Toward a Deep Contextual Product Recommendation for SO-DSPL Framework

Najla Maalaoui[a], Raoudha Beltaifa[b] and Lamia Labed Jilani[c]
RIADI Lab., National School of Computer Sciences, Manouba University, Tunisia

Keywords: Service Oriented Dynamic Software Product Lines, Recommender System, User Requirements, Ontology, Deep Neural Network.

Abstract: Today's demand for customized service-based systems requires that industry understands the context and the particular needs of their customers. Service Oriented Dynamic Software Product Line practices enable companies to create individual products for every customer by providing an interdependent set of features presenting web services that are automatically activated and deactivated depending on the running situation. Such product lines are designed to support their self-adaptation to new contexts and requirements. Users configure personalized products by selecting desired features based on their needs. However, with large feature models, users must understand the functionalities of features and the impact of their gradual selections and their current context in order to make appropriate decisions. Thus, users need to be guided in configuring their product. To tackle this challenge, users can express their product requirements by textual language and a recommended product will be generated with respect to the described requirements. In this paper, we propose a deep neural network based recommendation approach that provides personalized recommendations to users which ease the configuration process. In detail, our proposed recommender system is based on a deep neural network that predicts to the user relevant features of the recommended product with the consideration of their requirements, contextual data and previous recommended products. In order to demonstrate the performance of our approach, we compared six different recommendation algorithms in a smart home case study.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

On-time and On-demand service-based systems are constantly increasing from day to day. Thereby, software application providers have to take drastic measures to speed up their development process in order to survive in the ever-demanding competitive software market and deliver software systems according to specific customer needs . Faced with such a challenge, Service Oriented Architecture (SOA) provides a promising means for supporting continuously changing of customers' requests, needs, requirements and contexts. To address this issue, the reuse approach has been suggested as one of the pioneer solutions. The Service Oriented Dynamic Software Product Line (SO-DSPL) technique has already been adopted as one of the most promising techniques for reuse. This framework is addressed by combining SOA with Dynamic Software Product Lines Engineering (DSPLE). A SO-DSPL (Capilla et al., 2014) is known as a family of service-oriented systems sharing a set of common features and managing a set of variable features, which are managed according to the needs of a specific market segment and/or environment by its automatically activation and deactivation according to the running situation. SO-DSPLs support their self-adaptation to new contexts and requirements. In SO-DSPL framework, deriving a customized software refers to the dynamic configuration process that is defined by the decision-making process of dynamically selecting/activating/deactivating a set of features from the product line that complies with the feature model (FM) constraints and fulfill the product's requirements and the running context.

To enable such a dynamic configuration, several approaches have been proposed (Jonathan et al., 2005). Nevertheless, the applicability of these approaches is still limited. In particular, for SO-DSPLs with a huge exponential configuration space, they have shown to be infeasible. The diversity of options provided by the product line and their sensitivity to

[a] https://orcid.org/0000-0002-3896-920X
[b] https://orcid.org/0000-0003-4096-5010
[c] https://orcid.org/0000-0001-7842-0185

Maalaoui, N., Beltaifa, R. and Jilani, L.
Toward a Deep Contextual Product Recommendation for SO-DSPL Framework.
DOI: 10.5220/0011855800003464
In *Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2023)*, pages 138-148
ISBN: 978-989-758-647-7; ISSN: 2184-4895
Copyright © 2023 by SCITEPRESS – Science and Technology Publications, Lda. Under CC license (CC BY-NC-ND 4.0)

different factors, such as the user's context and the context of the DSPL, make configuration more difficult. With the large number of features, the user may have difficulty understanding the features, the relationships between them and the constraints. To tackle these challenges, expressing user needs in textual format became a necessity to help them, furthermore recommendation techniques have become also essential to efficiently filter the huge amount of SPL variants and suggest to the user suitable products. In recommender systems, user requirements may be inferred from consumption patterns and contextual information can be used to ensure relevant product recommendations (Pereira et al., 2016).

In this paper, we propose a deep-learning based approach encompasses a requirement-based recommender system that suggest to the user a product based on its textual product requirements and its contextual information. The main contribution of this paper is to ease the configuration process by effectively predicting which combination of feature is the best suited to users, based on their requirements that are collected in several ways. In addition, domain experts and product developers can also take advantage of this approach as it offers a simple configuration process. In summary, we provide these main contributions:

- We propose a requirement-based personalized recommender system that suggest to users the product configuration based on their given textual requirements and their contextual information. The system is based on the products chosen by previous users to generate personalized recommendations for a current user.
- We reuse a set of swrl rules proposed by DSPL sub-ontology (Maalaoui et al., 2021) in order to verify the validation of the product configuration with the DSPL constraint and context.
- As a pre-treatment phase, we use the approach presented in (Maalaoui et al., 2022) to extract the core requirements from the user requirements in order to have data that follows the same pattern and consequently have a reliable recommendation.
- We empirically evaluate the performance of our proposed recommender system on "smarthome" dataset (Murukannaiah et al., 2016) that contain crowd user requirements with different additional information. We extend the given data by other contextual data based on a proposed SO-DSPL ontology (Maalaoui et al., 2021) and we extract manually the associated products (the set of selected features) in order to evaluate our proposed approach.

