UDC 519.876

N.A. Bazhenov
National Technical University “Kharkov Polytechnical Institute”, Kharkov, Ukraine
nabazhenov@gmail.com

Combining probabilistic tagging with rule-based multilevel chunking for requirements elicitation

In this paper author describes a multi-layered NLP approach for the elicitation of ontology relevant information from free requirements text. To automate the requirements elicitation process from textual information of stakeholders, as well as to transform it into structured and validated fashion the combination of probabilistic and rule-based NLP methods are proposed. The developed methodology includes a multi-level chunking strategy as its core principle.

Introduction

Within the software development process the problem of requirements elicitation is one of the main complicated questions, which needs to be solved in unconventional case. Obviously the requirements information quite often exists in a not explicit and ambiguous textual format. In this case the elicitation of domain specific concepts poses many difficulties to the requirements engineer. The software development process implies the transform requirements from text specifications to the special kinds of intermediate predesign models [1], [2]. For solving this task the different NLP (Natural Language Processing) techniques can be proposed [3].

The goal of the work was to develop an approach to process a free requirements text, to elicit the relevant requirements-driven information, and to transform it into structured, available for system end-user validation and interaction view. To succeed in this goal **the following tasks were stated**:

- tokenization texts;
- allocation of possible linguistic characteristics;
- lemmatization;
- sentence limits detection;
- reduction of the characteristics of categories and disambiguation;
- chunking;
- implementation heuristic rules as e.g. conversion of categories;
- output in XML format.

Hence a NLP system for supporting the software designer to make implicit textual information easier to trace was developed. The methodology is a combining probabilistic and rule-based parser, which processes the free text using morphosyntactic, sentence-semantic and phrasal information and enriched in XML format available.

The proposed methodology includes algorithms for tagset mapping, pre-chunking and multi-level chunking of free English requirements text, the main layers are:

1. The tagging task carried forward to QTAG, a probabilistic tagger written in Java by O. Mason [4].
2. The mapping engine we developed for splitting up and reinterpreting the standard QTAG-Set.

3. The identification of compound nouns. We suppose that implicitness is very often motivated through ambiguity of complex terms, e.g. unclear structure of compounds or other groups of words.

4. The extraction and generation of inflectional word forms.

5. Some other morphological information extraction.

6. Multi-words units and idiomatic expression identification.

7. Verb subclass identification.

8. Some chunking heuristics needed for grouping words to morphological units and syntactical chunks, which we chose as candidates for conceptualization nodes in the ontology layer.

Related work

For tagging English free texts many open source systems like the decision based "Treetagger" [5], the rule- and transformation-based "Brill tagger" [6], the maximum-entropy "Stanford POS Tagger" [7], the trigram based probabilistic "QTAG" [8] etc. are available. We chose "QTAG" because it is an extendable, trainable, language independent tagger.

There are also approaches including components for chunking, e.g. "MontyLingua" [9], "MontyKlu" (an online-version of "MontyLingua" developed by members of the research group in Klagenfurt [10]) and the "NLTK Toolkit" [11]. These systems mainly provide standardized and acceptable output, but as we know they have been developed for educational purposes only. According to ontology engineering needs they are not really useful.

General procedure

In our NLP system (fig. 1) the data flow from the input state through processing blocks into the output state. The input is a raw text in ".doc" or ".txt" format, the existing free APIs, the QTokenizer [4] & QTAG, are used for producing tagged text. Between the blocks Q-Tag and QTokenizer there is the block "Tokenize Correction", which corrects some well-known tokenization problems, such as possessive case of nouns (e.g. Peter's), double adjectives, numbers with points, etc.

For the transformation to the extended tagging format we use a mapping engine which is based on a set of mapping rules.

Then after all possible information from Q-Tag output is extracted we need to get extra tagging information. The next processing step we call the pre-chunking procedure. It includes the correction of a possibly wrong QTAG output, it solves some simple ambiguity problems and it identifies certain verb subclasses according to those heuristic rules, which are based on internal heuristics. These procedures make the natural language material ready for chunking. Also on this stage our system makes lemmatization, corelex nouns identification. In the case of wrong tagged result one can create additional lexicons with needed entries of tags and its frequencies. These lexicons can be used as "before-consulted" resources for tagger and thus, text can be retagged for corrected result obtaining.

