«Системні технології» 1 (138) 2022 «System technologies»
DOI 10.34185/1562-9945-1-138-2022-14
UDC 004.38

O.A. Lytvynov, M.O. Lytvynov

ON PRACTICAL ASPECTS OF USING OF THE FRAME-BASED SENTENCE PATTERNS IN INTEGRATIVE COMPUTER-AIDED LANGUAGE LEARNING SYSTEM

Abstract. The given work is devoted to resolution of the problems appeared during the attempt to build an integrative system using frame based patterns. The main problems resolved in the work are as follows. Words reordering, effective error handling, general rules of transformation, effec tive scenario description.
Key words: frame, frame based paradigm, second language acquisition, language learning sys tem, integrative approach.

Actuality. There are three key classes of CALL systems connected to three language learning approaches: behavioristic, communicative, integrative [2, 3, 4]. The systems of the first type do not allow enough authentic communication to be of much value and are focused on "drill and practice" principle [1]. Communicative systems try to involve the computer system encouraging students to generate original utterances rather than manipulate prefabricated language. From the point of view of communicative systems developers, making the system flexible to a variety of student responses causes the creation of an environment in which using the target language feels natural. Integrative systems are based on two technological components - multimedia computers and the Internet [1]. The main features of such systems are as follows: authentic linguistic environment simulation using the media which makes it natural to combine reading, writing, speaking and listening in a single activity; individual approach to each student; focusing on the content the students are interested in. The typical integrative system is Dustin program developed by the Institute for Learning Sciences at Northwestern University [5, 6].

The paper [7] represents a formal description of the scenario of the scenario-based integrative language learning system. The description is presented in a compact form and based on mathematical background. It allows to understand the specific and potential can be achieved by the system; a set of operations should be considered; phases of evolution of such systems; make a flexible architectural solution.

According to [7] the scenario can be represented as a set of steps. The key element of the step is a message which passes a meaning of the event to the user. The message consists of a number of projections of the message: text, formula, video-clip, audio, image, diagram, virtual agent actions etc. There are two types of the messages: task-oriented messages which cause interrupts of the flow requiring user reaction and context-oriented messages. Context-oriented messages form a background to make a natural involvement of the user into communication based on task-oriented messages (Fig. 1).

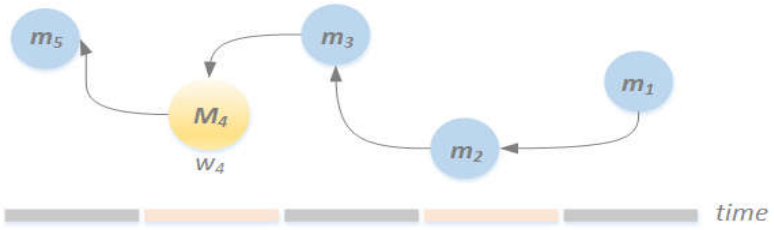

Figure 1 – The flow of the messages: context-oriented messages prepare the background for task-oriented message

Each task was defined as a 3-tuple “task definition – a set of weighted correct answers – a required time to perform the task”.

A set of weighted answers forms a pattern P^j which provides a set of pairs “expression - value”.

$$P^j =_{def} \{\langle s, v\rangle \big| s \in S^j, v \in \mathbb{Q}\},\ P^j \subseteq \mathcal{P} \tag{1}$$

And there was also introduced an operation ψ which can estimate a certain expression against a pattern of answers.

$$\psi(s, q^j{}_i) = \begin{cases} v, \exists s: \langle s, v\rangle \in P^j \\ 0, \nexists s: \langle s, v\rangle \in P^j \end{cases} \tag{2}$$

To make the interconnection between the user and the system more natural and effective means making the system flexible to a variety of student responses. This claim gets the developers faced the issue of a huge number of probable variants of acceptable answers connected to one task, considering their completeness, acceptability etc. It is obvious that this problem cannot be resolved simply by the enumeration of all the available variants. Even in connection with a simple sentence we have a huge number of variations: more or less acceptable, more or less complete etc. Thus, there should be applied a usable mechanism of patterns able to help teachers to describe a huge number of variants using compact structures.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

