ML algorithms are unable to analyze and comprehend these tokens, the most famous and used text vectorization techniques are: TF-IDF, Word2vec, and Bag of words.

- **TF-IDF:** stands for Term Frequency-Inverse Document Frequency. It is a statistical technique that considers the frequency of occurrences of a term to determine its importance in a particular document within the corpus [24].
- **Word2Vec:** it is a technique of word embedding that aims to represent the words into vectors of numerical values. This technique of word embedding can capture the similarities between the words from the training of a large corpus. The similar words are therefore grouped in the same block and have the same vector values [25].
- **Bag of words:** it is a simple technique to represent words into vectors. The words are represented in a vector of n elements. These n elements represent all the words discovered in the dataset and cataloged in a dictionary. Each n element receives a number y which can be the presence or the absence of n in the instance [26].

Each of the text vectorization technique has its own way to convert the text into the appropriate format required by the ML algorithms. For that reason, we deployed these three methods to find out the best performing one.

4.3. Model training

This subsection presents the implementation of ML models that generate the COSMIC functional size of each chatbot based on the name of its FPs. To address this regression problem, we selected the following ML regressors:

- **Support Vector Machine – SVM**: it is a very popular supervised ML algorithm introduced by Vapnik in 1995 [27]. It can be used to solve the regression and classification problems. Hyperplanes are used in this algorithm to separate the data by maximizing the margin distance between the nearest data points. Several Kernel functions (e.g., linear kernel, non-linear kernel, polynomial kernel, radial basis kernel, and sigmoid kernel) can be applied to produce the best hyperplane and make the data separable by increasing the margin distance.

- **Random Forest - RF:** it is an ensemble learning algorithm which can be used to address both the regression and classification problems. It uses a set of multiple decision trees and a technique of Bootstrap and aggregation. The Bootstrap step consists of splitting randomly the dataset rows and features to create a sample dataset for each tree. The aggregation step consists of combining the prediction result of all trees, and therefore calculate the final prediction by voting for classification problems or by averaging for regression problems [28].
- **Gradient Boosting Machine - GBM:** it is a ML technique which can be used to solve both the regression and classification problems by building a stronger learner using weak learners. It operates in an iterative process. In each iteration, a new model is trained to minimize the error committed in the previous models. The predicted model will then be added to the ensemble (a set of M trees). And the process will be repeated until a stop criterion is reached [28].

We used the K-fold cross validation to split the dataset into K-equal subsets (K=10 in our case), The model will be trained K times. In each iteration, K-1 subset will be used for training and one subset will be used for testing.

Table 2 presents the hyperparameters used for each ML algorithm. We used the linear kernel to build the Support Vector Regressor with a complexity of 2.0 and epsilon of 0.2. For Random Forest Regressor and the Gradient Boosting Regressor, we used a number of estimators of 100 that present the number of trees in the ensemble, a max_depth with None value that presents the maximum depth of each tree in the ensemble, and the squared_error for the loss function.

Table 2
ML hyperparameters

ML algorithm	Hyperparameters
Support Vector Regressor (SVR)	$kernel =' \ linear', C = 2.0, epsilon = 0.2$
Gradient Boosting Regressor (GBR)	$n\_estimators = 100, max\_depth = None, loss =' \ squared\_error'$
Random Forest Regressor (RFR)	$n\_estimators = 100, max\_depth = None, criterion =' \ squared\_error'$

4.4. Model evaluation

This sub-section presents the evaluation of the three selected ML algorithms for each selected text vectorization technique. The three following criteria were used to evaluate the obtained results: Mean Absolute Error (1), Mean Squared Error (2) and Magnitude Relative Error (3) defined by equations 1, 2, and 3, respectively.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{1}$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2}$$

$$MRE = \frac{1}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i} \tag{3}$$

4.5. Results

Table 3 presents the results of the ML models for each text vectorization technique:

- The SVR has the lowest MAE, MSE, and MRE in TF-IDF text vectorization technique. It also has the lowest MAE and MSE in Bag of words.
- The RFR has the lowest MRE in Bag of Words text vectorization technique and the lowest MAE, MSE, and MRE in Word2Vec text vectorization technique.