The chunking procedure consists of multi-words units and idiomatic expressions detection, default chunking rules engine and elements of fine-granulated chunking methods. The chunking rules engine operates step by step depending on their level numbers and the chunk trees are built up respectively.

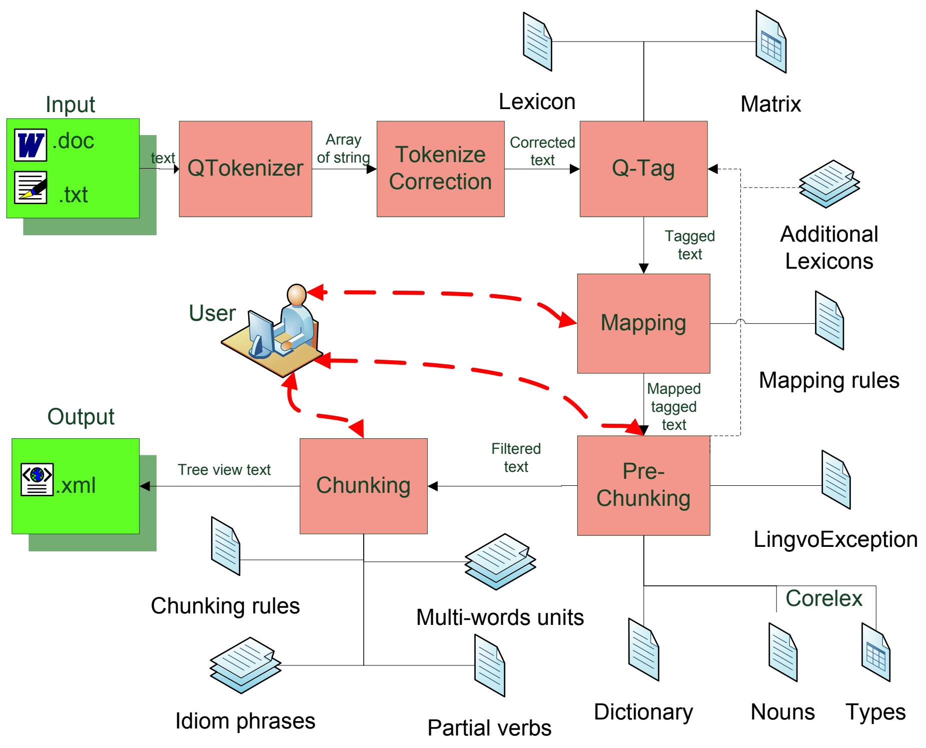

Figure 1 – Conceptual scheme of our NLP system

As a result of this approach we receive an extended POS (part-of-speech) tagged text in tree view with the root node <text>. We store the output in the XML format, which allows us to review it with any XML editor and of course to interpret it in the next stage of the workflow. Note that in principle all processing blocks work automatically in the default configuration, but the user can change the settings also manually. These possibilities are imaged with dashed arrows.

Details of linguistic processing

The mapping step. We use QTAG as the primary parsing and tagging engine. The output is needed for further processing. But as we pointed out the QTAG output had to be adopted for requirements engineering workflow needs since both systems have different view structures. So, we propose, that:

1. QTAG carries out the basic processing step.
2. We extract relevant information from the QTAG output and transform it into the enriched tagset format.
3. We have to use some additional methods and heuristics to elicit semantic information needed during the further processing steps of the requirements engineering workflow.

We propose that mapping from the shallow, standardized QTAG-Set [4] to the ontology-oriented tagset (developed in project NIBA [1], [12]), which consists of basic main POS-categories with arrays of attributes (e.g. v0 with subclass attribute “tvag2”(=monotransitive verb with agentive subject), is necessary for the identification of ontological key relations and terms. Figure 2 shows how part-of-speech tags are extracted from the QTAG output and reassigned using the NIBA tagset notation (e.g. v0, n0, a0, etc.).

Additional information about concrete part-of-speech instances is presented by using fine-granulated attributes. As an example, the verb “is” in QTAG gets the tag <BEZ>. This tag decodes, that “is” is an auxiliary verb with the inherent morphosyntactic values present tense, singular, third person and having the “be” as the base form. After mapping and restructuring we receive the main class tag <v0> and an explicit set of attributes belonging to this tag: verb-class=”aux”, temp=”pres”, form=”ind”, num=”sg” ps=”3”, baseform=”be”.

