Engineering Applications of Artificial Intelligence 87 (2020) 103322

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Understanding what the users say in chatbots: A case study for the Vietnamese language☆

Oanh Thi Tran [a,*], Tho Chi Luong [b]

[a] VNU International School, Vietnam National University, Hanoi, 144 Xuan Thuy, Cau Giay, Hanoi, Viet Nam
[b] FPT Technology Research Institute, 82 Duy Tan, Cau Giay, Hanoi, Viet Nam

ARTICLE INFO

Keywords:
User requests
Chatbots
Intent detection
Context extraction
Neural networks

ABSTRACT

This paper[1] presents a study on understanding what the users say in chatbot systems: the situation where users input utterances bots would hopefully (1) *detect intents* and (2) *recognize corresponding contexts* implied by utterances. This helps bots better understand what users are saying, and act upon a much wider range of actions. To this end, we propose a framework which models the first task as a classification problem and the second one as a two-layer sequence labeling problem. The framework explores deep neural networks to automatically learn useful features at both character and word levels. We apply this framework to building a chatbot in a Vietnamese e-commerce domain to help retail brands better communicate with their customers. Experimental results on four newly-built datasets demonstrate that deep neural networks could be able to outperform strong conventional machine-learning methods. In detecting intents, we achieve the best F-measure of 82.32%. In extracting contexts, the proposed method yields promising F-measures ranging from 78% to 91% depending on specific types of contexts.

1. Introduction

Chatbots are beginning to take over the world of e-Commerce. Many brands are using them to better communicate with their target audiences, recommend products, and get orders. One of the biggest challenges of building such bots is the ability to understand user utterances in natural languages. Let us take a takeout bot as an example. When users state a task they want to complete via the message '*Ship two cups of coffee to 144 Xuan Thuy Cau Giay right now.*', the bot would hopefully recognize the ordering command '*ship*' and the corresponding contexts which are: (i) the requested item with its Product Information (PI[2]) *'2 cups of coffee'* , (ii) the shipping time *'right now'* , and (iii) the shipping address *'144 Xuan Thuy Cau Giay'*. This information is further decomposed into more detailed elements as shown in Fig. 1. This allows our bot to response and act upon a much wider range of actions. For example, if the product attribute was not mentioned, the bot would provide an appropriate response to clarify it (e.g. *hot, cold*); the bot would also automatically fill missing address fields (e.g. the commune '*Quan Hoa*'), etc. Such systems usually consist of two key steps as follows:

1. **Intent Parser** detects intents implied by user utterances such as *greetings*, *placing orders*, *showing menu*, *asking for promotion, etc.* This is a non-trivial task because of various oral expressions in informal contexts.
2. **Context Extractor** extracts meaningful semantic chunks in texts. This helps bots to identify what they have already known and only seek out the unknown information needed to provide a proper response.

Several work has been done for popular languages like English and Chinese by using Information Retrieval techniques (Ji et al., 2014; Yan et al., 2016), hand-crafted rules (Ali and Habash, 2016), or neural mechanisms (Cui et al., 2017; Yan et al., 2017; Li et al., 2018). However, little work has been done for Vietnamese. Most studies on Vietnamese restricted themselves to detecting intents using conventional methods (Ngo et al., 2016) or extracting contexts using conjunction matching (Tran et al., 2016). To our knowledge, there has no public research about deeply analyzing what users say in chatbots, especially in Vietnamese. Therefore, this research focuses on studying and developing an intelligent module to equip our bot with the ability

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2019.103322.

* Corresponding author.
E-mail addresses: oanhtt@isvnu.vn (O.T. Tran), tholc2@fpt.com.vn (T.C. Luong).

[1] This paper is an improved and extended version of Tran and Luong.

[2] Selling online requires companies to collect clear basic PI that consumers can actually understand. Without PI (e.g product names, prices and categories), the product could not be found and sold online at all.

https://doi.org/10.1016/j.engappai.2019.103322
Received 7 April 2019; Received in revised form 24 September 2019; Accepted 24 October 2019
Available online 11 November 2019
0952-1976/

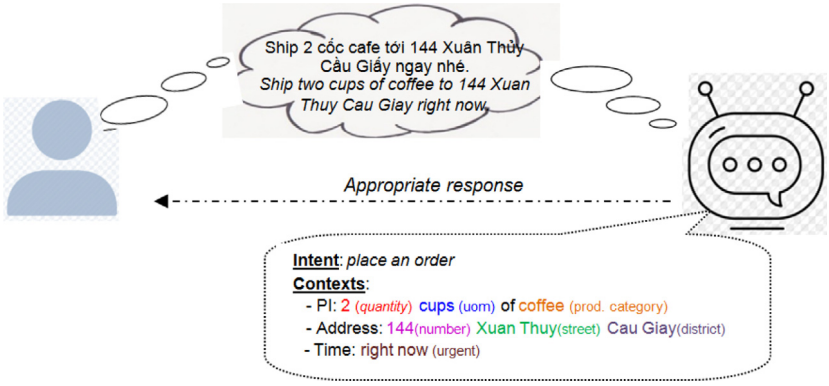

Fig. 1. Analyzing an utterance in terms of detecting the intent and extracting the contexts.

to better understand user utterances. Such utterances are usually characterized by no word boundary, different stylistic, lexical, and syntactic peculiarities, conversational slang words, teen-codes, etc.

To this end, we introduce a framework which consists of two main steps: (1) *intent detection* which is modeled as a classification problem and (2) *context extraction* which is modeled as a two-layer sequence labeling problem. An interesting point is that in building such models instead of heavily designing feature sets, advanced deep learning techniques are explored to automatically learn useful features at both character and word levels. This is motivated from the fact that recently, with the help of word embeddings, neural networks such as RNNs (Wilcox et al., 2018), and CNNs (Zhu et al., 2017) have demonstrated their great performance in many NLP tasks. Our paper makes the following contributions:

- Propose a framework to deeply analyze users' utterances in a Vietnamese e-commerce domain, which includes two key modules: an intent parser and a context extractor.
- Introduce four new corpora[3] to help building the e-commerce bots.
- Show that using automatically learnt features is effective and yields better performance than using hand-crafted ones for the both two tasks.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describes a solution to detect user intents. Section 4 describes a solution to extract useful contexts from user utterances. Section 5 introduces four new corpora in a retail domain and shows some statistics. Experimental setups, experimental results, and error analysis are reported in Section 6. Finally, we conclude the paper and point out some future work in Section 7.

2. Related work

To detect intents, most research trained a conventional classifier using an annotated corpus by investigating different kinds of features (Hu et al., 2009; Mendoza and Zamora, 2009). However, this method requires heavy feature engineering. Another alternative is to automatically learn discriminative features using deep neural methods. For example, Shi et al. (2016) proposed to stack different DLSTM feature mappings together to model multiple level nonlinear feature representations; Kato et al. (2017) applied the recursive autoencoder to the utterance intent classification of a smartphone-based Japanese spoken dialog system; Goo et al. (2018) proposes a slot gate that focuses on learning the relationship between intent and slot attention vectors in order to obtain better semantic frame results by the global optimization.

To extract contexts, most current chatbots use IR techniques (Ji et al., 2014; Yan et al., 2016; Qiu et al., 2017). That is given a question, the approach retrieves the most similar question in predefined FAQs and takes the paired answer as a response. This technique is usually applied to build open-domain chatbots (e.g. serving chit-chat), or answer FAQs in a given domain. For examples, Brixey et al. (2017) built a bot answering a wide variety of sexual health questions on HIV/AIDS. In e-commerce, Cui et al. (2017) selected the best answer from existing data sources (including PI, FAQs, and customer reviews) to support chit-chat, and give comments about a given product. Yan et al. (2017) presented a general solution towards building task-oriented dialog systems for online shopping. To extract PI asked by customers, the system matched the question to basic PI using DSSM model (Huang et al., 2013). Unfortunately, these studies do not support customers to perform ordering online, and the external data resources are intractable in many real-world applications.