Of course, these results were derived from the use of the use-cases extracted from MIS applications.

To make sure what is the most accurate ML model for chatbots systems, we tested 51 FPs extracted from chatbots systems on these ML models to determine their functional sizes. We compared the actual functional sizes with their corresponding generated functional sizes. Note that actual functional sizes are determined by following the COSMIC FSM rules and guidelines, and validated by the COSMIC certified experts. Table 4 presents the name and the actual functional sizes of some FPs that have been measured manually.

Table 5 lists the MAE, the MSE, and the MRE of the generated sizes and the actual sizes among the 51 FPs of the chatbots systems. The results showed that Random Forest has the lowest error in TD-IDF, Bag of words, and Word2Vec techniques.

Table 3
Performance evaluation of Machine Learning techniques using use-cases.csv dataset

	TD-IDF			Word2Vec			Bag of Words		
	MAE	MSE	MRE	MAE	MSE	MRE	MAE	MSE	MRE
SVR	1.74	4.91	31.33	1.31	2.58	23.76	1.82	5.56	32.90
GBR	1.94	6.86	33.83	1.72	4.54	30.40	2.00	6.98	35.49
RFR	1.79	5.32	31.92	1.28	2.48	22.80	1.83	5.69	31.94

Table 4
Actual CFP results (determined manually) of some chatbots FPs

FP	CFP
Register with email	6
Enter user prompt	7
Add feedback	4
Delete conversation	4
View conversation List	6
Login with email	4
View account info	6
Share Conversation	6
Rename Conversation	4

Table 5

Performance evaluation using ML Techniques of chatbots systems - N=51 FPs

	TD-IDF			Word2Vec			Bag of Words		
	MAE	**MSE**	**MRE**	**MAE**	**MSE**	**MRE**	**MAE**	**MSE**	**MRE**
SVR	1.22	2.74	28.81	1.45	3.02	35.48	1.33	2.96	31.54
GBR	1.45	4.19	32.54	1.69	4.58	38.90	1.19	3.22	28.77
RFR	**1.14**	**2.40**	**26.71**	**1.45**	**2.91**	**35.10**	**1.17**	**2.25**	**27.64**

5. Tool Development and Illustration

This section presents the implementation of our ML tool named **BotCFP** and the results discussion.

5.1. Tool Development

This sub-section presents the implementation of the tool which automates the functional size measurement of chatbots using ML techniques.

Two users' roles are involved in the tool:

- The super admin role is implemented to add some chatbots and their functional processes, generate their functional sizes, and compare the generated size with the actual size (manually sized by experts measurers).
- The project manager role is implemented to allow the project manager to identify the requirements the most important to be included within any chatbots, add their own chatbots, and generate their associated functional sizes.

Figure 4, 5, and 6 present some user interfaces of our automated tool. For instance, Figure 4 shows the dashboard of the super admin. Through this interface, the super admin can see the following features:

- How many chatbots within the database will be considered for measurement?
- How many FPs per chatbot is included within the database?
- The percentage of correctly determined functional sizes.
- An area chart illustrates the performance of the deployed ML model.
- A line chart illustrating the difference between the actual functional size of the chatbot with its corresponding size using ML (see Figure 7).

Figure 5 lists the existing chatbots. In this interface, the super admin and the project manager can:

- View the list of chatbots.
- Add or edit new chatbot by entering the chatbot name, photo, type (e.g., available for web and-or mobile apps) and actual functional size (only for super admin).
- Delete a chatbot.