2 BACKGROUND

2.1 Software Product-Line Engineering

Software Product-line engineering is a paradigm within software engineering, used to define and derive sets of similar products from reusable assets (Capilla et al., 2014). The development life cycle of a product line encompasses two main processes: domain engineering and application engineering (Capilla et al., 2014). While domain engineering focuses on establishing a reuse platform, application engineering is concerned with the effective reuse of assets across multiple products. Domain engineering is responsible of defining all common and variable assets of a product line and their respective interdependencies. The aim of the application engineering process is to derive specific applications by exploiting the variability of the software product line. Feature modeling is the main activity to represent and manage PL requirements as reusable assets by allowing users to derive customized product configurations (Capilla et al., 2014). Product configuration refers to the decision-making process of selecting an optimal set of features from the product line that comply with the feature model constraints and fulfill the product's requirements (Bashari et al., 2017). A common visual representation for a feature model is a feature diagram (Kang et al., 1990). The feature diagram defines common features found in all products of the product line, known as mandatory features, and variable features that introduce variability within the product line, referred to as optional and alternative features. In addition, feature diagrams often contain cross-tree constraints: a feature can include or exclude other features by using requires or excludes constraints, respectively. Figure 1 shows an example of a feature diagram associated to the Smart Home SO-DSPL.

2.2 Service Oriented Dynamic Software Product-Line Engineering

SPL approaches moved to recent development approaches like Dynamic Software Product Lines as an emerging paradigm to handle variability at runtime and at any time. Dynamic software product lines are able to dynamically switch an executing system from one variant to another, without stopping its execution, without violating its feature model's constraints and without degrading its functional and nonfunctional behavior. Consequently, the feature model itself must become a runtime entity that can dynamically evolve to satisfy new variability dimensions in the system. Dynamic Software Product Lines (SPLs)

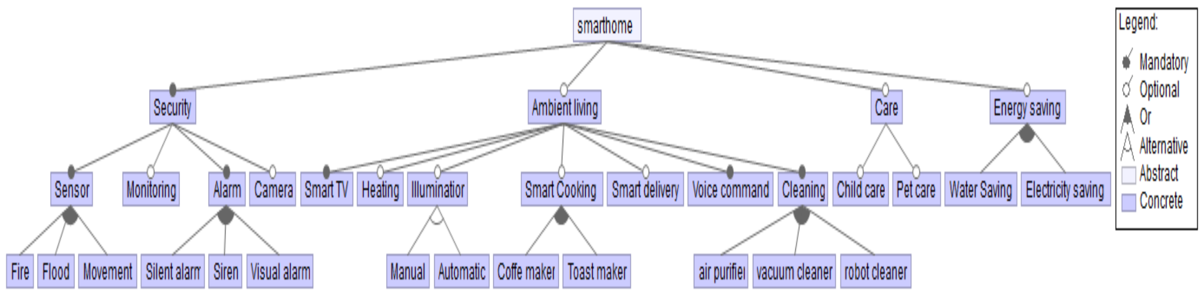

Figure 1: Smart home SO-DSPL.

provide configuration options to adjust a software system at runtime to deal with changes in the users' context (Capilla et al., 2014). Service oriented dynamic software product lines (SO-DSPL) represent a class of DSPLs that are built on services and service-oriented architectures (SOAs) (Capilla et al., 2014).

3 RELATED WORKS

Due to the importance and the complexity of SPL configuration process, a large body of literature has been dedicated to facilitate configuration process by recommending features. Several approaches are based on feature models. In this scenario, some studies (Martinez et al., 2015a) aim to predict the utility of an entire set of features that compose a product, others (Bagheri et al., 2010a),(Bagheri et al., 2010b),(Czarnecki et al., 2012) aim to predict the utility of each feature for the user. Mazo et al. (Mazo et al., 2014) present a collection of recommendation heuristics to prioritize choices and recommend a collection of features to be configured with the objective of reducing the number of configuration. However, configuration process still difficult while the authors do not propose any mechanism to help users and contextual information are not considered by the proposed recommendation heuristics.

Bagheri et al. (Bagheri et al., 2010a) propose a feature ranking approach based on soft and hard constraints in order to satisfy the soft stakeholder constraints. Then, the proposed decision maker select the recommended feature interactively until the full configuration of the feature model.

In (Pereira et al., 2016), Pereira et al. propose a recommendation approach that provides the decision maker with a smaller set of relevant features, such as a top-10 classification. The authors adapt in their work six state-of-the-art recommender algorithms to the product line configuration, which are neighbourhood-based collaborative filtering, collaborative filtering-significance weighting, collaborative filtering-shrinkage, collaborative filtering-Hoeffding, average similarity, and matrix factorization. Martinez et al. (Martinez et al., 2015b) propose an approach that predict configurations' relevance. The authors approach uses tailored data mining interpolation techniques to predict configuration likability based on people's vote feedback for a dataset of configurations. Their approach is developed in two phases. First, a dataset of configurations is created using a genetic algorithm. Second, based on people's vote feedback for the dataset of configurations, tailored data mining interpolation techniques are used to predict configuration likability. Thus, a ranking is created among all possible configurations.

Few works are interested in product configuration based on user requirements. Shamim Ripon et al. (Ripon et al., 2020)propose an approach that extracts automatically requirements, derive configuration and verify product configuration. The proposed approach is divided into two phases: requirement extraction and product configuration verification. In requirement extraction phase, requirements are collected from the user as a textual description, then features are extracted, and a technical product configuration is derived. In the second phase, the derived configuration is checked for validation before sending a feedback to user. In (Maalaoui et al., 2022), the authors proposed an approach to better understand the user requirements, to predict other information about the requirements and to derive an appropriate service (software application as a combination of several services) in the SO-DSPL application engineering phase. In the first step, the core requirements are extracted from user requirements, then features and knowledge are extracted to populate in an ontology which infer relevant knowledge and execute predefined rules to derive or adapt a product with the consideration of contextual information.