The paper [8] was devoted to resolving that problem. The solution is based on frame-based paradigm [9], according to which each sentence can be represented as a number of ordered slots (terminals and non-terminals) connected with the terms (variants of expression parts represented by the slots). Some slots are strongly required to be filled with the values others are optional ones. The collection of required slots forms the semantic core of the expression: the meaning cannot be passed if the slot is not filled with a proper value. Slots can be connected to a number of alternative values – the terms with the same meaning, some of which can be regarded as preferable. Some of these values can also be described by the complex structures(frames) but mostly they are equivalent to a word or a number of words. The order of the slots allows us to build syntactically right sentences.

Frame can be defined as a set of slots (Fig.1).

$$c =_{def} \vec{s} \tag{3}$$

We can describe the structure of slot as an ordered set of facets.

$$s_i =_{def} \vec{\varphi}_i,\ \vec{\varphi}_i \in \Phi \subseteq \Phi_1 \times \Phi_2 \times \dots \Phi_k \tag{4}$$

Domains represent the axes of slot's definition. For example, Φ_1 can represent the type of slot {terminal, nonterminal}, Φ_2 – number restriction used to describe the number of relationships of a particular slot that individuals can participate in etc.

We can say that two frames are similar if the sets of sentences generated by those frames are similar.

The language we use to describe the frames is very closed to IC (immediate constituent) patterns improved with new semantic marks. The description seems simpler and more understandable than the variant of pure frame-based language description. "<…>" – defines the borders of the slot. An asterisk ("*") placed before the denoted axis (the name of the terminal slot, e.g. Q, T, Tt, S) means that the slot is optional, "!" before the axis declares that the slot is required. The names of the slots can be selected randomly, their role is to represent a number of alternative slot values. Notably, that any part of the expression can be transformed into slot, means the slot is not connected to grammar structures (part of speech etc.). The value of the slot can be represented by the frame having some slots (Tt1 = today <*Tt>).

Let's look at the example.

*<*Q ><*T ><!S >*

Q1=as you're aware {.1}

Q2=as you know {.2}

*T1=today <*Tt> {.3}*

Tt1=in the evening {.1}

S1=we're going to be looking at your career options {.5}

Such frame produces the following set of sentences (decimal numbers in square brackets denotes the value of acceptance of the phrase):

We're going to be looking at your career options [.5]

Today we're going to be looking at your career options [.8]

Today in the evening we're going to be looking at your career options [.9]

As you know we're going to be looking at your career options [.7]

As you're aware we're going to be looking at your career options [.6]

As you know today we're going to be looking at your career options [1.0]

As you're aware today we're going to be looking at your career options [.9]

As you know today in the evening we're going to be looking at your career options [1.1]

As you're aware today in the evening we're going to be looking at your career options [1.0]

It is obvious, that the adding of only one required slot with two alternative values will double the number of generated sentences.

Task definition. The attractive idea of using such patterns suffers from following disadvantages. First, some of the languages have rigorous sentence structure (e.g. English, Japnese), others more flexible (e.g. Ukrainian, Russian), but it would not be a mistake if we say that every language allow to change words order in definite scope of restrictions (even the restriction of the sentence to 7-10 words). The question is how to process that type of allowable words moving. The next question considers some general rules of interpretation, e.g. according to the rule of indefinite article “a” becomes “an” before the word started with the vowel. When we have optional slots which can be excluded from the interpretation, we can face the situation when the result sentence can have such general rules mistakes. Of course, they could be solved by the rigorous frame description, but it would require more time of a person responsible for frames description and, therefore, could undermine the idea of simplicity of the system. Third question connected to error handling. When the user chooses a wrong word or a part of phrase instead of the right one, he/she have got the notification that his answer is incorrect and, in the best variant, how the most closed proper answer is looked like. But usually, users want to get some explanations about why the provided variant is not right. How to process the situation making the system more usable and friendly, increasing its interaction-orientedness.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

The last question is how to embed patterns in scenario scripts, how to make the process of scenario construction simpler and more effective. These are the questions the given work is devoted to.