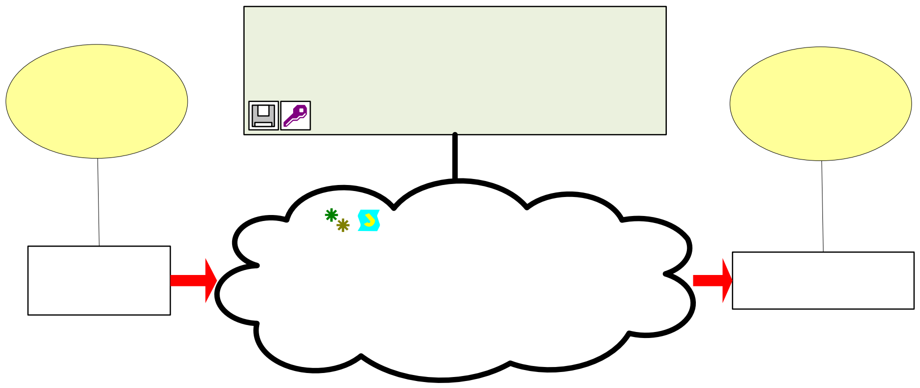

Figure 2 – Mapping process

Pre-Chunking. For the next step our aims are more sophisticated. We propose needing information about verb subclasses [12], verbal lemmata, the base forms of nouns and about some other inflectional parameters like passive voice construction. For solving this task we used some heuristics based on basic English grammar rules.

Disambiguation of auxiliary verbs. Quite often auxiliary verbs (like “be” or “have”) function as main verbs.

As a result of the mapping procedure our system assigns the value “aux” to all inflectional forms of “to be”, “to do”, “to have” and the modal verbs in the default case. In consequence we need to eventually reinterpret the subclass value “aux” as “copV/possV” (table 1) depending on the concrete syntactic context.

For solving this problem the following heuristic procedure is chosen. The filtering engine goes through all tagged words, identifies the auxiliary verbs and then checks their contextual conditions.

The following formal definitions are relevant for this purpose:

S – set of all sentences, $S = S\{S_j \in S, j = \overline{1,n}\}$;

W – set of all output words in the same way;

T – set of all output tags;

V^{aux} – set of enriched output tags, which are agreed with auxiliary modeless verbs, $V^{aux} = \{v0_j : v0_j.\text{verbclass}=\text{"aux"} \wedge v0_j.\text{form} \neq \text{"modal"}, v0_j \in V\}$, V – set of all verb tags.

Thus, the rule can be written as (for $i = \overline{1, N_{V^{aux}}}$):

$$\forall\, v0_i : v0_i \in V^{aux},\ v0_i \in S_k;\ \nexists\, v0_j : v0_j \in S_k,\ v0_j.\text{form} \neq \text{"modal"} \rightarrow v0_i.\text{verbclass} = \text{copV}, \quad (1)$$

$$\forall\, v0_i : v0_i \in V^{aux},\ v0_i \in S_k,\ v0_i.\text{baseform} = \text{"have"};\ \nexists\, v0_j : v0_j \in S_k,\ v0_j.\text{form} \neq \text{"modal"} \rightarrow v0_i.\text{verbclass} = \text{possV}. \quad (2)$$

Verb transitivity disambiguation. According to the requirements engineering purposes we also need to identify other verb subclasses (Table 1).

Table 1 – Verb subclasses

№	Abbreviation	Description
1	aux	Auxiliary verb
2	eV	Ergative verb
3	iV	Intransitive verb
4	lokV	Locations verb
5	possV	Possessive verb
6	psychV	Mental verb
7	tvag2	Monotransitive verb with agent subject
8	tv3	Ditransitive verb
9	sentV	Perception verb
10	copV	Copula verb
11	tv2	Monotransitive verb without agent subject

Due to the fixed and transparent subject-verb-object (SVO) structure of English, normally verb transitivity identification is a quite simple and straightforward task. Nevertheless we have to take into account that the phrasal structure very often inhibits simple solutions like for example counting of nouns.