Although there are many interesting studies that aim to make the configuration process easier by recommending features and products, the process is still far from trivial. Even features recommendation and

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

ranking algorithms are proposed, the user remains very involved in the configuration process while FM still complicated. However, contextual information is not taken into account during the configuration process. Also, the satisfaction of the recommended features with respect to the DSPL context is not yet verified, while a combination of features may be valid with respect to the constraints of the DSPL but invalid with respect to the DSPL context.

To address this issue, we propose a requirement-based recommender system to suggest to the user a product configuration (set of features) that satisfies its requirements. In contrast to the current literature, our approach predicts features that are not mentioned in the given requirements. The proposed recommendation approach takes into consideration the user context and the derived product configuration is verified with respect to the DSPL context. As well, our proposed approach benefits from the products configurations space, their associated requirements and the contextual information of their users and uses this data to learn intelligently by the use of deep leaning models.

4 CONTEXTUAL PRODUCT RECOMMENDATION FRAMEWORK

According to the limitations presented in related works (see Section 3), recommending products in DSPL framework is still challenging. To ease this process, we propose a product recommendation framework that aims to attend these main objectives:

1. To recommend a set of features which compose a product to an active user
2. To consider contextual information in the recommendation process.
3. To recommend a product adaptation in the case of requirements or context changes.

As shown in the Figure 3, our recommendation framework takes as input a set of requirements given by the user, which are associated to the desired product, and a set of contextual information related to the active user. Given a product requirements and user context, our recommender system is responsible to predict a set of features that satisfy users' requirements, its profile, context and its future context changes with the respect to the DSPL properties in one hand, and predict an adaptation of the product in the case of context change. To recommend relevant products and ensure the integration of an intelligent learning process, our proposed recommendation framework exploits the advantages of machine learning algorithms and the proposed ontology in (Maalaoui et al., 2021). Thus, we use:

- The NLP approach presented in (Maalaoui et al., 2022) to extract relevant information from the given product requirements. The relevant information extracted from the product requirements are named core requirements and this step is called "Core requirement Recognition".
- A neural network model to predict relevant feature based on the extracted core requirements with the attention of contextual data. The proposed model supports the dynamic change of requirements and context by recommending the adapted product.

Our proposed recommendation process starts by extracting core requirements from product requirements in order to keep only relevant information. Then, based on the user context and its given core product requirements, the recommended features are predicted based on a deep learning model which is trained on previous products requirements, their associated products, and the contextual information of their users. Finally, the combination of the prediction features is verified based on the proposed DSPL sub-ontology presented in (Maalaoui et al., 2021).

4.1 Core Requirement Recognition

The core requirement Recognition phase is the data pre-preparation step. In this step, the objective is to remove non-essential and noisy data from the given user requirements, by the interpretation of the user requirements through his/her textual request and its transformation to a formal representation (requirements structure defined as "Core Requirement") that will be later used by the recommendation process. In this step we use the method presented in (Maalaoui et al., 2022), where the input is a textual product requirements given by a user, and the output is the product core requirements built in accordance with the core requirement structure.

A core requirements has the following structure:
<feature>+ <obligationdegree>? <goal>+ <item >* <condition>*
Where :
– * : *denotes a zero repetition to an infinite number of times repetitions.*
– + : *denotes repetition once or more number of times.*
– ? :*denotes a repetition zero or one time.*
– — :*denotes a disjunction (it signifies OR).*
– ! : *denotes a negation*

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

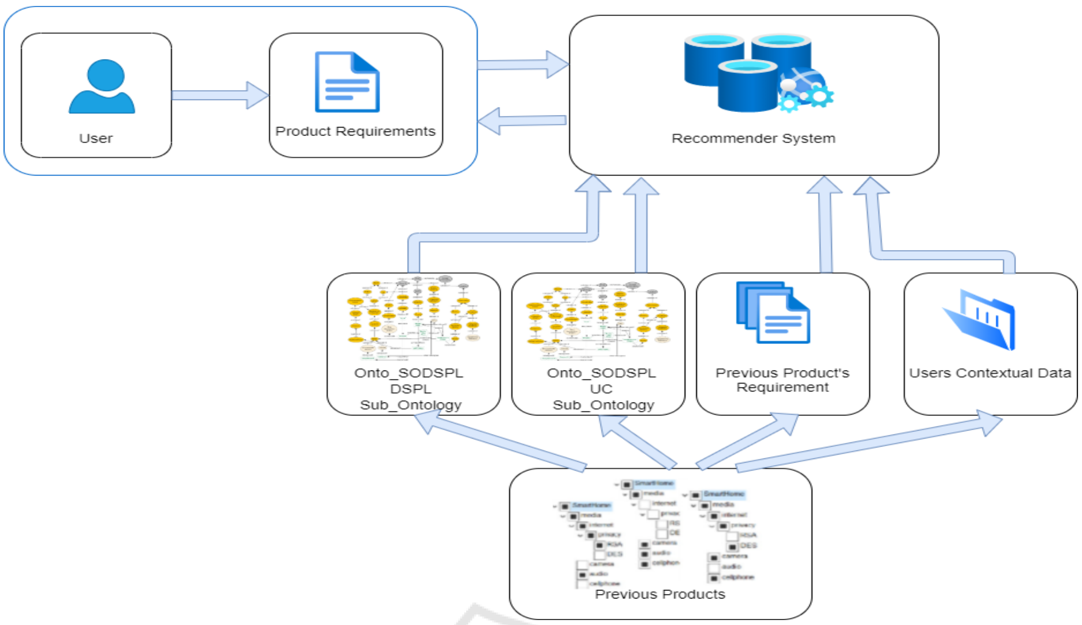
User
Product Requirements
Recommender System
Onto_SODSPL DSPL Sub_Ontology
Onto_SODSPL UC Sub_Ontology
Previous Product's Requirement
Users Contextual Data
Previous Products

Figure 2: Contextual product recommendation framework overview.