Main part. First problem mentioned above connected to the ability of an expert (a person responsible for making and maintaining the patterns) to describe the allowable combinations of slots considering words conversions and reordering without significant efforts. In general, we face the situation when one sentence should be described by several frames. These frames can be connected to the same set of slots and in that case, we have got the clones of the same fame differed by the compositions of the slots. But this solution does not cover the range of situation connected with the problem. A more proper solution would be to assume that some of those frames could have their own sets of slot values and even introduce new slots.

In general, we impact with the structures which partially connected to the source set of slots and its values.

$$s_j =_{def} F^j =_{def} f^j{}_1 \mid f^j{}_2 \mid \dots f^j{}_i \mid \dots \mid f^j{}_{n'} \quad s_j \in \mathfrak{S} \tag{5}$$

where s_j – a sentence from the set of sentences $\mathfrak{S}$ which is described by the set of frames F^j. User's answer in response of proposition to provide the version of the sentence can be connected to only the one of the frames, and the expression $F^j =_{def} f^j{}_1 \mid f^j{}_2 \mid \dots f^j{}_i \mid \dots \mid f^j{}_n$ means that using "|" sign to separate alternatives.

$$\exists s_l \in s(f^j{}_k) \rightarrow \exists f^j{}_i \neq f^j{}_k \wedge s_l \in s(f^j{}_i) \tag{6}$$

Formula (6) defines the rule of sharing slots among different frames. It states that if a frame from the set of frames F^j has a slot, that slot can also be used by the other frame from the same set. $s(f^j{}_k)$ expression means the slots of the frame, and, consequently, s means function that transforms a frame into the set of slots which the frame consisted of, more precisely, it can be defined by the expression $s: f \rightarrow \vec{s}$.

$$\exists s_i \in s(f^j{}_k) \rightarrow \forall f \in F^j \backslash f^j{}_k . s_i \notin s(f^j{}_i) \tag{7}$$

Formula (7) defines the rule of existence of the slots which belongs to only one frame.

Different types of frames are shown in Figure 2. First can be regarded as the basic one. Second is the modification of the first one differed only by the slots order. Third is the frame which shares partially only one basic slot (only sv^{3+} part) and adds new slot s4 to the set of slots.

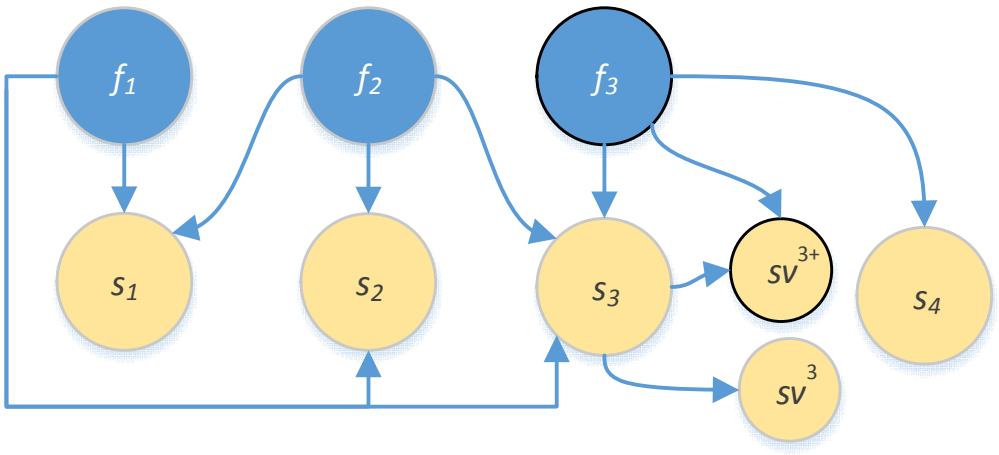

Figure 2 – Evaluation of the parts of the expression

To demonstrate the problem and the features of the provided solution let's see some examples. In brackets, we indicate a solution to the problem using the method mentioned above.