Figure 6 presents the chatbot details. The super admin and the project manager can:
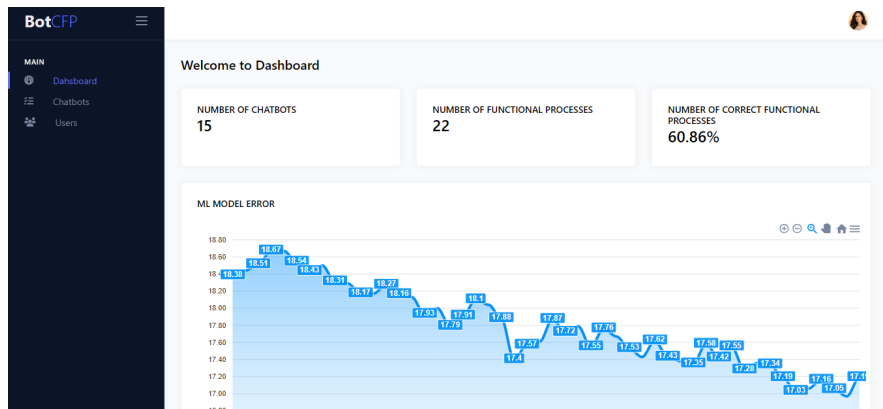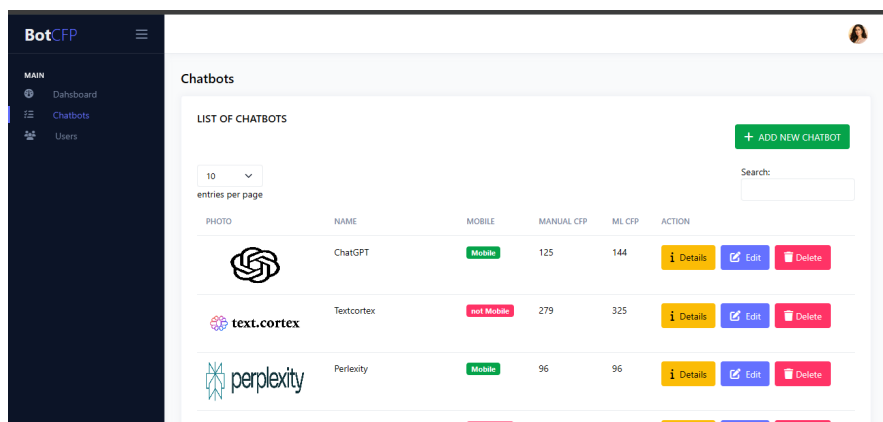
Figure 4: BotCFP Tool Dashboard

Figure 5: BotCFP Tool Chatbot List

- View the chatbot details such as its name, type (i.e., available for web and-or mobile apps), the actual functional size (only for super admin), and the generated functional size.
- View the list of FPs where each FP is characterized by its name, actual functional size (only for super admin), generated functional size, and the error between the generated size and the actual size (only for super admin).
- Add or edit a FP by entering the name of FP and the actual size (only for super admin).
- Delete a FP.

We added 15 chatbots to our tool and determine their functional sizes based on their corresponding FPs names. Figure 7 presents the difference between the actual functional size with the generated functional size for each chatbot. For instance:

- ChatGPT has an actual size of 125 CFP, while its generated size is 144 CFP.
- Tabnine has an actual size of 42 CFP, while its generated size is 43 CFP.
- Perplexity has an actual size of 96 CFP, while its generated size is 96 CFP.
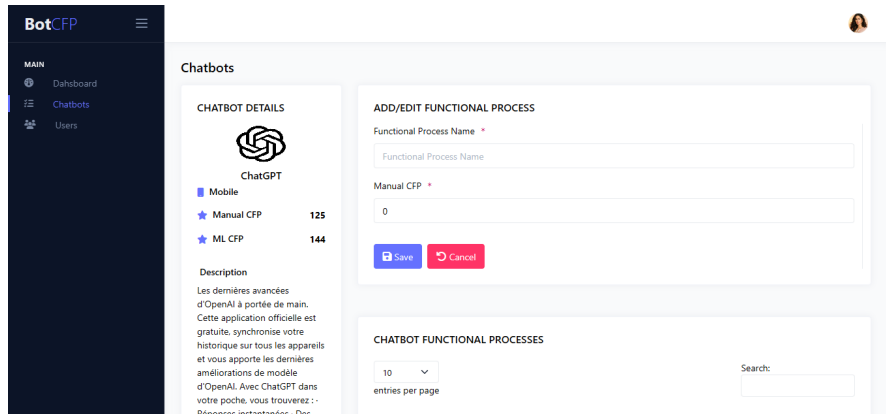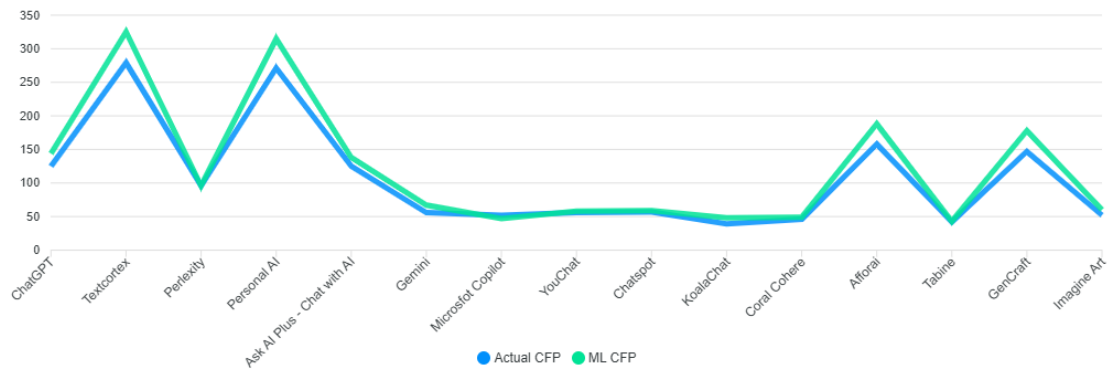
Figure 6: BotCFP Tool Chatbot details

Figure 7: Actual CFP VS. BotCFP's CFP

- Afforai has an actual size of 158 CFP, while its generated size is 188 CFP.

Figure 8 presents the percentage of the error rate of determined FPs size. In summary:

- 60.9% of the determined FPs have a size error between zero and one CFP.
- 36.4% between two and three CFP.
- 2.75% are equal to or greater than four CFP.

Figure 9 shows the dashboard of the project manager. Through this interface, the project manager can see the following features:

- The number of chatbots in the organization.
- The number of FPs per chatbot within the database.
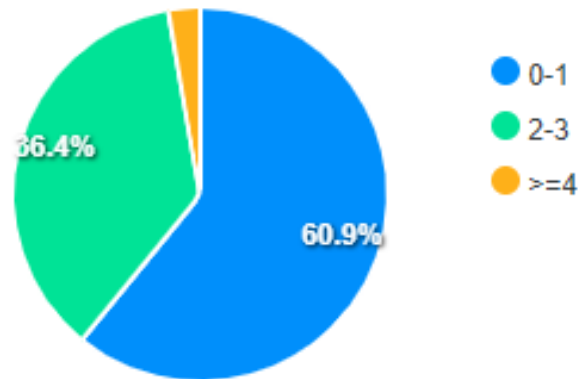- a bar chart of the most frequent requirements available within other chatbots.

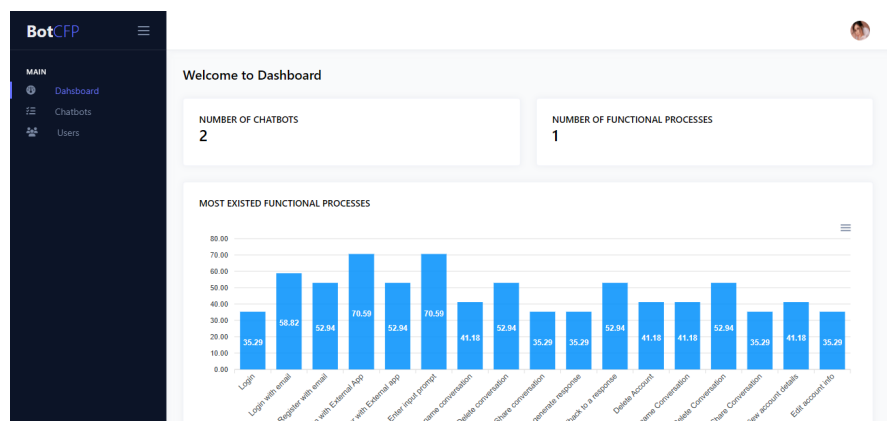**Figure 8:** Determined functional size error from BotCFP dashboard