As an example, the input is defined by this product requirement:
My sensor should detect movement. if I get up late night, lights in my house should turn on to enhance convenience and safety".
As a result, this product core requirements are generated:
Sensor should detect movement. Lights should turn on if I get up nigh. The used method is based on a set of linguistic rules and the support of uncertainty. These rules facilitate the building of core requirement structure which is then loaded as an instance of an ontology that infer new relevant knowledge.

4.2 Product Recommendation Based on Deep Learning Architecture

Understanding users' requirements is very crucial for the success of software development process, especially in dynamic software product line engineering. As an example, in the DSPL framework, the success of product derivation process is based on the good matching between the expressed user requirements and the features that satisfy them. However, matching users requirements that are expressed in a natural language and its corresponding features with regards to additional information (i.e. the user context) is a challenge. In this section, we present the product derivation approach, which is responsible for recommending a set of features that constitutes a valid product with the goal of the satisfaction of the expressed users' core requirements and its contextual information.

Our proposed recommendation process exploits previous configurations (i.e. selected features), their correspondence with the expressed requirements and their contextual information in order to predict hidden features that the user has not the attention to note them in his requirements. To achieve our objective, we propose a framework based on the combination of deep neural networks, where convolutional and recurrent neural networks are combined together to predict the corresponding product's features.

The choice of this combination is based on the success of Deep neural networks text learning and processing and in recommender system (Katarya and Arora, 2020). Recurrent neural networks (RNN) and short-term memory (LSTM) have been widely applied to text, they have good performance for learning and processing text representation (Sengar et al., 2021). Convolutional Neural Networks (CNN) has also been applied to text classification problems. Especially when integrating with the recurrent model, recurrent-CNN can achieve better performance. As examples, C-LSTM utilizes CNN to extract a sequence of higher-level phrase representations and are fed into a long short-term memory recurrent neural network (LSTM) to obtain the sentence representation. The 2-D CNN can extract not only local features between adjacent words but also the small regions (word stem) inside of words than 1-D CNN. Bi-LSTM can extract sequential features from both past and future.

Thus, in our proposed approach, the stacked 2-D CNN and Bi-LSTM are combined together to recommend product's features. This combination can achieve the best accuracy in our problem, because they can extract not only local features between adjacent words through convolution layers but also global features (long-term dependencies) of sentences by recurrent layers. As well, our proposed framework adopts Roberta (Bidirectional Encoder Representations from Transformers)(Liu et al., 2019) to embed both products requirements and user contextual information with a context-dependent word embedding method. Roberta is a method of pre-training language representations, which can generate different word embedding for a word based on its context(the other words in the sentence).

As consequence, our proposed model can not only automatically abstract the features from core requirements, but also the features from contextual information such as user roles, user interest, user preference and other contextual information chosen by the product line expert such as user region (to deduce the weather). In our approach, we used a smart home DSPL (FM in Figure1 as a running example, based on an analysis of data that influence the recommended result, we have chosen the user roles (worker, home occupant, student, pets owner..) and the user region as contextual information.

4.2.1 Product Recommendation Approach

To recommend products based on user's core requirements with context awareness, we propose an approach using deep neural networks. It mainly consists of three components, including Features Extraction of the Core requirements and the contextual data, feature merge and product's features generation. Figure 4 shows the overall framework of product recommendation. In the first stage, product's core requirements and contextual information are embedded using Roberta model. In the second stage, the stacked 2-D CNN and Bi-LSTM are then used to extract core requirements Features for the embedded ones. After getting the vectors embedding of the user roles and its region from Roberta output, we use a fully connected neural network for feature extraction. After retrieving the feature vectors of product's core requirements h1, user role h2 and user region h3, we merge them into a unified feature. Finally, the task layers do the final features recommendation based a fully connected feed-forward neural network, which inputs the high-level representation from the feature extraction layers (the merged features) and outputs a set of probabilities assigned to each feature of the product configuration that determines whether it is selected or not.

4.2.2 Core Requirements Embedding and Feature Extraction

In order to embed core requirements, Roberta Model is applied. The embedding layer takes a product's core requirements extracted from a product's requirements given by a user as an input, where each word is transformed into a n-dimensional vector according to the contexts of words. This matrix keeps the order and representation of each word, further feature extraction can be done based on this matrix. The embedding layer of product's core requirements outputs a nLen (nLen is the max length of inputs) by n description matrix e for the next layer. We assume that x1 is a set of product's core requirements. The embedding of product's core requirements froberta can be defined as :

$$e = froberta(x1) \tag{1}$$

To extract feature of a product's core requirements, the stacked 2-D CNN and Bi-LSTM are adopted, with f2-D CNN is a deep convolutional model, which uses the trained filters to extract local features between adjacent words and f-Bi-lstm is a deep sequence model, which extract sequential features through time-Therefore, the output feature h1 of product's core requirements is:

$$h1 = fBi - lstm * f2 - DCNN(e) \tag{2}$$

To prevent the model from over-fitting on the training, dropout layer is used. Furthermore, for faster convergence, an activation function **ReLU** is used to improve the model learning efficiency.