English examples:

1. The boy quietly went out ↔ The boy went out quietly. (Slots reordering)
2. After seeing the film, we discussed it ↔ We discussed the film after seeing it. (Slots reordering).
3. A lot of people worked for that plant ↔ There worked a lot of people for a plant. (Slots reordering plus additional terminal slot "There").
4. I have never seen such a beautiful park before ↔ Never before have I seen such a beautiful park (Slots reordering).
5. I would have felt better, if I had stayed at home ↔ I would have felt better, ~~if~~ had I stayed at home (Slots reordering and excluding "if" from the second frame).
6. Jack wouldn't take a taxi at night to go home, if he could stay at our place ↔ Jack wouldn't take a taxi at night to go home, ~~if~~ could he stay at our place (Slots reordering and excluding "if" from the second frame).
7. It wasn't a big problem for Mark ↔ For Mark it wasn't a big problem (Slots reordering).
8. The only person visible was the soldier near the castle's gates ↔ The only visible person was the soldier near the castle's gates (Slots reordering).

French examples:

1. Malheureusement, nous n'avons pas la possibilité de la faire ↔ Nous n'avons pas la possibilité de la faire malheureusement (Slots reordering).
2. Hier, il a plu ↔ Il a plu hier (Slots reordering).

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

3. Dans ce pays, la neige est rare ↔ La neige est rare dans ce pays (Slots reordering).

Chinese examples:

1. 今天我工作 ↔我今天工作 (eng. I'm working today) (Slots reordering).

2. 每天他们洗澡↔他们每天洗澡 (eng. They take a bath every day) (Slots reordering).

However, provided solution suffers from a problem of manual definition of the frames. In most cases the problem is not severe, but it needs time and efforts from the person responsible for frame creation and maintaining. The variant of the solution is to provide a set of rules. For example, for the adverb applied to the entire sentence (see French examples) we can define a rule represented by the Formula (8).

$$< Adv > \circ S\backslash Adv \ \leftrightarrow S\backslash Adv \circ < Adv > \quad (8)$$

where $< \boldsymbol{Adv} >$ is the slot of the adverb type and $S\backslash Adv$ is the rest of slots of the frame without the slot $< \boldsymbol{Adv} >$. Operator $\circ$ here denotes the composition of slots.

To apply the rules, we need to mark the slots we want to be processed by the special markers making them typed slots. Thus, we will have two kinds of slots within the system: typed and untyped. The result of processing is the set of frames with reordered slots generated by the system according to the order defined by the rule.

To make the system flexible we can introduce two types of rules: public and private. Public rules are always applied by the system in all cases, excluding situations when the expert rejects the public rule for a frame set. Private rules should be selected and attached manually by the expert to the set of frames. Experts can define public rules for the language they are working with and private rules for the sentences.

The next question considers the rules of interpretation, e.g. according to the rule of indefinite article "a" becomes "an" before the word started with the vowel. It seems to be better to apply the generic set of rules of transformation for each language according to which generated sentences could be refined. Another variant is to attach the set of such rules to each frame. The most flexible solution is to use the combination of above two approaches: first apply generic rules set and then the specific one.

The next problem is connected to error handling. To add the notification mechanism, we need to add erroneous variants to the set of slot values, but instead of marking them with a weight we should mark them with "error" modifier adding the text of the error or in a more general case adding the reference to the explanation to avoid the doubling of the same text among different slots of different frames.

To understand the necessity of the feature let's see some examples. English example. The user used "investigate" term instead of "research" in the phrase "*we will be also researching the international market*". So, his answer was "*we will be also investigating the international market*". Without error handling feature the system returns that the variant is not right and the mark for that variant is equaled to zero. When we add error handling the situation will be processed more effectively and user will be informed about the error. It could be a sentence with error explanation with examples. For example: "to investigate *means to inquire into or study in order ascertain facts without methodology*. E.g. to *investigate the causes of natural phenomena, to investigate the foregoing issues*. To research *means to search or examine something with continuous care*.".