For Vietnamese, there was little work about this topic (Ngo et al., 2016; Tran et al., 2016). Ngo et al. (2016) used a MaxEnt classifier to classify applications and a conjunction matching method to identify actions. Designing that matching is cost-consuming, requiring domain experts, and cannot handle with ambiguity. In that work, they focused on processing simple questions which are usually easier to analyze to support users completing some tasks using their mobile phones. Tran et al. (2016) studied recognizing named entities in Vietnamese spoken texts. This work proposed a lightweight machine learning model which incorporates a variety of rich hand-crafted features. Unfortunately, manually designing these features is challenging and requires domain-and-expert knowledge.

Despite the popularity and improved techniques, chatbots still face various challenges in understanding natural languages, especially in ordering situations. Users can express an idea in different forms, and styles of conversing also vary from person to person. Therefore, in this paper, we concentrate on the study of understanding what users say to help bots better understand the informal Vietnamese language. This language poses several challenges as previously mentioned. To this end, we propose a solution which exploits advanced deep neural networks. This solution does not use extra external resources and also does not suffer from drawbacks of rule-based approaches as in previous work.

3. Intent parser

This section formally defines the task of intent detection and describes the proposed solution.

3.1. Problem definition

Intent classification is the task of identifying a user utterance as belonging to one or more categories from a predefined set of user intents, L. This study deals with single-label intent detection. Given a

[3] Please contact the corresponding author.

user utterance in the form of $S = \{w_1, w_2, \ldots, w_n\}$, w_i is the ith word, the goal is to find the intent l' that maximizes:

$$l' = argmax_l p(l|w_1, w_2, \ldots, w_n), l \in L \quad (1)$$

As it has been shown there exist several methods proposed. However, none of these methods have been thoroughly evaluated or proved to be outperformed with others in diversity of contexts. Thus, the question here is which is the most appropriate method to perform intent detection in Vietnamese ordering chatbot systems? With regard to this question, we propose a method which explores deep neural networks towards automatic semantic feature extraction. In this paper, two neural architectures, LSTMs (Hochreiter and Schmidhuber, 1997) and CNNs (LeCun and Bengio, 1998), are exploited.

3.2. A bi-LSTM model for detecting intents

Let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ denote an input sentence consisting of the word representations of n words. At each position t, the RNN outputs an intermediate representation based on a hidden state $\mathbf{h}$:

$$\mathbf{y}_t = \sigma(\mathbf{W}_y \mathbf{h}_t + \mathbf{b}_y),$$

where $\mathbf{W}_y$ and $\mathbf{b}_y$ are parameter matrix and vector which are learned in the training process, σ denote the element-wise Softmax function. The hidden state $\mathbf{h}_t$ is updated using a non-linear activation function on the previous hidden state $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$ as follows:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t).$$

LSTM cells use a few gates, including an input gate $\mathbf{i}_t$, a forget gate $\mathbf{f}_t$, an output gate $\mathbf{o}_t$ and a memory cell $\mathbf{c}_t$ to update the hidden state $\mathbf{h}_t$ as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{V}_i \mathbf{h}_{t-1} + \mathbf{b}_i),$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{V}_f \mathbf{h}_{t-1} + \mathbf{b}_f),$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{V}_o \mathbf{h}_{t-1} + \mathbf{b}_o),$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{V}_c \mathbf{h}_{t-1} + \mathbf{b}_c),$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),$$

where $\odot$ multiplication operator functions, $\mathbf{V}$ is a weight matrix, and $\mathbf{b}$ is vectors to be learned.

To improve model performance, two LSTMS are trained on user utterances. The first on the utterance from left-to-right and the second on a reversed copy of the utterance. The forward and backward outputs should be combined before being passed on to the next layer by concatenation by default. Finally, an activation function is applied on this concatenation vector to obtain the prediction.

3.3. A CNN model for detecting intents

Consider the ith word of a given user utterance, $x_i \in R^k$ represents its k -dimensional word embedding vectors. Let $\mathbf{X}$ be the vector concatenation of the n words in the utterance. In general, let $x_{i:i+j}$ refer to the concatenation of words $x_i, x_{i+1}, \ldots, x_{i+j}$. The convolution layer $\mathbf{w} \in R^{h \times k}$ is employed to learn representations from sliding h-words with h denotes the window size to generate a new feature for the ith word as follows:

$$c_i = \tanh(\langle \mathbf{w}, x_{i:i+h-1} \rangle + b) \quad (2)$$

where $\langle , \rangle$ is the Frobenius inner product, $b \in R$ is a bias factor and tanh is a hyperbolic tangent function. This filter is applied to each possible window of words in the utterance to produce a feature map. The max pooling is then applied to obtain the feature corresponding to filter $\mathbf{w}$:

$$\hat{c} = max(c_1, c_2, \ldots, c_{n-h+1}). \quad (3)$$

The idea is to capture the most important features with the highest value for each feature map. By using multiple filters with different widths, the feature representation $\mathbf{c}$ for the utterance is obtained. The dimension of the feature representation equals to the number of filters $\mathbf{w}$. The final activation layer then receives this feature vector as input and uses it to classify the utterance.

4. Context extractor

This section first states the problem, then proposes a solution to solve the task. We focuses on three typical contexts: PI, dates/time and addresses.

4.1. Problem definition

Given a user utterance in the form of $X = \{x_1, x_2, \ldots, x_n\}$, x_i is the ith word. The model would extract contexts inside it. The contexts we are trying to assign here are product descriptions, shipping dates/time and shipping addresses. We also go a step further to parse its product description into basic PI (e.g category, brand, attribute, pack-size, quantity, etc.); parse its shipping time and addresses into its corresponding individual elements (e.g date, month, hour, week, prefix, etc. for dates/time; street, commune, district, etc. for addresses).

The task is modeled as a two-layer sequence labeling problem. In layer 1, each context is considered as an entity. In layer 2, detailed elements of each context are considered as entities. These layers can be performed independently.

4.2. A proposed model for extracting contexts

Sequence labeling deals the task by making the optimal label for a given element dependent on the choices of nearby elements. For this, we use CRFs (Lafferty et al., 2001) which are widely applied, and yield state-of-the-art results in many NLP problems. Specifically, the conditional probability of a state sequence $S = \langle s_1, s_2, \ldots, s_T \rangle$ given an observation sequence $O = \langle o_1, o_2, \ldots, o_T \rangle$ is calculated as:

$$P(s|o) = \frac{1}{Z} exp(\sum_{t=1}^{T} \sum_{k} \lambda_k \times f_k(s_{t-1}, s_t, o, t)) \quad (4)$$

where $f_k(s_{t-1}, s_t, o, t)$ is a feature function (manually designed or automatically learnt from a neural model) whose weight λ_k is to be learned via training. To make all conditional probabilities sum up to 1, we must calculate the normalization factor Z over all state sequences:

$$Z = \sum_{s} exp(\sum_{t=1}^{T} \sum_{k} \lambda_k \times f_k(s_{t-1}, s_t, o, t)) \quad (5)$$

To build the strong model, CRFs need a good feature set. These features will be automatically learnt via neural models.