Figure 9: BotCFP - project manager dashboard

5.2. Discussion

The proposed ML tool can be used to determine the COSMIC functional size of chatbots based on their FPs. After adding a set of FPs in the database and determining their functional sizes using the ML API, we noticed a small difference between the generated functional size and the actual size (determined manually):

- Difference of zero between the generated size and the actual size of the FP. This indicates that both the generated and actual sizes are the same.
- Difference of one CFP between the generated and the actual size of the FP, which can be

minus one CFP or plus-one CFP. If it is minus one, it can be explained by the fact that ML algorithm considered the Error/Confirmation message with a size of 1 CFP in the COSMIC FSM method. If it is plus-one, this indicates that the ML algorithm has added 1 CFP for one data movement not detected in the manual measurement process.

- Difference is greater than 1 or more CFPs between the generated size and the actual size of the FP. This may be because the FP is a new requirement and does not exist in the dataset. It also may be explained because of the distinct contextual diagrams used as a basis to measure the functional size of the FPs from the dataset or manually.

6. Conclusion

In this study, we applied three ML techniques: Support Vector Machine, Random Forest, and Gradient Boosting Machine using the use-cases.csv dataset to build measurement models. We also used three text vectorization techniques: TF-IDF, Word2Vec, and Bag of words to facilitate data processing by the three selected ML techniques.

The outcomes of the three models are the chatbots' sizes in CFP units, which are derived from the names of their corresponding FPs. The most accurate model is Random Forest with the use of TF-IDF technique deployed to implement the **BotCFP** tool.

Most of the obtained results were relevant, indicating minimal discrepancies between generated and actual sizes.

As future work, we intend to use another dataset with effort data to extend our tool to estimate the effort and the duration required for developing chatbots systems and build a project development planning. We also plan to extend this tool to determine the functional size of other types of AI-based systems.

References

[1] E. Adamopoulou, L. Moussiades, An overview of chatbot technology, in: IFIP international conference on artificial intelligence applications and innovations, Springer, 2020, pp. 373–383.

[2] ISO/IEC14143-1, ISO/IEC 14143-1: Information Technology - Software Measurement - Functional Size Measurement. Part 1: Definition of Concepts, 2007.

[3] The COSMIC Functional Size Measurement Method, Version 5, Measurement Manual, 2020. URL: https://cosmic-sizing.org/measurement-manual/.

[4] L. D'Avanzo, F. Ferrucci, C. Gravino, P. Salza, Cosmic functional measurement of mobile applications and code size estimation, in: Proceedings of the 30th annual ACM symposium on applied computing, 2015, pp. 1631–1636.

[5] N. A. S. Abdullah, N. I. A. Rusli, M. F. Ibrahim, A case study in cosmic functional size measurement: angry bird mobile application, in: 2013 IEEE Conference on Open Systems (ICOS), IEEE, 2013, pp. 139–144.

[6] M. Haoues, A. Sellami, H. Ben-Abdallah, A rapid measurement procedure for sizing web and mobile applications based on cosmic fsm method, in: Proceedings of the 27th

International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, 2017, pp. 129–137.

[7] N. Chamkha, A. Sellami, A. Abran, Automated cosmic measurement of java swing applications throughout their development life cycle., in: IWSM-Mensura, 2018, pp. 20–33.

[8] A. Sahab, S. Trudel, Cosmic functional size automation of java web applications using the spring mvc framework., in: IWSM-Mensura, 2020.

[9] E. Ungan, C. Hammond, A. Abran, Automated cosmic measurement and requirement quality improvement through scopemaster® tool., in: IWSM-Mensura, 2018, pp. 1–13.