4.2.3 Contextual Information Embedding and Feature Extraction

In this step, contextual information chosen by the expert are embedding and its features are extracted. In our running example (smart home DSPL), user roles and user region are chosen as contextual information since they influence the choice of the recommended features and consequently the user's satisfaction. Firstly, the user roles are embedded using Roberta Model. Since the user role usually contains a few words, the further complex feature extraction is not necessary. After getting the embedding of user role from Roberta output, we use a fully connected neural network *ffull* with activation function tanh for feature extraction. We assume that x2 is the user roles associated with a given core requirements. The embedding and feature extraction of a user roles can be defined as:

$$h2 = ffull * froberta(x2) \tag{3}$$

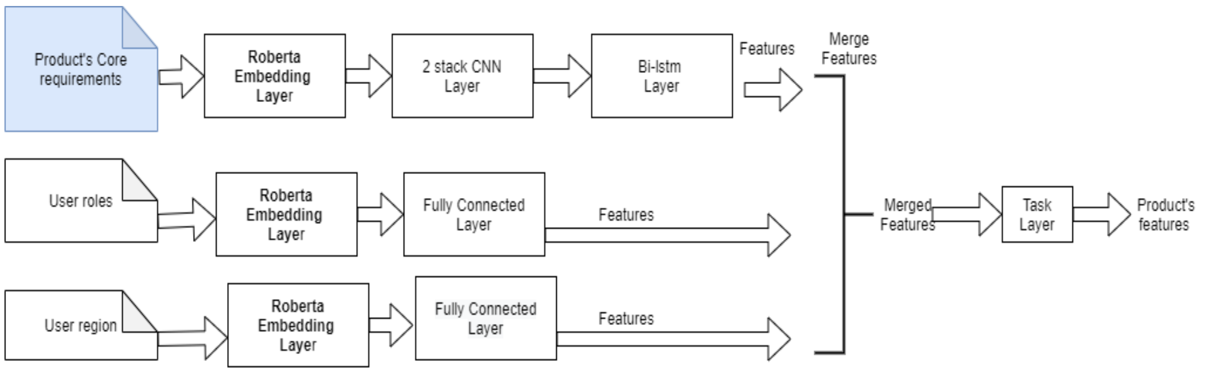

Figure 3: Product recommendation approach.

The same process is applied to the user region. In the first step, Roberta Model is used to embed user region. After that, a fully connected neural network ffull with activation function tanh for is applied to the Roberta output for feature extraction. We assume that x3 is the user region associated with a given core requirements. The embedding and feature extraction of a user region can be defined as:

$$h3 = ffull * froberta(x3) \tag{4}$$

4.2.4 Feature Merge and Features Recommendation

After retrieving the feature vectors of product's core requirements h1, user roles h2 and user region h3, we merge them into a unified feature while the lengths of those features are the same, we adopt simple vector addition for feature merging. The output of feature merging is:

$$h = h1 + h2 + h3 \tag{5}$$

After feature merging, the output result is used by the task layers to do the final product's features recommendation. It contains a fully connected feed-forward neural network ffc, which inputs high-level representation h from feature extraction layers and outputs a recommended list of product's features l:

$$l = ffc(h) \tag{6}$$

ffc contains three fully connected layers with activation function σ1,σ2 and σ3. Each layer computes ai+1 by the weight Wi, bias bi, and the output ai from the previous layer:

$$ai + 1 = \sigma(Wi\_ai + bi) \tag{7}$$

where σ1 and σ2 are the tanh function and σ3 is the sigmoid function that computes the probability of each feature.

In short, our proposed model takes product's core requirements x1, user roles x2 and user region x3 as input, and outputs the vector l that denotes the probability of the selection of each feature of the DSPL to be recommended:

$$l = ffc * (fBi - lstm * f2 - DCNN * froberta(x1) + ffull * froberta(x2) + ffull * froberta(x3)) \tag{8}$$

4.2.5 Hyper-Parameters

The hyper-parameters of our proposed model contains the network configuration and training setting.

1. Network hyper-parameters: We choose the pre-trained ROBERTA model (Roberta-base) from Transformers (rob,), which transforms each word of the input into a 768-dimension vector.

   Our proposed model contains two convolution layers and one bidirectional LSTM layer. The first convolution layer has 64 filters with kernel size 9 by 9, the second convolution layer has 32 filter with kernel size 3 by 3. The hidden state of LSTM is a 1024-dimension vector. The task layer contains 34 nodes with an activation function sigmoid to compute the probabilities of each DSPL feature. To avoid over-fitting, we add a dropout layer between every two layers of our proposed model.

2. Training hyper-parameters: our proposed recommendation model adopts the binary cross-entropy as the loss function. The Adam optimization algorithm is used for training with the learning rate 5e-5, beta1 0.9, beta2 0.999, epsilon=1e-6, and learning decay 0.01.
   Xavier normal initializer is used to initialize the kernel parameters.The total epoch number is 40 with batch size 32. The hidden state of Bi-LSTM and all bias are initialized to zero.