4.3. Extracting features automatically using neural models

Without heavily designing features by hands, the proposed methods exploits non-linear neural networks which are LSTMs and CNNs to automatically learn features for CRFs. These models use no language-specific resources beyond a small amount of supervised training data and unlabeled data.

biLSTM

This study adapts the method of Lample et al. (2016) which has the ability of capturing both orthographic evidence and distributional evidence. In order to train the model, we use pre-trained word embeddings from general texts. We also apply character level embeddings from words to deal with out-of-vocabulary (OOV) problems of using pre-trained word embeddings. Character embeddings were initialized randomly and trained with the whole network. Fig. 2 shows the overall architecture. An forward LSTM computes a representation of the left context of the sentence and a second backward LSTM that reads the

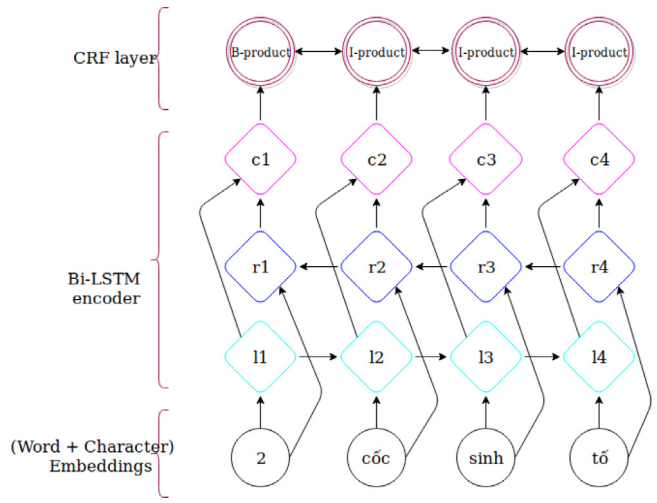

Fig. 2. A framework of using biLSTM-CRFs in recognizing users' contexts.

same sequence in reverse. These representations are concatenated and linearly projected onto a layer whose size is equal to the number of distinct contexts. We then use a CRF as previously described to take into account neighboring tags, yielding the final context predictions for every word.

CNNs

biLSTM captures the information contained in whole sentences. However, LeCun and Bengio (1998) argued that long sentences could contain information unrelated with the target entities, and hence, in domains with long sentences (which many user utterances belong to), the utilization of local information rather than whole sentences may help improve precision. CNNs allow the model to extract local information between a target word and its neighbors, and hence leverages the local contexts based on n-gram character and word embeddings via CNNs. The architecture is shown in Fig. 3 and undergoes through the following steps.

1. Generate the character and word embeddings and then concatenate all embeddings into a combined vector.
2. Extract each word's local features with several kernel sizes. The kernel size is equal to the size of a convolutional window across k tokens ($k = 1, 3, 5$). Suppose we are selecting a word indexed by i in a sentence from feature maps f_j convoluted by kernel size j.

$$O_i = \tanh(f_j[i - j + 1 : i]) \tag{6}$$

3. Apply CRF to model labels jointly based on the output of CNNs.

5. Datasets

This section introduces four corpora to analyze user utterances in e-commerce chatbots. The annotation process is first described, then some statistics are shown. The annotation process in Fig. 4 includes four main steps as follows:

1. *Collecting raw data*: We collected raw data from a history log of a famous restaurant and several forums, social websites. The raw texts cover a wide range of different variations in product names, dates/time, addresses.
2. *Pre-processing*: The data is pre-processed to remove html tags, emotion icons, sticky words, etc.
3. *Label Designing*: For each context type, we designed a corresponding set of labels which is useful in helping bots to understand what users say.

Table 1
Some statistics about the number of samples in the intent detection corpus.

Intents	Num	Intents	Num	Intents	Num
1. ordering	1205	6. promotion	76	11. close-open	55
2. shipping	1483	7. greetings	121	12. payment	190
3. cancel order	130	8. show menu	185	13. bye	53
4. deny/reject	117	9. thanks	80	14. others	1560
5. agree	92	10. asking PI	568		

Total samples: 5915

Table 2
Some statistics about the number of PI samples per PI type in the corpus of PI.

Types of PI	Quantities	Types of PI	Quantities
1. number	1322	5. uom	131
2. category	1231	6. extra-attribute	670
3. packsize	277	7. brand	664
4. attribute	641		

Total samples: 4936 product descriptions

4. *Manual Tagging*: Two people are required to manually tag raw data using the predefined sets of labels designed in the previous step.

To measure the inter-annotator agreement, the Cohen's kappa coefficient (Cohen, 1960) is used. The following sub-sections describe in detail each dataset.

5.1. A corpus about users' intents in an e-commerce domain

By observing the chat-logs, we realized that more than 90% of users' chat texts belonging to some common categories. Specifically, Vietnamese customers usually ask about the menu which contains their favorite foods/drinks, ask for any promotion, shipping policy, information of their desired products, etc., then make ordering, cancel previous wrong orders, deny/reject some information to correct their orders, finally say thanks before closing their chat sessions. In addition, there are also some customers having interests in closing time, opening time of the shop. A minority of chat-logs belongs to the intent of recruitment information, chit-chat, or asking unrelated things, etc. This group is classified as others. At the end, we finalized the details of 13 main categories of intentions (excluding other) with lots of training samples as shown in Table 1. This dataset was labeled by two people independently. The Cohen's kappa coefficient of our corpus was 0.85, which is usually interpreted as almost perfect agreement.

5.2. A corpus about product description on a retail domain

PI in selling online products usually consists of the following types:

- *Category*: Product types having particular shared characteristics, e.g. in the drinks domain, we may have categories of *coffee*, *tea*, *juice*, etc.
- *Attribute*: More details about products, e.g. *hot* or *cold*.
- *Extra-attribute*: Some remarks about products, e.g. *little sugar*, etc.
- *Brand*: A variation of products, e.g we may have different types of juice such as *orange juice*, *lemon juice*, etc.
- *Packsize*: Product sizes provided such as *small*, *medium*, and *large*.
- *Number*: Product quantities that customers want to order.
- *Uom*: the standard packing unit of measurement.

Using this label set, two people were asked to annotate that data at two levels. The number of training examples per PI type is indicated in Table 2. The Cohen's kappa coefficient of our corpus was 0.92. It suggests that the agreement between two annotators is very high.

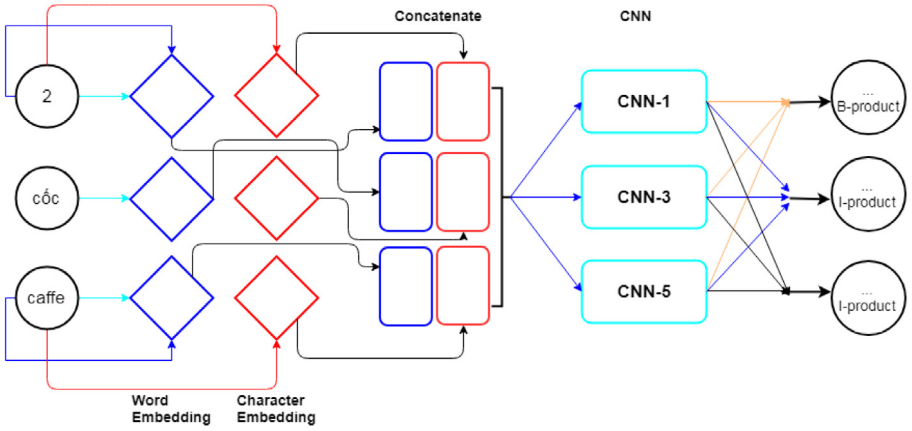

Fig. 3. A framework of using CNNs in recognizing PI.