French example. The user used " decouvrir" term instead of "explore" in a phrase "*Explorer la jungle s'est avéré être un défi pour nous tous.*". In response he/she will get the following explanation: "*Explorer quelque chose, c'est traverser une région inconnue pour en apprendre davantage ou pour examiner quelque chose. Par exemple, Les biologistes étaient ravis d'explorer la nouvelle terre; elle a exploré les objets de près pour que rien ne passe inaperçu. Par contre, découvrir, c'est trouver quelque chose ou acquérir des connaissances ou prendre conscience de quelque chose. Par exemple, le radium a été découvert par Marie Curie; nous avons découvert un joli petit chalet à la périphérie de la ville. "*.

Thus, we can see that the system with proper error handling becomes more interactive, communication-oriented. It is notably, that even the right variants can be accompanied with commentaries letting the user understand why the score of his/her answer is not so high as it was expected.

And the last problem mentioned by the task definition is how to make the process of scenario description easier and convenient for the expert. It must, ideally, be the plain file without databases and editors. In other words, databases and editors can be regarded as extended version of the base one which based on the use of plain text file.

The structure of the scenario that we propose is shown in the following listing 1.

Listing 1

```
Victor [en] Ok, I'm here. [animation:placed emotion:glad place:window]
Victor [en] [animation:walking emotion:glad place:door animationSpecific:nowait]
Alex [en] [animation:placed emotion:thinking place:window1]
```


ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Victor [en] Hello, sir. Do you speak English? We are from New York Times. May I ask you some questions? [animation:talk emotion:glad]

Paul [fr] Bonjour. Je ne parle pas anglaise. Je ne parle que français. Quelles questions?

Alex [en] He doesn't speak English. Only French. What are the questions?

Victor [en] Well I want to talk about strikes and manifestations.

*T [fr] <*Q ><!N ><!V ><!VV >< !S>*

Q1= Alors{.1}

N1= je {.1}

V1= voulais {.2}

VV1= qu'on parle {.3}

*S1= de la grève <et *K> {.4}*

K1= des manifestations{.2}

The scenario consists of two parts: context and tasks. Each task relates to the frame set and suggests the user answer. The structure of the context sentence can be defined as a tuple <actor, language, phrase, modifiers block>. Sometimes, depending on the role, actor does not have any phrase to say but have modifiers block which defines his behavior within the system. For example, let's take a look at the line

Victor [en] [animation:walking emotion:glad place:door animationSpecific:nowait]

According to the modifiers the actor whose name is Victor should go to the door from his initial place with glad emotion, and he can start talking only after he takes his place. Details of what modifiers can be used in scenarios and how they are processed by the system are out of scopes of the article.

The step of the scenario of the Task type always starts with letter T (means Task) accompanied by the language identifier and a frame set described using the script language which we already know.

As we can see, the method provided is rather simple, based on the plain file. Of course, we can build editor and save this scenario into database, but the editor and the database cannot be regarded as an integral part of the scenario management system.

Summary. The work provides some variants of solutions of the problems appeared during the attempt to build an integrative system using the model represented in [8]. To make the system more adoptable to the expert needs we introduce some methods which would help making a set of frames with reordered slots more effectively: additional set of frames which can the share slots among each other and with

the basic one, public and private rules of frame transformation. To make error handling more convenient for end-user we propose to include erroneous variants of answer into the list of the slot values attaching detailed description of the error, the same method could be used for detailed description of the right variants with low weight. We also provide a mechanism of interpretation rules which could be used for refining the text built by the system using frames. In the end we provide a way to describe the scenario using plain text including task steps which based on frames. We hope that all the described aspects of the problems and their solutions will help developers to make integrative systems more interactive, communication-oriented and effective.