In our approach we used the following algorithm to cope with the problem of phrasal complexity:

1. Create a set of rules which can operate on simple singular term subjects and objects (e.g. proper nouns and personal pronouns).
2. Consult the exception database with already-assigned verbal subclass tags using training sentences which include default argument patterns.
3. Reconstruct the structure of the primarily assigned phrases if relevant morphosyntactic features don't fit.
4. Leave open the possibility to manually change wrong/exceptional assignments or to add new information about verb classes.

Presupposing the above defined default heuristic rules we add the following definitions for verb transitivity identification explained in (1) and (2) above:

$$\forall v0_i \in V, v0_i \notin V^{aux}, v0_i \in S_k; \exists t_{i-r} = n0.type = "proper" \mid pron0.type = "pers" \Rightarrow$$

$$\exists w_{i+r} = "in" \mid "on", w_{i+r} \in S_k \rightarrow v0_i.verbclass = "lokV", \tag{3}$$

$$\exists t_{i+r} = nPO, t_{i+r} \in S_k \rightarrow v0_i.verbclass = "tvag2", \tag{4}$$

$$\exists t^1_{i+r} = nPO \cup t^2_{i+r} = nPO; w^1 \neq w^2; t^1, t^2 \in S_k \rightarrow v0_i.verbclass = "tv3", \tag{5}$$

Note that nPO (noun pseudo-object) decodes word groups, which can include more than one noun. We identify those groups as noun compounds which can be taken as pseudo-objects for transitivity disambiguation. In the following example:

I write a system requirements document

the sequence of three nouns in the right context of the verb is interpreted as one argument and in consequence as a compound noun. That's why we classify "write" as a monotransitive verb. This process we call pseudo-object identification, because it doesn't presuppose any syntactic analysis.

In the next example sentence:

He gave Peter balls

we identified two nouns in object position representing two different objects because of the missing congruence between these nouns with respect to proper/common opposition. Thus, we classify “give” as a ditransitive verb.

Passive voice identification. For ontological design needs we have to detect passive constructions.

In the default case passive constructions are built with the auxiliary verb “to be” and the participle II of a main verb.

So, presupposing the definitions above, we define the following heuristic rule (for $i = \overline{1, N_V}$):

$$\forall v0_i : v0_i \in V, v0_i \in S_k, v0_i.baseform = "be"; \exists v0_{i+1} : v0_{i+1} \in S_k, v0_{i+1}.temp = "perf"$$
$$\rightarrow v0_i.\mathrm{mod}\,e = "pass", v0_{i+1}.\mathrm{mod}\,e = "pass". \qquad (6)$$

During the development of this approach the use of the additional trigger for passive identification was concluded. This one takes into account the reduced form, so-called participial construction:

$$\forall v0_i : v0_i \in V, v0_i.temp = "perf", \exists w_{i+1} \in \mathrm{Pr}\,epPassList \rightarrow v0_i.\mathrm{mod}\,e = "pass". \qquad (7)$$

This rule is used for the reduced form identification and only in the case if the current verb is used with one of the predefined preposition. For example:

Parents or legal guardians are filling in an application **provided by** *the day nursery.*

Verbal base form generation. During the pre-chunking procedure we relate the inflectional variants of a verb to the base form through synchronization of the main class tags and diversification of attributes which are used to identify the verbal variants (e.g. v0 is assigned to “see”, “saw”, “seen”; type: perception verb (“sentV”) is assigned to all three forms, “past” is assigned to “saw”, “perf” is assigned to “seen”).

For this purpose the procedure of base form generation was developed (fig. 3). Verb forms are extracted from the dynamically-extendable dictionary with the following verbal entry structure: <base-form>=<related form1> <related form2>…. If a certain word is found in the dictionary, the engine simply returns all needed information. If the searched word is not in dictionary, one of two possible strategies is executed. In one case default endings are assigned depending on the paradigm of regular verb forms. In the other case the user is offered the possibility to interact with the system. He can define base forms and/or inflected forms himself (fig. 3).