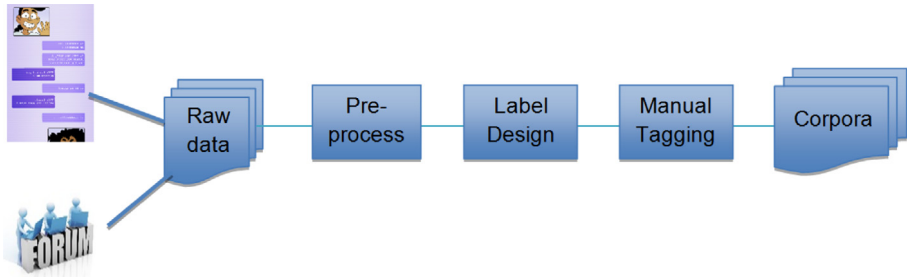

Fig. 4. The process of building new corpora in Vietnamese.

No.	Entities	Quantity	Examples
1	Dates/Time	6408	ngày mai sau giờ hành chính *tomorrow after office hours, etc.*
2	time	3989	giờ cụ thể: 8 giờ, giờ hành chính, ... *specific date: 8 o'clock, in office hours, etc.*
3	prefix	3607	sau, trước, đúng, ... *after, before, right, etc.*
4	date	3079	specific date: 23/5 *May 23rd , etc.*
5	period_of_day	2964	buổi sáng, buổi trưa, ... *in the morning, at noon, etc.*
6	urgent	241	giao luôn, ngay bây giờ *ship now, right now, etc.*
7	day_of_week	222	hàng ngày, thứ 2 *everyday, Monday, etc.*
8	month	151	tháng 5, tháng tới, ... *May, next month, etc.*
9	suffix	88	trở đi, về sau, ... *downwards, forward, after, etc.*
10	holiday	70	dịp 30/4, ngày của mẹ *April 30th occasion, mother's day, etc.*
11	year	28	năm 2018, sang năm, ... *year of 2018, next year, etc.*
12	week	27	tuần này, tuần đầu tháng 8 *this week, the first week of August, etc.*
13	anytime	27	khi nào cũng được, lúc nào cũng okie *anytime, whenever, etc.*

Fig. 5. Some statistics and examples about entities in Vietnamese dates/time.

5.3. A corpus about shipping dates/time

From the raw data, two annotators were hired to manually filter out 5234 sentences that may contain dates/time information. Then, annotators were required to assign labels at two levels. At level 1, any sequence of texts that appears to be a valid date or time will be labeled as *Dates/Time*. Then, at level 2 these dates/time will be decomposed into several elements which are useful for bots in understanding the dates/time implied by users. 10 main types of dates/time are considered and presented in Fig. 5. The Cohen's kappa coefficient of our corpus was 0.83.

5.4. A corpus about shipping addresses

From the raw data, two people were hired to manually filter out 2465 sentences that may contain Vietnamese addresses. By conducting a preliminary scan of several addresses, we realized that addresses usually have the following form: in general, the specific part (e.g numbers of houses) comes first and followed by lane number of street names, commune, districts and provinces. This specific part is commonly divided into several types as follows:

- *To locate in urban areas*: It is subdivided into some more detailed labels including street, number, lane, alley, crossroad, building names, and apartment building complex.
- *To locate in rural areas*: It is subdivided into more detailed labels including group and hamlet.
- *To locate a relative area which is close to an easier-to-locate area*: we marked this area by using the prefix label (*opposite to, next to, etc.*) and the suffix label (*300 m from, etc.*) to relatively locate the referred address.

Finally, we obtained 15 detailed labels to capture address information from users as shown in Fig. 6. The Cohen's kappa coefficient of this corpus was 0.81. This number is interpreted as almost perfect.

6. Experiments

This section firstly present strong baselines for each task. These baselines incorporate manually engineered features to train the models. Then, experimental setups, experimental results of each task, and some discussions are presented.

6.1. Baselines

For the task of intent detection, the baseline is the re-implementation of the previous work (Ngo et al., 2016) using MaxEnt classifiers. In

No.	Entities	Quantity	Examples
1	address	2585	B3 Khương Thượng, phố Tôn thất Tùng *B3 Khuong Thuong, Ton that Tung street*
2	street	1825	phố Kim Mã *Kim Mã street*
3	district	1144	quận thanh xuân *Thanh xuân district*
4	number	1002	số 123/23 *123/23*
5	city/province	829	tỉnh Hà Nam *Ha Nam province*
6	project	654	khu đô thị Golden palace *Golden palace apartment building complex*
7	location	582	cửa hàng hoa tươi Hàn Quốc *fresh Korean flower shop*
8	prefix	433	gần, đối diện, cạnh, ... *near, opposite to, next to, etc.*
9	commune	343	phường Hai bà Trưng *Hai ba Trung commune*
10	lane	283	ngõ Thông Văn *Thong Van lane*
11	building-name	253	tòa CT1A *CT1A building*
12	suffix	47	cách .... khoảng 300m *about 300m far from ....*
13	number-floor	46	tầng 1, tầng 23 *first floor, 23rd floor*
14	alley	44	hẻm 2 *alley 2*
15	group	44	tổ 3, khu tập thể Thanh Xuân *group 3, Thanh Xuân group*
16	crossroads	27	ngã 3, ngã tư *three-way crossroad, four-way crossroad*

Fig. 6. Some statistics and examples about entities in Vietnamese addresses.

this baseline, we also extract word unigram and word bigram, regular expressions for capturing common dates/time, URLs, phone numbers, etc.

For the task of context extraction, we compare with the traditional methods which add only hand-crafted features to CRFs. We incorporate the following features which are effectively used in extracting entities (Tran and Luong, 2018):

- Word unigram and word bigram
- Prefixes/suffixes: the first/last letters of tokens (up to four letters).
- Word shape: token-level evidence for "being a particular entity" such as whether a token is a valid, first letter capitalized, etc.

For unigram and bigram, we consider the context of $[-2, -1, 0, 1, 2]$. It means that when extracting features f, we have $f[-2]$, $f[-1]$, $f[0]$, $f[1]$, and $f[2]$ for unigram; and $f[-2]f[-1]$, $f[-2]f[0]$, $f[0]f[1]$, and $f[1]f[2]$ for bigram.

To make the baseline stronger, new features based on the Brown clustering algorithm (Brown et al., 1992) are proposed. These features have not been utilized in previous researches. The input to the algorithm is a set of words and a text corpus. In the initial step, each word belongs to its own individual cluster. The algorithm then gradually groups clusters to build a hierarchical clustering of words. We then use this word representation as features.

6.2. Experimental set-ups

6.2.1. Text preprocessing

Pre-processing is one of the key steps in a typical system working with social texts. With the datasets collected, it can be seen that these informal texts usually do not conform to regular patterns of our official language. So, there existed many minor errors in the original utterances as discussed in Section 1. Each utterance was processed by various steps as follows:

- Remove special characters (e.g = , < , @ , $, :)), and icons.
- Split words which stick together (e.g '*cho tôisinh tô' nhé*/(give me smoothie)' is split to '*cho tôi sinh tô' nhé*').
- Correct elongate words (e.g '*alooooo*' (*hello*) is replaced by '*alo*').
- Because the datasets are crawled directly from users' chat logs and forums, there are many freestyle letters. After observing and doing surveys, we just decide to replace typical words with its correct one (e.g one negation word, '*không/no*', can be written as '*khong*', '*ko*', '*khg*', '*k*', etc.).

