CIRP Annals - Manufacturing Technology xxx (2018) xxx–xxx

Contents lists available at ScienceDirect

CIRP Annals - Manufacturing Technology

journal homepage: http://ees.elsevier.com/cirp/default.asp

Mapping customer needs to design parameters in the front end of product design by applying deep learning

Yue Wang [a], Daniel Y. Mo [a], Mitchell M. Tseng (1)[b,*]

[a] Hang Seng Management College, Hong Kong
[b] Feng Chia University, Taiwan

A R T I C L E I N F O

Keywords:
Design
Customization
Deep learning

A B S T R A C T

The key to successful product design is better understanding of customer needs (CNs), and efficiently translating CNs into design parameters (DPs). With the recent trend toward the diversification of CNs, the rapid introduction of new products, and shortened lead times, there is a growing need to speed up the mapping from CNs to DPs. By leveraging on product review data extracted e-commerce websites, this paper proposes a deep learning-based approach to improve the effectiveness and efficiency of mapping CNs to DPs. The results show that the proposed approach can meet customer needs with high efficiency.

1. Introduction

Due to shortening product life cycles and the rapid introduction of new products, companies today face an ever-growing need for fast, effective product design and development [1]. This challenge is accentuated by the growing trend toward responsive retail systems that are developed to better satisfy customer needs (CNs). Effective design is increasingly necessary for development of new methods that can keep pace with the demand for shorter product development lead times. It is widely acknowledged that the front end of the design stage is critical to the success of product development [2]. Perhaps the most essential factor is the process of eliciting and understanding CNs and translating them into tangible product specifications for each customized order [2]. Thus, a better process of mapping CNs onto product design specifications can greatly contribute to shortening product development times and improving design quality.

In the conventional design process, the potential users of products are invited to express their needs, and the designers seek to transfer these needs into tangible product specifications [3]. Extensive research has been conducted to characterize, understand, and elicit CNs for product design. The approaches devised for this purpose include online product configurator design, conjoint analysis, voice of customer analysis, and Kansei/affective engineering [4–6]. Most of these methods depend on the customers' ability and willingness to explicitly express their preferences and needs in well-defined formats. However, the difficulties involved in articulating CNs have also been well documented [7]. Customers may not possess the necessary expertise concerning products with which they are unfamiliar. This difficulty in expressing their intents and needs in clear, specific language has been reported. Customers may, for example, use vague and ambiguous language, such as saying "A PC with a very good graphics card to play video game," instead giving a product design parameter (DP), such as "a PC with Nvidia GeForce GTX 1060 graphical card." This kind of ambiguity increases the level of frustration and confusion that customers often experience, which can make them reluctant to accept product offerings [8]. Such miscommunication may easily lead to design defects. To address these difficulties, an effective approach is needed to transform needs to product DPs.

Today, a large amount of product review data is available on numerous e-commerce websites. However, CNs data are not available on a large scale. Research has acknowledged significant similarities among the product review text and actual CNs data [9]. Thus, we treat product review keywords as CNs throughout the paper. There is the potential to develop an approach to effectively translate the commonly ill-defined CNs into sets of valid product DPs via data-driven approaches.

However, challenges remain in attempting this mapping task. First, the amount of review data to be assessed is often enormous. It is difficult to process such vast amounts of data and leverage that data to perform manual mapping. Second, product review data are not always relevant to the specific questions being asked. The review data, which are usually in the format of text input by users, commonly contain a great deal of irrelevant information. It is critical to filter out this unnecessary information and extract what is most relevant to CNs regarding specific products. Finally, the review texts are almost always unstructured. Generally, no predefined template or model exists for users to follow when inputting review text.

To meet these various challenges, we draw on deep learning approaches to devise a mapping network that can overcome the barrier between the CN and DP domains and thus connect e-commerce with the front end of design with speed and clarity. Here, we assume that the DP domain is fixed, as is the common

* Corresponding author.
E-mail address: mmtseng@mail.fcu.edu.tw (M. Tseng).

https://doi.org/10.1016/j.cirp.2018.04.018
0007-8506/© 2018 Published by Elsevier Ltd on behalf of CIRP.

practice for online product searches and online product customization, such as the configure-to-order process. Deep learning has attracted considerable attention in recent years due to its superior performance in various tasks such as enabling computer vision, processing natural language, and enhancing traditional engineering domains [10,11]. Deep learning models are learned on the basis of large sets of training data and complicated deep structures. When the model is learned sufficiently, customers are simply required to express their needs in layman's language, and these needs are automatically mapped to the model's design specifications. This approach helps shield customers from the confusion they often encounter when using the traditional design, and designers can save a great deal of tedious and time-consuming work.

2. Deep learning-based mapping from CNs to DPs

2.1. Deep network structure

Deep learning is a type of machine learning that is designed to accumulate high-level representations of domain information from raw data. The models usually contain multiple layers, and this approach has proven to be effective for identifying abstractions or representations in data [11]. Among the various techniques of deep learning, the long short-term memory (LSTM) technique is widely used to analyze sequential data such as text, speech, and video information [12]. Because product review data are usually in the format of text, our investigation applies LSTM to extract keywords from raw product review text.