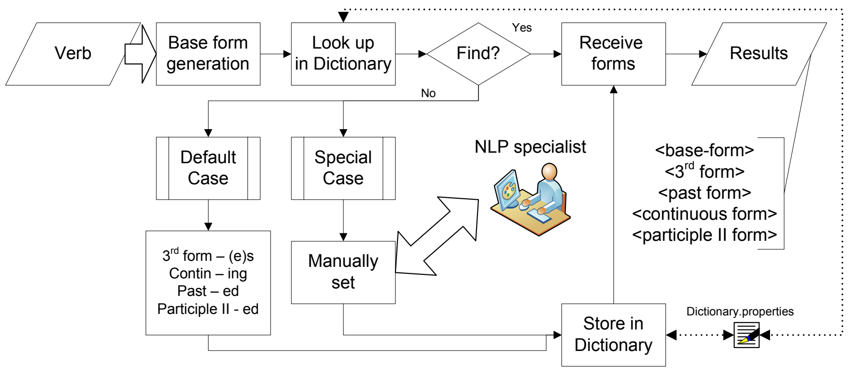

Figure 3 – Verbal base form generation

The common chunking rules. Based on some variants of the X-bar Theory [13] and on some core definitions in the project NIBA [1], [12] we composed a set of chunking rules for English for the production of syntactically and morphosynctactically motivated chunks (Table 2).

Table 2 – Excerpt from chunking rules

Rule (Summands → Result)	Rule level	Rule descriptions	Examples
n0+n0 → n0	1	Compound Noun	blood pressure
[pt0]+a0 → a2	1	Adjective Phrase	very nice
[a0]+a0 → a2	1	Adjective Phrase	bright green
[pt0]+q0 → q2	1	Quantor Phrase	very many
[q0]+q0 → q2	1	Quantor Phrase	one million
[pt0]+adv0 → adv2	1	Adverb Phrase	very often
[adv0]+adv0 → adv2	1	Adverb Phrase	yesterday noon
pron0(type=pers) → n3	1	Noun Phrase	she
v0(verbclass=aux)+[adv0]+v0 →v0(type=complex)	1	Complex Verb	will certainly go
v0(verbclass=aux)+pt0(type=neg)+v0 →v0(type=complex)	1	Complex Verb	would not write
v0+pt0(type=verbal) →v0(type=complex)	1	Complex Verb	wake up
q2+q2 → q2	2	Quantor Phrase	two hundred million
pron0(type=poss)+n0 → n3	2	Noun Phrase	his mother
[det0]+[a2]+[q2]+n0 → n3	3	Noun Phrase	the nice two girls
[det0]+[q2]+[a2]+n0 → n3	3	Noun Phrase	the three busy scientists
p0+n3 → p2	4	Prepositional Phrase	of blood pressure measurement

There are several types of chunking rules, which are arranged in a certain order that should be followed during the chunking process. Summands are the array of input nodes which are needed for building the next resulting upper node of the chunking tree. Some of summands are strictly required for rule producing, they are written without square brackets, but some are not obligatory, they are placed inside brackets.

The chunking process starts on the first level of the rule system, e.g. with the identification of compound nouns and complex verbs, and climbs up the rule hierarchy, ending up at the fourth level of the rule system with the identification of prepositional phrases. The step of constructing compound nouns is a very important precondition for ontology elicitation, because compounds normally serve as specialization terms in domain ontologies. In general we think that the number of simple nouns involved in compound correlates with the degree of specialization.

Multi-words units. Multi-word units are phrases that function grammatically as single words, e.g. conjunction *so that* or preposition *in spite of*, *receive* a single POS tag, so they are treated here as single words.

Before the work of the common chunking rules engine the phase of multi-word unit's assignment is run. Since these collocations have only grammatical function we don't combine them into the new node of more high level, but only assign to the parts of these units the relevant attributes. Algorithm of this procedure is the same as for the chunking rules with the exception of that we don't create new node here. One of the particular cases of multi-word units is partial (or phrasal) verbs.

Idiomatic expressions identification. Building up the chunking tree begins with predefined node phrases lists comparison. In English there are a lot of stable set expressions, e.g. idiomatic expression. That's why for a good phrase-based parsing it is very important use these expressions for identifying POS-phrase on a much higher level. For this purpose we have extendable lists for different POS nodes and compare the entries from these lists with our sentence constructs. And if one would find this construct, the new node on the next level would be created.