In addition, unlike Western languages, Vietnamese words are not separated by white spaces. Therefore, word segmentation is usually recognized as the first step for many NLP tasks. It disambiguates intents' meanings and increases the detection quality in general. For this reason, we also performed word segmenting to break sentences into separated words by using the Pyvi library.[4]

6.2.2. Pre-trained word embeddings and Brown word clustering

To create word embeddings, we collected the raw data from Vietnamese newpapers (≈ 7 GB texts) to train the word vector model using Glove.[5] The number of word embedding dimensions was fixed at 50, the number of character embedding dimensions was fixed at 25.

Moreover, these raw texts were also used to induce clustering over words using Brown clustering algorithm. The number of clusters was set at 200. Features were extracted using 4-bit and 6-bit depth.

6.2.3. Evaluation metrics

The system performance is evaluated using precision, recall, and the F_1 score as in many classification and sequence labeling problems as follows:

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

For the task of context extraction, TP (True Positive) is the number of contexts that are correctly identified. FP (False Positive) is the number of chunks that are mistakenly identified as a context. FN (False Negative) is the number of contexts that are not identified.

For the task of intent detection, TP is the number of utterances that are correctly detected as the currently-considered intent t. FP is the number of utterances belonging to t but mis-identified to another intent. FN is the number of utterances of t but not recognized by the models. We also report the overall AUC numbers for the entire ROC and precision–recall curves to provide additional comparability of the results.

6.2.4. Model training

In performing experiments, we implemented the framework using CNNs and bi-LSTMs for detecting intents using keras libraries. For extracting contexts, we exploited three available tools with some mofifications to fit our task:

- CRFs: use the library of pyCRFsuite
- BiLSTM-CRFs: https://github.com/glample/tagger
- CNN-CRFs: https://github.com/valdersoul/GRAM-CNN

For each experiment type, we conducted 5-fold cross-validation tests. The hyper-parameters were chosen via a search on the development set. We randomly select 10% of the training data as the development set.

To detect intents, we use rectified linear units, filter windows of 3, 4, 5 with 100 feature maps each, dropout rate of 0.5, mini-batch size of 50 for CNNs. For bi-LSTMs, we set the number of epochs equals 100, the batch size as 20, early stopping as $True$ with 4-epoch patience, dropout rate of 0.5.

To extract contexts, we use filter windows of 2,3,4 and 5, dropout rate of 0.5, number of epochs of 100, optimization methods of mini-batch SGD of batch-size 20 for CNNs. For biLSTMs, we set dropout rate

[4] https://github.com/trungtv/pyvi

[5] https://github.com/standfordnlp/GloVe

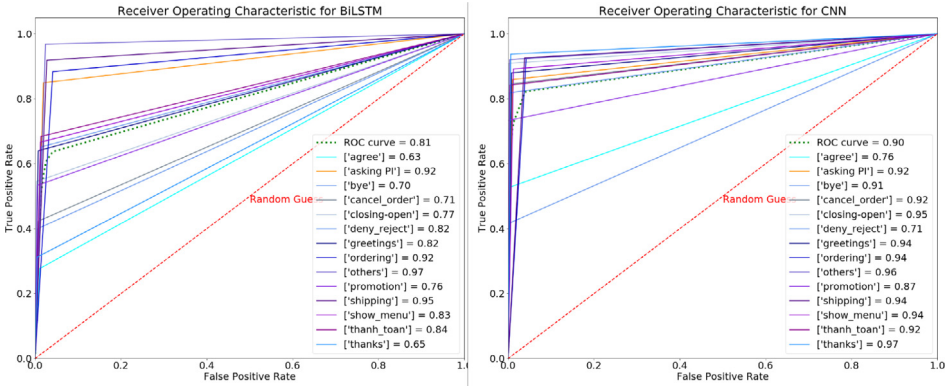

Fig. 7. ROC curves of two neural methods: biLSTMs and CNNs. (the averaged ROC curves are indicated by dotted lines).

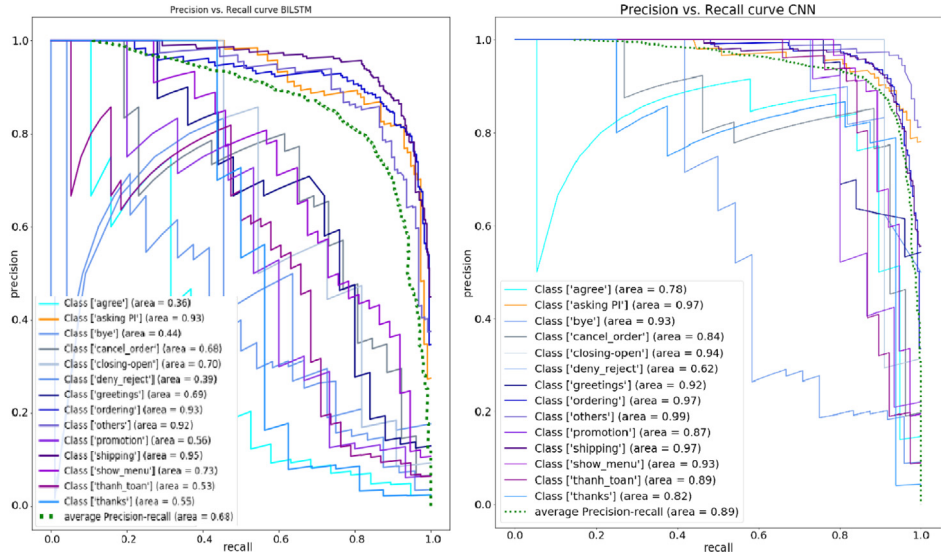

Fig. 8. Precision–Recall curves of two neural methods: biLSTMs and CNNs.

of 0.5, learning rate of 0.005, optimization methods of mini-batch SGD of batch-size 50, number of epochs at 100. In the following sections, we summarize two main types of experiments to parse intents and extract contexts from user utterances in AI bots.

6.3. Experimental results of detecting intents

Table 3 indicated experimental results of the intent detection using evaluation metrics of precision (P), recall (R), and the F1 score. We also presented two other useful metrics called ROC curves and precision–recall curves as illustrated in Figs. 7 and 8. They gives us a measurement of classifier performance without the need for a specific threshold. Better methods will have higher AUC areas. The results suggest that using neural models is superior to the baseline. Specifically, the baseline yielded the lowest performance of 77.01% in the F1 score. BiLSTMs enhanced the F1 measure in comparison to MaxEnt by 2.45%. For the best performance, we achieved 82.32% in the F1 score, 0.90 averaged ROC AUC score and 0.89 averaged precision–recall AUC score using CNNs.

Among intent types, some intent types (such as *agree, deny or reject*) are more difficult to predict than other types (e.g. *asking PI, ordering, shipping, and others*). Some intent types are usually misclassified into others. For example: several utterances are actually multi-classes rather than single class as in our setting (e.g. the utterance *'I change my mind to hot coffee, please!'* belongs to both *deny/reject* and *make ordering* intents.); some utterances when adding/removing just one or two words, they change intent types (e.g. when adding the word '*not*' into the utterance '*I agree*' of the intent '*agree*', the sentence changes the intent type into '*deny or reject*').

Observing the output of the best model, we realized some errors caused by spelling errors, the OOV problem in working with pre-trained word embeddings or non-standard languages in social media texts as will be explained in more detail in Section 6.4.3.

Table 3
Experimental results of detecting intents using MaxEnt, biLSTM, and CNN.