4.3 Recommended Product Consistency Checking

In order to ensure the validity of the recommended product, we check its consistency with respect to SO-DSPL constraints, we use SWRL rules presented in (Maalaoui et al., 2021). In the first step, the recommended product populate the ontology conceptualized for SO-DSPL framework (Maalaoui et al., 2021), then swrl rules are executed. If the product is valid (no violation constraint is detected), it will be taken as the final recommended product, else if the recommended product is not valid (violation constraint is detected), the appropriate swrl rules are triggered to correct the product by activating the valid features and removing the invalid ones. We present in Table1 an extract of the proposed SWRL rules.

5 EVALUATION

In this section, we introduce metrics, and the evaluation results of our proposed approach for product's features recommendation, which includes the comparison results with other machine learning methods. To prepare dataset, we have extended the smart home requirements data set (dat,)on a larger volume of data, we have augmented the existed dataset using data augmentation algorithms (Murukannaiah et al., 2016), which consists in altering an existing data to create a new one. The objective is to augment the dataset by generating new product requirements with the same meaning as the existing requirements but written in another form. Thus, we have used "nlpaug" libraries using the Substitution by contextual word embeddings RoBERTA technique. This data collection consists of textual product requirements. We extend the data with user context information based context elements presented in the "User context sub-ontology" and the "DSPL sub-ontology" (Maalaoui et al., 2021). Indeed, we associated to each product requirements the contextual information of its user that are basically: user interests, user roles, user skills, user preferences and personal information such as its region.

5.1 Competitive Methods

In order to show the feasibility and effectiveness of our approach, we carried out experiments and compared with different ML approaches and using various embedding methods.

We compare 6 combination of machine learning methods from conventional machine learning methods to deep learning models for product's features recommendation on the product's core requirements dataset, which include CNN , LSTM, BI-LSTM and using three embedding methods that include Bert, Roberta and FASTtext. All the conventional models are implemented by scikit-learn library4 and all the deep learning models are implemented by TensorFlow. All the models are trained and tested on Google Colab (col,).

5.2 Evaluation Metrics

In order to measure the accuracy of our proposed product's features recommendation approach among different approaches, Recall and MAP (Mean Average Precision) are used as the evaluation metrics. Each evaluation indicator is described as below.
(1) Recall. It refers to the ratio of the correct number of the recommended product's features in the Product's recommendation list to the total number of the relevant product's features which is defined as:

Recall =
$$\frac{|\{pfeatures\}| \cap \{recomendedfeature\}}{|\{pfeatures\}|}$$

Where **pfeatures** is the real set of features of a product itself, and recomendedfeature is a set of predicted features recommended by a method. For example, given a bloc of core requirements crs associated to a product p and its corresponding features are f1; f2; f3, if the recommended features are f1; f3; f5, then the recall of features recommendation is:

$\frac{|\{f1,f3\}|}{|\{f1,f2,f3\}|}$=0,667

Recall@n refers to the rate when the total number of features of a product itself $|\{pfeatures\}|$= n. That is, the first n features are used to calculate the recall rate. Here, recall measures the degree of the recommendation results that cover all of the Product features.
(2) MAP. It is the expected average of the precision that is the ratio of the correct number of predicted features in the recommendation list, which is defined as:

$$MAP@n = \frac{1}{U}\sum_{u=1}^{U}\sum_{K=1}^{n} P(K) * Rel(K) \qquad (9)$$

where where P(k) is the precision value at the kth recommendation, rel(k), is just an indicator that says whether that kth recommendation was relevant (rel(k)=1) or not (rel(k)=0) and U is the number of

Table 1: Extract of the SO-DSPL SWRL rules.

ID	SWRL Rule	Description
R1	dspl:configuration(?cf)∧ composed-of(?cf,f1)∧ recommended-with(?f1,?f2) -> composed-of(?cf,?f2)	if a feature is selected in a configuration and it recommends another feature (with the semantic relation ”recommended-with” then the recommended feature is with be selected in the running configuration.
R2	swrlx:makeOWLThing(?S, ?y)∧ dspl:Feature(?y)) -> implements(?y, ?S))∧ ws:Service(?S)	For each dspl feature an individual service is created and related by the relationship “implements” to execute its functionalities.
R3	swrlx:makeOWLThing(?f, ?c)∧ dspl:Feature(?F) -> composed-of(?f, ?c)∧ Configuration(?c)	each configuration must be composed by more than one feature.
R4	uc:CoreRequirement(?CR1)∧ uc:CoreRequirement(?CR2) ∧ uc:ProductRequirement(?PR) ∧ uc:composed-of(?PR,?CR1) ∧ uc:composed-of(?PR,?CR2) ∧ has-priority(?CR1, Desirable) ∧ has-priority(?CR2, Essential) ∧ dspl:Feature(?F1) ∧ dspl:Feature(?F2) ∧ dspl:alternative-with(?F2,?F1) ∧ dspl:satisfy(?F1,?CR1) ∧ dspl:satisfy(?F2,?CR2) ∧ dspl:Configuration(?CF) ∧ satisfy(?CF,?PR) ∧ dspl:composed-of(?CF,?F1) -> dspl:composed-of(?CF,?F2)∧ dspl:selected-for(?CF,?F2) ∧ dspl:eliminated-for(?CF,?F1)	The features associated to a product’s core requirements and their are related with the relationship ”alternative-with” then an adaptation is triggered to the running configuration by eliminating the optional feature and selecting the essential one.
R5	uc:CoreRequirement(?CR1)∧ uc:CoreRequirement(?CR2) ∧ uc:ProductRequirement(?PR) ∧ uc:composed-of(?PR,?CR1) ∧ uc:composed-of(?PR,?CR2) ∧ has-priority(?CR1, Essential) ∧ has-priority(?CR2, Essential) ∧ dspl:Feature(?F1) ∧ dspl:Feature(?F2) ∧ dspl:alternative-with(?F2,?F1) ∧ dspl:satisfy(?F1,?CR1) ∧ dspl:satisfy(?F2,?CR2) ∧ dspl:Configuration(?CF) ∧ satisfy(?CF,?PR) ∧ dspl:composed-of(?CF,?F1) -> uc:alternative-conflict(?CR1,?CR2)	The features associated to a product’s core requirements and their are related with the relationship ”alternative-with” and the two core requirements are essential then a conflict of alternative constraint violation is detected.