REFERENCES

1. Warschauer M. (1996) "Computer Assisted Language Learning: an Introduction". In Fotos S. (ed.) Multimedia language teaching, Tokyo: Logos International: 3-20.
2. Wan Y. An Integrative Approach to Teaching English as a Second Language: The Hong Kong Case. Conference on English Language Learning for the 21st Century (Hong Kong, January 1996). – 14 p.
3. Lightbown P. M., Spada, N. How languages are learned. Oxford: Oxford University Press. – 1998. – 135 p.
4. Lightbown P. M., Spada, N. How languages are learned. Oxford: Oxford University Press. 1998. 135 p.
5. Warschauer M. (1996) "Computer Assisted Language Learning: an Introduction". In Fotos S. (ed.) Multimedia language teaching, Tokyo: Logos International: 3-20.
6. Wang C. On Linguistic Environment for Foreign Language Acquisition. Asian culture and history. Vol.1 No.1. 2009, pp. 58-62.
7. Litvinov A.A. On formalization of computer-aided language learning system scenario. // System technologies. – N.1(108). – Dnepropetrovsk, 2017, pp. 63-70.
8. Litvinov A.A. On using of frame-based sentence patterns in integrative computer-aided language learning systems. // System technologies. – N.1(114). – Dnepropetrovsk, 2018, pp. 71-78.
9. M. Minsky 1975. ‘A Framework for Representing Knowledge’. In The Psychology of Computer Vision, ed. P. H. Winston, 211 – 277. New York: McGraw-Hill.

Received 19.01.2022.
Accepted 24.01.2022.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Практичні аспеки застосування фреймових шаблонів речень в нтегратив ній лінгвістичній системі

Аналіз останніх досліджень та публікацій. Існують три ключові класи систем CALL, пов'язані з трьома підходами до вивчення мови: біхевіористський, комунікативний, інте гративний [2, 3, 4]. Системи першого типу не зосереджені на принципі «тренуйся і практикуй» [1]. комунікативні системи намагаються залучити комп'ютерну систему, заохочуючи студентів вирішувати оригінальні завдання у середовищі, в якому використання цільової мови є природним. Інтегративні системи базуються на двох технологіч них компонентах – мультимедійних комп'ютерах та Інтернеті [1]. Основними особливо стями таких систем є: моделювання автентичного мовного середовища за допомогою засобів медіа контенту, що робить природним поєднання читання, письма, говоріння та аудіювання в одній діяльності; індивідуальний підхід до кожного учня; зосередження на змісті, який цікавить студентів. Типовою інтегративною системою є програма Dustin, розроблена Інститутом наук навчання Північно Західного університету [5, 6].

У роботі [7] представлено формальний опис сценарію інтегративної системи на вчання мови. Відповідно до [7] сценарій можна представити у вигляді набору кроків. Ключовим елементом кроку є повідомлення, яке передає сенс події користувачеві. Повідо млення складається з ряду проекцій повідомлення: тексту, формули, відеоролика, аудіо тощо. Існує два типи повідомлень: повідомлення, орієнтовані на завдання, які виклика ють переривання потоку, що вимагає реакція користувачів і контекстно орієнтовані повідомлення. Контекстно орієнтовані повідомлення утворюють фон для природного залучення користувача до спілкування на основі повідомлень, орієнтованих на завдання.

Зробити взаємозв'язок між користувачем і системою більш природним і ефектив ним означає зробити систему гнучкою для різноманітних відповідей учнів. Очевидно, що цю проблему неможливо вирішити просто перерахуванням усіх доступних варіантів. Та ким чином, слід застосувати корисний механізм шаблонів, здатний допомогти вчителю описати величезну кількість варіантів за допомогою компактного конструкцій. Розв'я занню цієї проблеми була присвячена стаття [8]. Рішення базується на парадигмі на ос нові фрейму [9], згідно з якою кожне речення може бути представлено у вигляді ряду впорядкованих слотів (терміналів і нетерміналів), пов'язаних із термінами (варіантами частин виразу, представленими слотами). Деякі слоти настійно повинні бути заповнені значеннями, інші є необов'язковими. Колекція необхідних слотів утворює семантичне яд ро виразу: значення не може бути передано, якщо слот не заповнений належним значен ням. Слоти можуть бути пов'язані з низкою альтернативних значень – термінів з одна ковим значенням, деякі з яких можна вважати кращими. Деякі з цих значень також можна описати складними структурами (фреймами), але здебільшого вони еквівалентні слову або кільком слів. Порядок розташування слотів дозволяє нам будувати синтаксично пра вильні речення.

Мова, яку ми використовуємо для опису фреймів, нагадує шаблони IC (безпосередніх складових), покращених новими семантичними позначками. Опис здається простішим і зрозумілішим, ніж варіант чисто фреймового мовного опису. «<...>» – визначає межі слота. Зірочка («») , розміщена перед позначеною віссю (назва термінального слоту, наприклад, Q, T, Tt, S), означає, що слот є необов'язковим, «!» перед тим, як вісь оголошує, що слот необхідний.*

Розглянемо приклад.