	MaxEnt			biLSTM			CNN		
Intents	P	R	F_1	P	R	F_1	P	R	F_1
ordering	88.32	93.60	90.89	89.57	94.1	91.78	91.15	93.27	**92.2**
shipping	92.61	94.67	**93.63**	91.64	93.18	92.41	91.88	93.86	92.86
cancel order	86.92	72.09	**78.81**	82.24	68.22	74.58	77.17	75.97	76.56
deny/reject	58.42	50.86	54.38	67.27	63.79	**65.49**	66.07	63.79	64.91
agree	69.81	40.66	51.39	70.93	67.03	**68.93**	77.46	60.44	67.9
promotion	76.79	57.33	65.65	68.66	61.33	64.79	84.38	72	**77.7**
greetings	76.74	79.20	77.95	72.92	84	78.07	71.01	96	**81.63**
show menu	88.34	79.56	83.72	86.23	79.56	82.76	86.98	81.22	**84.00**
thanks	90.16	69.62	78.57	90.00	79.75	**84.56**	93.75	75.95	83.92
asking PI	888.81	91.17	89.97	90.61	88.69	89.64	91.65	93.11	**92.38**
close-open	91.89	62.96	74.73	87.76	79.63	83.5	93.62	81.48	**87.13**
payment	87.5	81.48	84.38	86.53	88.36	87.43	88.66	91.01	**89.82**
bye	79.41	51.92	62.79	83.33	57.69	68.18	84.85	53.85	65.88
others	87.68	95.22	91.3	95.45	93.63	94.53	97.83	93.47	**95.6**
Average	83.1	72.88	77.01	83.08	78.5	80.47	85.46	80.39	**82.32**

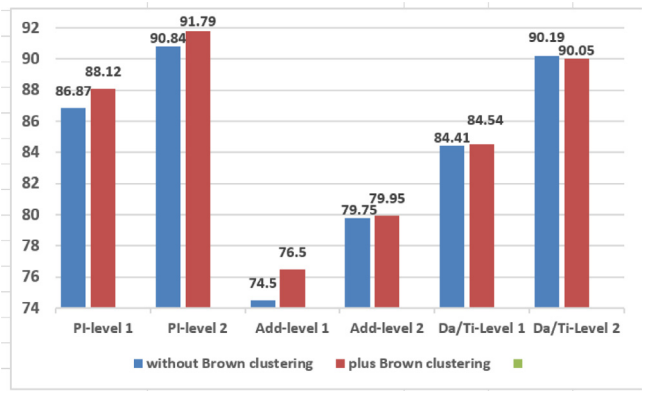

Fig. 9. Experimental results in the F1 score of using CRFs with/without Brown clustering as features for three types of contexts.

6.4. Experimental results of extracting contexts

In extracting different contexts, two types of experiments are performed. The first one is to show the performance of two baseline methods which integrate and do not integrate features generated from using Brown clustering. The second one is to illustrate the strength of the proposed methods over the stronger baseline.

6.4.1. Experimental results of two baseline settings

Fig. 9 indicated experiment results of two baselines with/without using additional features. It indicated that integrating Brown clustering features yielded better performance in the F1 score for all three datasets except for the case of extracting dates/time at level 2. However, the decrease is not remarkable (only 0.14%). This result expressed that using Brown clustering was quite effective and overall improved the performance of the systems in both layers on three datasets. In the next sub-section, we show experimental results of the proposed methods and this stronger baseline (named *hand-crafted*_CRF).

6.4.2. Experimental results using two deep neural networks

Table 4 showed the experimental results of the baseline *hand-crafted*_CRF as well as the two proposed methods on three datasets. We now analyze these results for each context in more detail.

Product descriptions: in Level 1, *hand-crafted*_CRF produced the lowest performance with the F-measure of 88.12%. By using neural networks, we could boost the performance of the system. Specifically, in comparison to this baseline, CNNs and biLSTMs improved the performance by 2.79% and 2.4%, respectively. Between these two neural models, CNNs yielded a slightly higher results than biLSTMs. For recognizing PI in Level 2, CNNs also yielded the best performance with a F-measure of 93.08%, which is 1.29% higher than the second best system - *hand-crafted*_CRF. The lowest performing approach is biLSTMs with the F-measure of 90.11%. Among PI, detecting some information (such as *product-number, product-attribute*) is easier than others (such as *product-brand, product-extra-attribute*). However, in general the proposed method could produce promising results for all PI of the dataset.

Addresses: In recognizing addresses in Level 1, the baseline still yielded the lowest performance. The experimental results suggest that among two neural network architectures, biLSTMs produced slightly higher results with 79.33% in the F1 score. However, CNNs could produce competitive results. In Level 2, CNNs outperformed the other two methods and yielded a significant performance improvement (approximately 3% in the F1 score) over the second best methods. On average, using CNNs we obtained 82.23% in the F1 score.

Among address sub-types, some contexts are easier to detect such as *alleys, cities, crossroads, districts, lanes, numbers, streets*. Unfortunately, there also exist some contexts which are more difficult to detect such as *suffixes, prefixes, locations, building_names*. The reasons will be explained later.

Table 4
Experimental results of two layers using *hand_crafted*_CRFs, biLSTM-CRFs, and CNN-CRFs on three types of contexts.

	hand-crafted_CRF			biLSTM-CRFs			CNN-CRFs		
	P	R	F_1	P	R	F_1	P	R	F_1
Product Information - Level 1									
prod_desc	87.36	88.9	88.12	89.71	91.35	90.52	90.6	91.24	**90.91**
Product Information - Level 2									
attribute	95.81	94.38	95.08	93.69	95.63	94.63	95.9	97.24	**95.8**
brand	85.32	86.86	86.07	82.44	83.24	82.77	89.38	88.64	**88.98**
category	90.23	90.63	90.42	86.24	88.45	87.32	91.44	91.9	**91.67**
extra_attr	87.43	87.78	**87.53**	87.89	86.76	87.26	88.83	86.24	87.39
packsize	89.94	90.12	90.01	85.03	86.82	85.84	91.62	93.14	**92.36**
number	96.07	94.67	95.36	95.24	95.35	95.28	95.88	95.92	**95.89**
uom	89.61	93.53	91.48	88.8	91.73	90.16	92.12	92.33	**92.19**
Average	91.6	91.98	91.79	89.39	90.86	90.11	92.95	93.21	**93.08**
Address - Level 1									
address	78.68	74.44	76.50	80.15	78.56	**79.33**	79.26	77.83	78.53
Address - Level 2									
alley	82.28	70.84	74.78	76.38	69.98	72.49	92.64	88.64	**90.42**
building_name	68.23	57.16	61.63	56.89	56.79	56.70	68.01	68.24	**67.7**
city	96.96	95.05	95.97	95.48	95.62	95.54	96.95	96.05	**96.48**
commune	68.93	57.11	62.37	62.71	57.94	60.04	70.21	60.9	**65.03**
crossroads	75.81	77.00	75.60	84.26	77.68	79.28	86.67	86.29	**85.86**
district	92.96	92.59	**92.76**	77.96	75.75	76.67	90.85	92.3	91.55
group	75.83	52.23	60.05	51.29	37.19	42.82	65.38	59.55	**61.56**
lane	90.79	86.32	88.42	69.89	68.70	69.26	89.15	91.43	**90.19**
location	53.60	50.01	51.61	58.91	64.01	**61.33**	61.36	55.21	57.8
number	87.58	88.63	88.10	81.01	87.17	83.74	90.49	91.33	**90.89**
number_floor	81.33	58.99	66.49	75.61	66.66	70.71	81.15	79	**79.04**
prefix	74.19	66.41	**69.98**	69.24	69.62	69.35	72.93	66.61	69.56
project	68.83	67.34	68.01	68.68	70.56	69.58	72.19	75.29	**73.7**
street	80.73	82.49	81.60	74.02	73.13	73.21	83.53	85.19	**84.34**
suffix	37.44	18.55	24.46	28.87	18.40	21.94	25.42	21.32	**21.96**
Average	81.27	78.69	79.95	79.07	79.59	79.33	82.66	81.83	**82.23**
DATESTIME - Level 1									
Dates/Time	85.84	83.28	84.54	86.52	85.06	85.59	86.61	85.37	**85.98**
DATESTIME - Level 2									
anytime	100	52.92	**63.33**	58.33	36.67	42.99	55.95	40.71	46.54
date	89.92	89.87	**89.88**	88.49	89.68	89.14	90.26	89.5	**89.88**
day_of_week	72.41	68.61	70.3	65.36	65.84	65.41	76.57	76.48	**76.28**
holiday	70.82	56.72	**62.8**	62.35	59.02	60.34	65.98	61.49	61.49
month	91.19	85.83	88.33	92.78	86.89	89.64	91.55	88.27	**89.81**
period_of_day	91.74	92.53	92.13	91.98	92.09	92.01	92.61	91.95	**92.28**
prefix	90.67	92.73	91.68	92.44	92.1	92.26	92.80	92.48	**92.63**
suffix	68.49	56.71	61.23	62.92	59.71	60.88	68.42	56.43	**61.30**
time	92.10	91.40	91.74	92.08	91.08	91.57	93.81	92.25	**93.02**
urgent	59.82	50.04	54.30	55.10	55.21	54.92	61.79	57.48	**58.96**
week	71.43	62.76	**64.24**	54.67	50.32	47.62	60.55	60.32	57.45
year	64.95	54.67	**58.32**	68	43.78	51.29	61.67	42.67	48.28
Average	90.15	89.94	90.05	89.86	89.58	89.53	91.34	90.16	**90.58**