relevant recommendation, with P(K) is defined as:

P(K)= $\frac{|\{pfeatures\}|\cap\{recommendedfeature(k)\}}{|\{K\}|}$—

and precision P is defined as:

P = $\frac{|\{pfeatures\}|\cap\{recomendedfeature\}}{|\{recomendedfeature\}|}$

5.2.1 Experimental Results and Analyses

To test the performance of our proposed approach, the extensive experiments on product recommendation are conducted among the mentioned competitive approaches. Table 2 shows the experimental results on the different evaluation metrics.

Table 2: Comparison result of machine learning methods.

ML Methods	MAP@10	RECALL@10
Roberta+CNN	0,57	0,41
Roberta +LSTM	0,72	0,587
Roberta +BI-LSTM	0,78	0,594
Bert+cnn+lstm	0.857	0,65
FASTTEXT+cnn+lstm	0.753	0,61
Roberta+cnn+lstm	0,88	0,692
Roberta+2-D cnn+BI-lstm	0,91	0,78

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

It can be seen from the experimental results that our proposed product's features recommendation approach is superior to the existing ones among the evaluation metrics. In the experiments, the CNN has the lowest performance on MAP@10 and Recall@10, that means only considering local feature between adjacent words is not working for SPL features recommendation problem. For sequence models, they can abstract learn global features such as long-term dependencies from the past time steps, which makes LSTM reach 72,34% MAP@10 and 58,79% Recall@10. When stacking sequence model with 1-D CNN for learning the local features, LSTM can get higher MAP@10 and Recall@10 88,21% and 69,32%. BI-LSTM can learn the long-term dependencies from the past as well as the future information. That makes BI-LSTM reach 78.70% MAP@10 and 59.43% Recall@10. Finally, our proposed model uses 2-D CNNs with BI-LSTM, it can learn both local and global features as well as the features in small regions inside of words. Moreover, it uses both information from core requirements, user roles and user region with context-dependent word embedding. Applying three context-dependent word embedding methods with our proposed model, it has a lower performance with FASTTEXT embedding than with BERT embedding. While, using Roberta embedding, our proposed model has the highest performance.

In order to further test the parameter influence on the product's features recommendation, a set of experiments are carried out among four competitive approaches. The experimental results of parameter tuning are illustrated in Figure4 and Figure5.

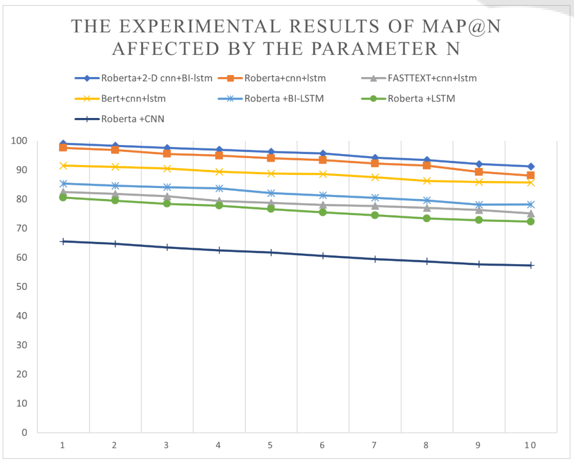

Figure 4: The experimental results of MAP@n affected by the parameter n.

Along with the changes of parameter for the number of recommended features, Figs. 5 and 6 illustrate the experiments results on MAP@n and Recall@n among four competitive approaches. In the experiments, the parameter of MAP@n and Recall@n ranges from 1 to 10 and the results are calculated with different values.

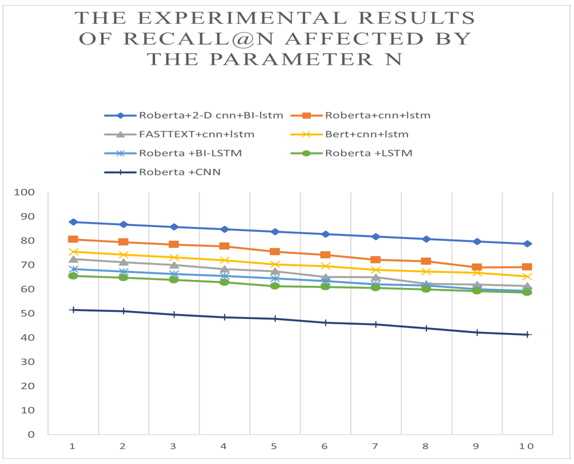

Figure 5: The experimental results of Recall@n affected by the parameter n.