[10] D. K. Moulla, E. Mnkandla, H. Soubra, A. Abran, Automated cosmic function points measurement for c program using regular expressions (2022).

[11] A. Sellami, M. Haoues, H. Ben-Abdallah, Automated cosmic-based analysis and consistency verification of uml activity and component diagrams, in: Evaluation of Novel Approaches to Software Engineering: 8th International Conference, ENASE 2013, Angers, France, July 4-6, 2013, Revised Selected Papers 8, Springer, 2013, pp. 48–63.

[12] T. Zaw, S. Z. Hlaing, M. M. Lwin, K. Ochimizu, Automated Size Measurement of Embedded System based on XML using COSMIC FSM, Ph.D. thesis, MERAL Portal, 2018.

[13] N. Muthukrishnan, F. Maleki, K. Ovens, C. Reinhold, B. Forghani, R. Forghani, Brief history of artificial intelligence, Neuroimaging Clinics of North America 30 (2020) 393–399. URL: https://www.sciencedirect.com/science/article/pii/S105251492030054X. doi:https://doi.org/10.1016/j.nic.2020.07.004, machine Learning and Other Artificial Intelligence Applications.

[14] A. S. POTHEN, Artificial intelligence and its increasing importance, "Success is no accident. It is hard work, perseverance, learning, studying, sacrifice and most of all, love of what you are doing or learning to do". (2022) 74.

[15] A. Valavanidis, Artificial intelligence (ai) applications. The most important technology we ever develop and we must ensure it is safe and beneficial to human civilization I (2023) 1–49.

[16] P. Pospieszny, B. Czarnacka-Chrobot, A. Kobylinski, An effective approach for software project effort and duration estimation with machine learning algorithms, Journal of Systems and Software 137 (2018) 184–196.

[17] L. Lavazza, A. Locoro, G. Liu, R. Meli, Estimating software functional size via machine learning, ACM Transactions on Software Engineering and Methodology 32 (2023) 1–27.

[18] A. Z. Abualkishik, J.-M. Desharnais, A. Khelifi, A. A. Abd Ghani, R. Atan, M. H. Selamat, An exploratory study on the accuracy of fpa to cosmic measurement method conversion types, Information and Software Technology 54 (2012) 1250–1264.

[19] M. Ochodek, S. Kopczyńska, M. Staron, Deep learning model for end-to-end approximation of cosmic functional size based on use-case names, Information and Software Technology 123 (2020) 106310.

[20] O. Trofymenko, Y. Prokop, O. Zadereyko, N. Loginova, Classification of chatbots, System technologies 2 (2022) 147–159.

[21] G. De Vito, F. Ferrucci, C. Gravino, Design and automation of a cosmic measurement procedure based on uml models, Software and Systems Modeling 19 (2020) 171–198.

[22] A. Sellami, H. Ben-Abdallah, Functional size of use case diagrams: a fine-grain measurement, in: 2009 Fourth International Conference on Software Engineering Advances, IEEE,

2009, pp. 282–288.

[23] A. Kulkarni, A. Shivananda, Natural language processing recipes, Springer, 2019.

[24] H. M. al Khateeb, G. Epiphaniou, How technology can mitigate and counteract cyber-stalking and online grooming, Computer Fraud & Security 2016 (2016) 14–18.

[25] D. Jatnika, M. A. Bijaksana, A. A. Suryani, Word2vec model analysis for semantic similarities in english words, Procedia Computer Science 157 (2019) 160–167.

[26] M. Zampieri, Using bag-of-words to distinguish similar languages: How efficient are they?, in: 2013 IEEE 14th international symposium on computational intelligence and informatics (CINTI), IEEE, 2013, pp. 37–41.

[27] V. Vapnik, The nature of statistical learning theory, Springer science & business media, 2013.

[28] S. B. Nadkarni, G. Vijay, R. C. Kamath, Comparative study of random forest and gradient boosting algorithms to predict airfoil self-noise, Engineering Proceedings 59 (2023) 24.