Dates/Time: In recognizing dates/time in Level 1, the results demonstrated once again that using deep neural networks is quite effective. CNNs and LSTMs increased more than 1% in the F1 score over the baseline. In Level 2, the results showed that deep neural networks are able to perform as well as traditional machine learning methods using a variety of manually engineered features. CNNs slightly boosted the performance up to 90.58% in the F1 measure. This result is consistent with the previous observation on the address and product description datasets. Compared to addresses, dates/time seems to be easier to recognize. This is not surprising because on average, the length of dates/time (3.7 words each) is shorter than addresses' (8 words each), and hence easier to recognize.

Among dates/time sub-types, some contexts are easier to detect such as *dates, period_of_days, time, months*. Unfortunately, other contexts are

Cases	Error Examples
1. Spelling errors	.... mười năm (đúng là "*mười lăm*") .... Ten years (should be "fifteen")
2. Non-standard languages	... T4 này .... (có thể là "*thứ 4*" hoặc "*tháng 4*") ... this T4 ... (may be "*Wednesday*" or "*April*")
3	... sinh tố size S vị việt quất ... ... smoothie size S with the flavour of biberry ....
4	... 1 cốc hazenut loại cốc to ... ... 1 cup of hazenut with the cup of large size ...
5	... cách BigC Thăng Long khoảng 300m ... ... it is about 300m far from BigC Thang Long ... ... Trong trưa nay tầm 12h hoặc trước đó ... ... at this noon about 12h or before that ....
6	... nhận hàng luôn .... receive products right after ... (might be '*always*' or '*right now*')
7	... nửa cuối tháng 10 ... (không bắt được prefix "*nửa*") ... half end of October ... (could not capture the prefix "*half*")
8	... qua sinh nhật thứ 4 .... (nhận nhầm thành "*thứ 4*") ... over the 4th birthday anniversary .... (misrecognized as "*Wednesday*")

Fig. 10. Some error examples (texts in red are mis-recognized) of the best systems using CNN-CRFs models. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

more difficult to detect such as *anytime*, *holidays*, *weeks*, *years*. In the next sub-section, we will show some main reasons.

6.4.3. Error analysis

Observing the output of the best system using CNNs, we saw some typical errors which can be classified into three main types as follows:

- Spelling errors: Social texts contain various types of spelling errors such as *extra letter, missing letter, incorrectly repeated consonants, confusion with similar words, etc.*. One typical example is given in Case 1 (see Fig. 10), which result in recognizing '*ten*' rather than '*fifteen*' as a number.
- Non-standard languages and hybrid linguistics: An alphabet soup of acronyms, abbreviations, and neologisms, and new words grown up on social media, which causes difficulty in recognizing correct contexts. (e.g. Case 2)
- The performance of Vietnamese word segmenters: Most current Vietnamese segmenters were built on general texts. Thus, the quality is not good enough when applied on social media texts. This also leads to OOV problems in working with pre-trained word embeddings.

Going a further step, we also performed analyzing errors of each context type and observed some mis-recognition due to ambiguities as follows:

- Product descriptions:
  - Some categories are mis-recognized as attributes, and categories are mis-recognized as brands. For example, the system mis-classified '*tea with grape flavor*' as a product category. The ground truth should be '*tea*' as a product category, and '*grape flavor*' as its brand.
  - In some sentences, when referring attributes of a product, users occasionally add some extra descriptions in informal speeches (e.g. Cases 3, 4) which make our system confused (not learnt from training data).
- Addresses:
  - Some *suffixes* are address endings that add more explanation to the main address. They are usually long and quite tough to be recognized (e.g. the first example in Case 5).
  - *Location* is usually long and go right before the address, and not clearly separated from other elements. Hence, it is usually mis-recognized into others such as *streets* and *prefixes*.
  - Many new *building names* is constantly appearing in Vietnam, especially in big cities. These names are usually in English and may also include street names or district names. Thus, it might be confused with other elements likes *communes*, *districts*, etc.
- Dates/Time:
  - Similar to address recognition, some dates/time suffixes are quite tough to be recognized (e.g the second example of Case 5).
  - Several words have different meanings depending on the their surrounding contexts (e.g Case 6)
  - Some cases require deeper semantic meaning to distinguish between different dates/time elements (e.g Case 8). Another example is the case of the *prefix* as shown in Case 7.

To overcome these errors, we should improve the quality of word segmenters on social texts. In addition, the size of datasets should be expanded to cover the diversity of the Vietnamese language used among youth on social media. If possible, deeper semantic features should be integrated to enrich the models.

6.4.4. Discussion

This section discusses the imbalanced data problem and how to reuse the methods and pre-trained models for other applications.

Imbalanced Data Problem: The intent corpus contains some classes with a few examples which may lead to imbalanced dataset problems. Experimental results indicated that the performance of these minority classes is quite low in comparison to other majority classes' performance. Thus, we did try some strategies to deal with this problem. Several methods have been used such as SMOTETomek, BorderlineSMOTE, ADASYN, etc. Unfortunately, the results showed that the best ADASYN technique could not improve the performance of the models. Specifically, using biLSTMs, the experimental results were slightly decreased in the F1 score by nearly 1%. This may be caused by its disadvantages of over generalization and variance as discussed in He and Edwardo (2009). Due to the time constraint, we left this problem for the future work.