Fine-granulated chunking rules. After that all default simple chunking rules and predefined patterns of idiomatic expressions are applied, the chunked tree of raw text is not complete. English as each modern language has a lot of complex internal structures, whose identification is a very important and implicit thing in NLP today. For the requirements elicitation sense it is also very useful to extract some of that things, turns of speech and others. These tasks can be implemented with using of some predefined heuristics based on the common rules of English usage. So, consider the things, which are relevant for this work, and methods which help to implement them:

Preposition phrase as the post-modifier of the noun phrase. In our approach the following heuristic method is proposed (8):

$$\begin{gathered}\forall s_i : s_i \in S^{cur},\ s_i = p2, s_i.head \in N3P2List, \\ \exists s_{i-1} : s_{i-1} \in S^{cur},\ s_{i-1} = n3 \rightarrow s_{i-1}.children_{last} = s_i, \\ \exists s_{i-1} : s_{i-1} \in S^{cur},\ s_{i-1} = p2 \rightarrow s_{i-1}.children_{last}.children_{last} = s_i\end{gathered} \tag{8}$$

Defining this rule the notations from previous definitions are used and some additional:

– $s_i.head$ – the children of the node s_i, which means the head of relevant phrase. In this case it is the preposition;

– $N3P2List$ – the list of prepositions, which signal about the post-modifier case. It consists of the following: “of”, “for”, “with”, “to”, “in”, “as”;

– $children_{last}$ – the element of the array “children” with the highest index.

Note that the identification of post-modifier prepositional phrases should be realized as LIFO process, the searching engine must go through the sentence form right to left.

Parenthesis is an explanatory or qualifying word, clause, or sentence inserted into a passage with which it doesn’t necessarily have any grammatical connection, and from which it is usually marked off by round or square brackets, dashes, or commas. In the capacity of parenthesis could be the noun groups or the whole sentence construct. To fix the idea let confine ourselves only to use brackets for parenthesis identification:

$$\begin{gathered}\forall\{s_i\} : \{s_i\} \in S^{cur}, i = \overline{l,m},\quad s_{l-1} = '(', s_{m+1} = ')' \\ \exists s_k \in \{s_i\} : s_k = v0 \rightarrow s_{l-1} = sentence, s_{l-1}.type = parenthesis, s_{l-1}.children = '(' \cup \{s_i\} \cup ')', \\ \exists s_k \notin \{s_i\} : s_k = v0 \rightarrow s_{l-1} = n3, s_{l-1}.type = parenthesis, s_{l-1}.children = '(' \cup \{s_i\} \cup ')' \\ \Rightarrow s_{l-2}.children_{last} = s_{l-1}\end{gathered} \tag{9}$$

Homogeneous parts are parts of the same category standing in the same relation to other parts of the sentence a “contracted” sentences:

$$\begin{gathered}\forall\{s_i\} : \{s_i\} \in S^{cur}, i = \overline{k,r}, \\ \exists s_j : s_j = ',' \vee s_j = con0, s_j.type = coord, j = i(i \bmod 2 = 0) \cap, \\ \exists s_l : s_l = s_{l+1}, l = i(i \bmod 2 \neq 0) \rightarrow s_i = s_l, s_i.children = \{s_i\}, s_i.type = homogeneous\end{gathered} \tag{10}$$

Descriptions of relative clause, infinitive groups, subordinate sentences and post-chunked verb subclasses assignment verification are omitted here for brevity.

Conclusion and further work

NLP driven ontology engineering is certainly one of the key technologies in the requirements engineering realm of the upcoming decade [14-16]. The proposed approach, based on the combining probabilistic POS-tagging with multilevel chunking, allows to proceed from free requirements texts into the special ontology-oriented linguistic model enriched with the set of elicited attributes, relations and different characteristics. Parts of our approach are based on the algorithms which we described in this paper. The involved procedures are heuristically founded and follow a multi-level chunking strategy.

This extended ontology-oriented linguistic model can be used further to generate the conceptual predesign requirements model [2] and fill its glossaries with concepts and model elements. For this actions the interpretations transformations should be used.