*<*Q ><*T ><!S >*

Q1=as you're aware {.1}

Q2=as you know {.2}

*T1=today <*Tt> {.3}*

Tt1=in the evening {.1}

S1=we're going to be looking at your career options {.5}

Такий фрейм створює наступний набір речень (десяткові числа в квадратних дуж ках позначають значення сприйняття фрази):

We're going to be looking at your career options [.5]

Today we're going to be looking at your career options [.8]

Today in the evening we're going to be looking at your career options [.9]

As you know we're going to be looking at your career options [.7]

As you're aware we're going to be looking at your career options [.6]

As you know today we're going to be looking at your career options [1.0]

As you're aware today we're going to be looking at your career options [.9]

As you know today in the evening we're going to be looking at your career options [1.1]

As you're aware today in the evening we're going to be looking at your career options [1.0]

Визначення задачі. Приваблива ідея використання фреймів страждає від наступних недоліків. По перше, не буде помилкою, якщо ми скажемо, що кожна мова дозволяє зміню вати порядок слів у реченні (навіть обмеження пропозиції до 7 10 слів). Питання полягає в тому, як обробити такий тип допустимих варіантів. Наступне питання: загальні пра вила трансформації, напр. за правилом неозначеного артикля «а» стає «ап» перед сло вом, що починається з голосного. Третє питання, пов'язане з обробкою помилок. Коли ко ристувач вибирає неправильне слово або частину фрази замість правильного, він/вона отримує повідомлення, що його відповідь неправильна і, в найкращому варіанті, як ви глядає найбільш закрита правильна відповідь. Але зазвичай користувачі хочуть отри мати пояснення, чому наданий варіант неправильний. Як обробити ситуацію, роблячи систему більш зручною та дружньою, підвищуючи її орієнтацію на взаємодію. Останнє питання – як описати шаблони в сценаріях, як зробити процес побудови сценарію прос тішим та ефективнішим. Саме цим питанням і присвячена дана робота.

Головна частина. Перша проблема, згадана вище, пов'язана зі здатністю експерта без значних зусиль описувати допустимі комбінації слотів з урахуванням перетворення слів і зміни порядку. Загалом ми стикаємося з ситуацією, коли одне речення має бути опи сано кількома фреймами. Ці фрейми можна підключити до одного і того ж набору слотів, і в цьому випадку ми отримаємо клони однієї і тієї ж структури, що відрізняються за композицією слотів. Але це рішення не охоплює коло ситуацій, пов'язаних із проблемою. Більш правильним рішенням було б припустити, що деякі з цих фреймів можуть мати власні набори значень слотів і навіть вводити нові слоти.

Однак надане рішення страждає від проблеми ручного визначення фреймів. У біль шості випадків проблема не є серйозною, але потребує часу та зусиль від особи, відпові дальної за створення та обслуговування каркаса. Варіант розв'язання — надати набір правил.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Щоб застосувати правила, нам потрібно позначити слоти, які ми хочемо обробити, спеціальними маркерами, щоб зробити їх набраними слотами. Таким чином, ми матимемо два види слотів у системі: типізовані та нетиповані. Результатом обробки є набір фреймів із переупорядкованими слотами, згенерованими системою відповідно до порядку, визначеного правилом.

Щоб зробити систему гнучкою, ми можемо ввести два типи правил: публічні та приватні. Загальнодоступні правила завжди застосовуються системою у всіх випадках, за винятком ситуацій, коли експерт відхиляє публічне правило для набору фреймів. Приватні правила повинні бути обрані та прикріплені вручну експертом до набору фреймів. Експерти можуть визначити публічні правила для мови, з якою вони працюють, і приватні правила для речень.

Наступне питання розглядає правила трансформації. Здається, краще застосувати загальний набір правил трансформації для кожної мови, відповідно до яких створені речення можна було б уточнювати. Інший варіант – прикріпити набір таких правил до кожного кадру. Найбільш гнучким рішенням є використання комбінації двох вищезазначених підходів: спочатку застосувати загальний набір правил, а потім специфічний.