Compared to the existing approaches, Fig. 4 shows that the experimental results of our proposed approach has better MAP@n along with the changes of n. Specifically, as n of MAP@n changes from 1 to 10, the experimental results of each approach have a certain decrease, and reach the best features recommendation at MAP@1. Therefore, as the number of features to be evaluated increases, the predictive power of each recommendation approaches declines. However, our self-developed approach has the lowest decline compared with the other approaches.

6 CONCLUSION AND FUTURE WORKS

In this paper, we targeted an open research question in the SO-DSPL configuration domain: How to predict a suitable set of features based on explicit information from users? We answer this question by providing an advanced a contextual product recommender system. Our system helps user in the product configuration process by suggesting relevant features according to its described textual requirements. Our experimental results show that the proposed approach is very useful as it provides feature predictions that are in accordance with the user context and requirements. Since there are no other publicly available datasets with real user requirements and associated configurations, we could not test our work on further datasets. Moreover, we will extend this work to take into account non-functional properties, adapt our approach to be more efficient with a large number of features and training our proposed deep learning model on more dynamic adaptation cases that can be triggered at runtime.

REFERENCES

Google colab. https://colab.research.google.com/. Accessed: 2022-05-15.

nlpaug. https://github.com/makcedward/nlpaug. Accessed: 2022-05-15.

Roberta. https://huggingface.co/roberta-base. Accessed: 2022-05-15.

Bagheri, E., Asadi, M., Gasevic, D., and Soltani, S. (2010a). Stratified analytic hierarchy process: Prioritization and selection of software features. In Bosch, J. and Lee, J., editors, *Software Product Lines: Going Beyond*, pages 300–315, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bagheri, E., Di Noia, T., Ragone, A., and Gasevic, D. (2010b). Configuring software product line feature models based on stakeholders' soft and hard requirements. In *Proceedings of the 14th International Conference on Software Product Lines: Going Beyond*, SPLC'10, page 16–31, Berlin, Heidelberg. Springer-Verlag.

Bashari, M., Bagheri, E., and W.Du (2017). Dynamic software product line engineering: A reference framework. *International Journal of Software Engineering and Knowledge Engineering*, pages 191–234.

Capilla, R., Bosch, J., Trinidad, P., Ruiz-Cortes, A., and Hinchey, M. (2014). Overview of dynamic software product line architectures and techniques:observations from research and industry. *The Journal of Systems and Software*, pages 3–23.

Czarnecki, K., Grünbacher, P., Schmid, R. R. K., and Wasowski, A. (2012). Cool features and tough decisions:a comparison of variability modeling approaches. In *Proceedings of the 6th International Workshop onVariability Modelling of Software-Intensive Systems (VaMoS'12)*, pages 173–182. ACM.

Jonathan, Y., James, T., and Audrey, T. (2005). Evaluating ontology criteria for requirements in a geographic travel domain. In Meersman, Robert, Tari, and Zahir, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pages 1517–1534, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kang, K., Cohen, S., Hess, J., Novak, W., and Peterson, A. (1990). Feature-oriented domain analysis (foda) feasibility study. *Rep. CMU/SEI-*, 90.

Katarya, R. and Arora, Y. (2020). Capsmf: a novel product recommender system using deep learning based text analysis model. *Multimedia Tools and Applications*, pages 1–22.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Maalaoui, N., Beltaifa, R., and Labed Jilani, L. (2022). Deriving service-oriented dynamic product lines knowledge from informal user-requirements: Ai based approach. pages 58–68. ICSEA 2022.

Maalaoui, N., Beltaifa, R., Labed Jilani, L., and Mazo, R. (2021). An ontology for service-oriented dynamic software product lines knowledge management. In *ENASE*.

Martinez, J., Rossi, G., Ziadi, T., Bissyandé, T. F. D. A., Klein, J., and Le Traon, Y. (2015a). Estimating and predicting average likability on computer-generated artwork variants. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, page 1431–1432, New York, NY, USA. Association for Computing Machinery.

Martinez, J., Rossi, G., Ziadi, T., Bissyandé, T. F. D. A., Klein, J., and Le Traon, Y. (2015b). Estimating and predicting average likability on computer-generated artwork variants. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, page 1431–1432, New York, NY, USA. Association for Computing Machinery.

Mazo, R., Dumitrescu, C., Salinesi, C., and Diaz, D. (2014). Recommendation Heuristics for Improving Product Line Configuration Processes. In M, R., W., M., R., W., and T., Z., editors, *Recommendation Systems in Software Engineering*, page 100. Springer.

Murukannaiah, P. K., Ajmeri, N., and Singh, M. P. (2016). Acquiring creative requirements from the crowd: Understanding the influences of personality and creative potential in crowd re. *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 176–185.

Pereira, J. A., Matuszyk, P., Krieter, S., Spiliopoulou, M., and Saake, G. (2016). A feature-based personalized recommender system for product-line configuration. *SIGPLAN Not.*, 52(3):120–131.

Ripon, S., Rasel, F. S., Howlader, R. K., and Islam, M. (2020). *Automated Requirements Extraction and Product Configuration Verification for Software Product Line*, pages 27–51. Springer Singapore, Singapore.

Sengar, N., Singh, A., and Yadav, V. (2021). Classification of documents using bidirectional long short-term memory recurrent neural network. In Reddy, V. S., Prasad, V. K., Wang, J., and Reddy, K. T. V., editors, *Soft Computing and Signal Processing*, pages 149–156, Singapore. Springer Singapore.