Literature

1. Fliedl G. NIBA Project: Overview / G. Fliedl, Ch. Kop, J. Vöhringer, Ch. Winkler // ER 2005: 24th International Conference on Conceptual Modeling, Alpen-Adria-Universität Klagenfurt, 2005.
2. Shekhovtsov V.A. Capturing the Semantics of Quality Requirements into an Intermediate Predesign Model / V.A. Shekhovtsov, Christian Kop, Heinrich C. Mayr // SIGSAND-EUROPE Symposium 2008. – Marburg, Germany, 2008. – P. 25-38.
3. Manning C.D. Foundations of statistical natural language processing / C. Manning. – The MIT Press, 1999. – 680 p.
4. Oliver Mason's. Webpages [Электронный ресурс] / O. Mason. – Режим доступа : http://www.english.bham.ac.uk/staff/omason/index.html.
5. Schmid H. Probabilistic part-of-speech tagging using decision trees / H. Schmid // Proceedings of the International Conference on New Methods. – In Language Processing. – 1994.
6. Brill E. Unsupervised learning of disambiguation rules for part of speech tagging / E. Brill // Proceedings of the 3rd Workshop on Very Large Corpora. – 1995. – P. 1-13.
7. Goldwater S. A fully Bayesian approach to unsupervised part-of-speech tagging / S. Goldwater, T. Griffiths // Proceedings of the 45th Annual Meeting of the ACL. – Prague, 2007. – P. 744-751.
8. Mason O. Programming for Corpus Programming: how to do text analysis with Java / O. Mason. – Edinburgh University Press, 2000. – 224 p.
9. Montylingua : a free, commonsense-enriched natural language understander [Электронный ресурс] – Режим доступа : http://web.media.mit.edu/~hugo/montylingua.
10. Monty Klu Web v0.1 [Электронный ресурс]. – Режим доступа : http://montyklu.knospi.com.
11. Bird S. Natural Language Processing in Python [Электронный ресурс] / S. Bird, E. Klein, E. Loper. – Режим доступа : http://nltk.org/doc/en/book.pdf.
12. Weber G. NIBA<tag> Aspekte der Implementierung eines erweiterten Taggers für die automatische Textannotation in NIBA. Master Thesis / G. Weber. – Alpen-Adria-Universität Klagenfurt, 2007. – 114 p.
13. Black C.A. A step-by-step introduction to the Government and Binding theory of a syntax [Электронный ресурс] / C.A. Black. – Режим доступа : http://www.sil.org/americas/mexico/ling/E002-IntroGB.pdf.
14. Межуев В.И. Использование онтологий как предметных областей / В.И. Межуев // Искусственный интеллект. – 2009. – № 4. – С. 4-11.
15. Звенигородский А.С. Концепция и задачи понимания смысла текста в системах искусственного интеллекта / А.С. Звенигородский // Искусственный интеллект. – 2009. – № 3. – С. 6-10.
16. Шевченко О.Ю. Метод побудови інтелектуальних систем обробки інформації та документообігу за допомогою онтологічної бази знань / О.Ю. Шевченко, М.В. Климова // Искусственный интеллект. – 2009. – № 2. – С. 91-97.

Н.А. Баженов

Сочетание вероятностного тегирования с основанным на правилах многоуровневым синтаксическим анализом для извлечения требований

Статья посвящена описанию NLP подхода к извлечению соответствующей онтологической информации из необработанных текстов требований. Для автоматизации процесса извлечения требований из текстовой информации заинтересованных лиц, а также для её трансформации в структурированную и пригодную для валидации форму предлагается использовать сочетание вероятностных и основанных на правилах методов NLP. Разработанная методология в качестве основного принципа включает в себя многоуровневую стратегию синтаксического анализа.

М.О. Баженов

Поєднання імовірнісного тегування із заснованим на правилах багаторівневим синтаксичним аналізом для витягу вимог

Стаття присвячена опису NLP підходу до витягу відповідної онтологічної інформації з неопрацьованих текстів вимог. Для автоматизації процесу витягу вимог з текстової інформації зацікавлених осіб, а також для її трансформації в структуровану і придатну для валідації форму пропонується використовувати поєднання імовірнісних та заснованих на правилах методів NLP. Розроблена методологія у якості основного принципу включає в себе багаторівневу стратегію синтаксичного аналізу.

Статья поступила в редакцию 01.04.2010.