Наступна проблема пов'язана з обробкою помилок. Щоб додати механізм сповіщень, нам потрібно додати помилкові варіанти до набору значень слотів, але замість того, щоб позначати їх вагою, ми повинні позначити їх модифікатором «error», додаючи текст помилки або в більш загальному випадку додаючи посилання на пояснення, щоб уникнути подвоєння одного і того ж тексту між різними слотами різних кадрів.

Остання проблема, про яку йдеться у визначенні завдання, полягає в тому, як зробити процес опису сценарію легшим і зручним для експерта. В ідеалі це має бути простий файл без баз даних і редакторів.

Структура сценарію, яку ми пропонуємо, показана в наступному лістингу 1.

Лістинг 1.

```
Victor [en] Ok, I'm here. [animation:placed emotion:glad place:window]
Victor [en] [animation:walking emotion:glad place:door animationSpecif
ic:nowait]
Alex [en] [animation:placed emotion:thinking place:window1]
Victor [en] Hello, sir. Do you speak English? We are from New York Times. May I
ask you some questions? [animation:talk emotion:glad]
Paul [fr] Bonjour. Je ne parle pas anglaise. Je ne parle que franзais. Quelles
questions?
Alex [en] He doesn't speak English. Only French. What are the questions?
Victor [en] Well I want to talk about strikes and manifestations.
T [fr] <*Q ><!N ><!V ><!VV >< !S>
Q1= Alors{.1}
N1= je {.1}
V1= voulais {.2}
VV1= qu'on parle {.3}
S1= de la gruve <et *K> {.4}
K1= des manifestations{.2}
```

Сценарій складається з двох частин: контексту та завдань. Кожне завдання відно ситься до набору кадрів і пропонує відповідь користувача. Структуру контекстного ре чення можна визначити як кортеж <актор, мова, фраза, блок модифікаторів>. Іноді, за лежно від ролі, актор не може сказати жодної фрази, але має блок модифікаторів, який визначає його поведінку в системі. Крок сценарію типу Task завжди починається з літери T (означає завдання), що супроводжується ідентифікатором мови та набором кадрів, описаним за допомогою мови сценаріїв, яку ми вже знаємо.

Як бачимо, запропонований метод досить простий, заснований на звичайному файлі.

Резюме. У роботі наведено деякі варіанти розв'язання задач, що виникли під час спроби побудови інтегративної системи за допомогою моделі, представленої в [8]. Щоб зробити систему більш пристосованою до потреб експертів, ми вводимо деякі методи, які допоможуть зробити набір фреймів із переупорядкованими слотами більш ефектив но: додатковий набір фреймів, які можуть ділити слоти між собою та з основними, публічними та приватними правилами трансформації кадру. Щоб зробити обробку помилок більш зручною для кінцевого користувача, ми пропонуємо включати помилкові варіанти відповіді до списку значень слотів із докладним описом помилки. Ми також надаємо механізм правил інтерпретації, який можна використовувати для уточнення тексту, створеного системою за допомогою фреймів. Наприкінці ми надаємо спосіб описати сценарій за допомогою простого тексту, включаючи кроки завдання, які базу ються на фреймах. Сподіваємося, що всі описані аспекти проблем та їх вирішення допо можуть розробникам зробити інтегративні системи більш інтерактивними, комуніка ційними та ефективними.

Литвинов Олександр Анатолійович - кандидат технічних наук, доцент кафедри електронних обчислювальних машин Дніпропетровського національного університету ім. О. Гончара.

Lytvynov Olekcandr Anatoliyovich - candidate of technical sciences, associate professor, associate professor of the department of Electronic Computing Machinery, Oles Honchar Dnipro National University.

Литвинов Михайло Олександрович – студент, Дніпропетровський національний університет ім. О. Гончара.

Lytvynov Mihailo Oleksandrovich - student of the group KI-21m-1 of the faculty of Physics, Electronics and Computer Systems, Oles Honchar Dnipro National University.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)