An LSTM memory unit contains three essential gates and a cell, as shown in Fig. 1. The input gate is used to determine what type of new information should be stored in the LSTM cell. The output gate determines what kind of information should be outputted. The recurrent part is updated by the current status and then feeds the information into the next iteration. The forget gate decides which information from the previous iteration should be abandoned. The dash lines in the figure indicate time-lag input. The gates and cells are updated as follows:

$$\begin{aligned} z_t &= \sigma(W_z x_t + R_z y_{t-1} + b_z) \\ i_t &= \sigma(W_i x_t + R_i y_{t-1} + p_i \otimes c_{t-1} + b_t) \\ f_t &= \sigma(W_f x_t + R_f y_{t-1} + p_f \otimes c_{t-1} + b_f) \\ c_t &= i_t \otimes z_t + f_t \otimes c_{t-1} \\ o_t &= \sigma(W_o x_t + R_o y_{t-1} + p_o \otimes c_t + b_0) \\ y_t &= o_t \otimes \mathrm{h}(c_t) \end{aligned} \qquad (1)$$

where W_z, W_i, W_f, W_o, R_z, R_i, R_f, R_o are weight matrices for each gate's input, and the recurrent parts b_z, b_i, b_f, b_o are the bias vector. Here, σ is sigmoid function and h is tanh function. Also, $\otimes$ is the point-wise multiplication calculation of two vectors, and p_z, p_i, p_f, p_o are used to denote the peephole connection in an LSTM network. To simplify the notations, the set of functions in Eq. (1) is represented as $y_t = LSTM(x_t, y_{t-1})$, because the output at t is only dependent on the input at time t and the output of the previous time period.

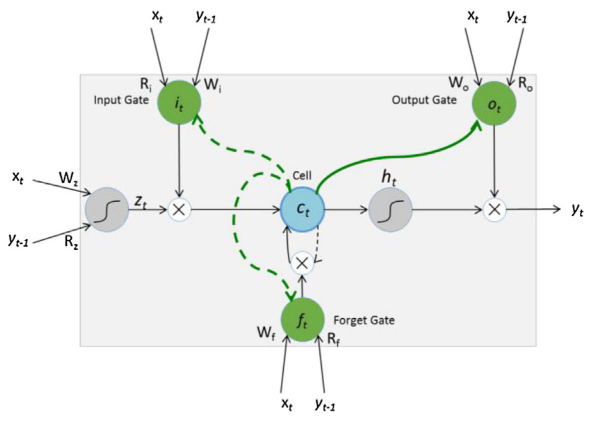

Fig. 1. A long short term memory (LSTM) unit.

An LSTM network leverages previous information to determine current output. For a given words sequence $(x_1, ..., x_n)$ in a review sentence, the corresponding output of the LSTM network is a sequence $(y_1, ..., y_n)$ where $y_t = LSTM\ (x_t, y_{t-1})$. However, the information contained in a text chunk is context dependent. The meaning of one sentence is not only dependent on the previous sentences but also related to the sentences that follow. To fully use such contextual information, a bi-directional LSTM (BLSTM) network is proposed [13]. Unlike an LSTM network, a BLSTM network contains two parallel layers that propagate both forward and backward, thus allowing the network to obtain information in a sequential series of both the past and the projected future. Each forward or backward layer functions similarly to a regular LSTM in Eq. (1). The output of BLSTM y_t is thus a concatenation of the output from the two layers y_t^f, y_t^b, corresponding to the forward and backward direction of the context, i.e., $y_t = [y_t^f, y_t^b]$, where $y_t^f = LSTM(x_t, y_{t-1}^f)$ and $y_t^b = LSTM(x_t, y_{t+1}^b)$.

2.2. Conditional random field

The CNs text sequence $(x_1, ..., x_n)$ transforms to the output sequence $(y_1, ..., y_n)$ where $y_t = [y_t^f, y_t^b]$ using BLSTM. y_t is the high-level representation of the input text and contains the semantic information of the text. In a typical natural language processing task, $(y_1, ..., y_n)$ is usually fed into a conditional random field (CRF) to map the result of BLSTM to the domain of interest [14], and thereby forms a general classifier. This type of BLSTM CRF classifier has been acknowledged to be efficient and effective in natural language processing tasks such as the identification of name entities. In this paper, BLSTM CRF is used as a critical element in the entire mapping framework.

The output of BLSTM $(y_1, ..., y_n)$ serves as the input of a CRF. Let $z = \{z_1, z_2, ..., z_n\}$ be the set labels of the input sequence. For example, if the $(y_1, ..., y_n)$ is the BLSTM output of a paragraph of the review text, then the label for each input item (i.e., sentence) is a binary indicator that quantifies whether the sentence is product-relevant. Let $Z(y)$ be the set of all possible label sequences for y. Then a CRF defines the conditional probability $p(z|y)$ over all of the possible label sequences z, given y:

$$p(z|y) = \frac{\prod_{i=1}^{n} \psi_i(z_{i-1}, z_i, y)}{\sum_{z' \in Z(y)} \prod_{i=1}^{n} \psi_i(z'_{i-1}, z'_i, y)} \text{ where } \psi_i(z_{i-1}, z_i, y) = \exp\left(W_{z_{i-1}, z_i}^T y_i + b_{z_{i-1}, z_i}\right).$$

Also, W_{z_{i-1}, z_i}^T and b_{z_{i-1}, z_i} are the weight and bias vectors of the label pair (z_{i-1}, z_i), as calculated from the data input [14]. Then, for the CRF-based classifier, we need only identify the label sequence z^* with the greatest conditional probability, i.e., $z^* = \underset{z \in Z(y)}{arg\ \max}\, p(z|y)$.

2.3. Framework of mapping from CNs to DPs

The general framework of the entire mapping process can be illustrated by Fig. 2. In the offline learning stage, three cascaded classifiers are learned from training data. Once the mapping from CNs to DPs, which consists of three classifiers, is obtained, it can be applied to map a new customer's CNs to the corresponding DPs in the online application stage. Specifically, there are three steps in the training stage.

1) Filtering raw product review data. Many sentences in product review text chunks are irrelevant to the products being reviewed. Thus, preprocessing is needed to filter out those irrelevant sentences. We manually annotate whether the sentence in the test chunk is relevant. Based on the annotated sentences, we build a BLSTM CRF-based classifier (classifier 1) to