Model Generalization: The proposed method can be generalized to be used for different domains and applications as long as some requirements must be satisfied. For examples, the input texts should be word-segmented and the pre-trained word embedding for the specific language/domain should be also provided. Even some of the pre-trained models can be directly reused such as:

- The models of recognizing addresses, time and intents can be used for building ordering chatbots of different domains (e.g. clothes, beauty product, etc.) rather than drinks and beverages.
- The models of address and time recognition can be used for building other chatbot applications (e.g. making an appointment with customer services centers, booking an appointment with doctors, etc.) or doing other works of enabling the computers to identify and translate various possible formulations of date-time formats into a data understandable by an API such as Merlo and Pasin (2018) done for French; or recognizing of postal address on web documents such as the work of Blumenstein and Verma (1998) done for English).

7. Conclusion

This paper described the task of deeply analyzing user utterances. The main points are to *identify intents* implied by user utterances and *parse the content to extract useful contexts* for AI bots. This work is dedicated to the Vietnamese language which poses several challenges. To achieve these goals, a framework which modeled the former as a classification problem and the latter as a two-layer sequence labeling problem was proposed. In both tasks, instead of using conventional methods, we explored state-of-the-art deep neural networks to learn useful features at both character and word levels. To verify their effectiveness, four new corpora were introduced to conduct extensive experiments. The experimental results demonstrated that in general using neural networks could indeed improve the performance of the system over conventional ones. Overall, in detecting intents, we achieved the best F-measure of 82.32% using CNNs. In extracting contexts of Level 1, we got the best performance of 90.91% in extracting PI, 79.33% in extracting addresses, and 85.98% in extracting dates/time. In extracting contexts of Level 2, on average we obtained the best F1-scores of 93.08% in parsing product descriptions, 82.23% in parsing dates/time, and 90.58% in parsing addresses into more detailed elements. These results are very promising to help bots better understand what users say.

In the future, we intend to adapt the proposed method so that it can work well in other domains rather than e-commerce one. Studying deeper semantic features extraction might also be another future direction. Moreover, as can be seen that the new datasets are quite imbalanced. Hence, another direction is to investigate different techniques to deal with this problem.

Acknowledgment

This work was funded by Vietnam National University, Hanoi under the project QG.19.59.

References

Ali, D., Habash, N., 2016. Botta: An Arabic Dialect chatbot. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations. pp. 208–212.

Blumenstein, M., Verma, B., 1998. A neural network for real-world postal address recognition. In: Soft Computing in Engineering Design and Manufacturing. Springer, pp. 79–83.

Brixey, J., Hoegen, R., Lan, W., Rusow, J., Singla, K., Yin, X., Artstein, R., Leuski, A., 2017. SHIHbot: A Facebook chatbot for sexual health information on HIV/AIDS. In: 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue. Association for Computational Linguistics, pp. 370–373.

Brown, P., deSouza, P., Mercer, R., Pietra, V., Lai, J., 1992. Class-based n-gram models of natural language. J. Comput. Linguist. 18 (4), 467–479.

Cohen, K., 1960. A coefficient of agreement for nominal scales. Educ. Psychol. Meas. 20 (1), 37–46.

Cui, L., Huang, S., Wei, F., Tan, C., Duan, C., Zhou, M., 2017. SuperAgent: A customer service chatbot for e-commerce websites. In: Proceedings of ACL 2017, System Demonstrations. Association for Computational Linguistics, pp. 97–102.

Goo, C., Gao, G., Hsu, Y., Huo, C., Chen, T., Hsu, K., Chen, Y., 2018. Slot-gated modeling for joint slot filling and intent prediction. In: Proceedings of the 2018 Conference of the NAACL: Human Language Technologies. pp. 753–757.

He, H., Edwardo, A., 2009. Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. 21 (9), 1263–1284.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Hu, J., Wang, G., Lochovsky, F., Sun, J.-t., Chen, Z., 2009. Understanding user's query intent with Wikipedia. In: Proceedings of the 18th International Conference on World Wide Web. pp. 471–480.

Huang, P., He, X., Gao, J., Deng, L., Acero, A., Heck, L., 2013. Learning deep structured semantic models for web search using clickthrough data. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management. ACM, pp. 2333–2338.

Ji, Z., Lu, Z., Li, H., 2014. An information retrieval approach to short text conversation. arXiv:1408.6988 [Cs.IR].

Kato, T., Sumitomo, R., Nagai, A., Wu, J., Noda, N., Yamamoto, S., 2017. Utterance intent classification of a spoken dialogue system with efficiently untied recursive autoencoders. In: Proceedings of the SIGDIAL 2017 Conference. Association for Computational Linguistics, pp. 60–64.

Lafferty, J.D., McCallum, A., Perera, F.C.N., 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: 18th International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., pp. 282–289.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C., 2016. Neural architectures for named entity recognition. In: 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, pp. 260–270.

LeCun, Y., Bengio, Y., 1998. Convolutional Networks for Images, Speech, and Time Series. the Handbook of Brain Theory and Neural Networks. MIT Press Cambridge, MA, USA, pp. 255–258.

Li, C., Li, L., J., Q., 2018. A self-attentive model with gate mechanism for spoken language understanding. In: Proceedings of Conference on Empirical Methods in Natural Language Processing. ACL, pp. 3824–3833.

Mendoza, M., Zamora, J., 2009. Identifying the intent of a user query using support vector machines. In: String Processing and Information Retrieval. Springer, pp. 131–142.

Merlo, A., Pasin, D., 2018. An open source library for semantic-based datetime resolution. In: The 26th International Conference on Computational Linguistics: System Demonstrations. The COLING 2016 Organizing Committee, pp. 107–111.

Ngo, T., Nguyen, V., Vuong, T., Nguyen, T., S.B., P., Phan., X., 2016. Identifying user intents in Vietnamese spoken language commands and its application in smart mobile voice interaction. In: Proceedings of Asian Conference on Intelligent Information and Database Systems. pp. 190–201.

Qiu, M., Li, F., Wang, S., Gao, X., Chen, Y., Zhao, W., Chen, H., Huang, J., Chu, W., 2017. AliMe Chat: a sequence to sequence and rerank based chatbot engine. In: Annual Meeting of the Association for Computational Linguistics. pp. 498–503.

Shi, Y., Yao, K., Tian, L., Jiang, D., 2016. Deep LSTM based feature mapping for query classification. In: Proceedings of NAACL-HLT 2016. Association for Computational Linguistics, pp. 1501–1511.

Tran, O., Luong, T., 2018. Towards understanding user requests in AI bots. In: Proceedings of the 15th Pacific Rim International Conference on Artificial Intelligence. pp. 864–877.

Tran, P., Ta, T., Truong, Q., Duong, Q., Nguyen, T., Phan, X., 2016. Named entity recognition for vietnamese spoken texts and its application in smart mobile voice interaction. In: Proceedings of Asian Conference on Intelligent Information and Database Systems. pp. 170–180.

Wilcox, E., Levy, R., Morita, T., Futrell, R., 2018. What do RNN language models learn about filler–gap dependencies? In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP. ACL, pp. 211–221.

Yan, Z., Duan, N., Bao, J., Chen, P., Zhou, M., Li, Z., Zhou, J., 2016. Docchat: An information retrieval approach for chatbot engines using unstructured documents. In: Proceedings of ACL. pp. 516–525.

Yan, Z., Duan, N., Chen, P., Zhou, M., Zhou, J., Li., Z., 2017. Building task-oriented dialogue systems for online shopping. In: Proceedings of AAAI. pp. 4618–4626.

Zhu, Q., Li, X., Conesa, A., Pereira, C., 2017. GRAM-CNN: a deep learning approach with local context for named entity recognition in biomedical text. J. Bioinform. 1 (8), 1–8.