Expert Systems with Applications 41 (2014) 5308–5322

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Formalisation method for the text expressed knowledge

Alen Jakupović [a,*], Mile Pavlić [b,1], Zdravko Dovedan Han [c,2]

[a] Business Department, Polytechnic of Rijeka, Vukovarska 58, 51000 Rijeka, Croatia
[b] Department of Informatics, University of Rijeka, Radmile Matejčić 2, 51000 Rijeka, Croatia
[c] Department of Information and Communication Sciences, Faculty of Humanities and Social Sciences, University of Zagreb, Ivana Lučića 3, 10000 Zagreb, Croatia

ARTICLE INFO

Keywords:
Knowledge representation method
Semantic relationship
Phrase structure grammar
Word
Phrase
Wh-question

ABSTRACT

This paper describes a method of knowledge representation as a set of text expressed statements. The method is based on the identification of word-categories/phrases and their semantic relationships within the observed statement. Furthermore, the identification of semantic relationships between words/phrases using wh-questions that clarify the role of the word/phrase in the relationship is described. A conceptual model of the computer system based on the formalization method of text-expressed knowledge is proposed. The subsystem text formalization is described in detail, especially its parts: syntactic analysis of the sentence, sentence formalization, phrase structure grammar and lexicon. The phrase structure grammar is formed by induction and it is used to generate the language of the formalized notation of a sentence. The derivation of grammar is based on the simple phrase structure grammar which was used for the syntactical analysis of informal language notation. In its base, the suggested method translates sentences of the informal language into formal language sentences which are generated by the derivated phrase structure grammar. Current limitations of the method that also set the path of its further development are shown. Next concrete steps in the development of the method are also described.

1. Introduction

The standard definition of artificial intelligence (AI) defines it as a scientific field that studies how to accomplish intelligent behaviour by computational means. In other words, AI studies computational modelling of human intelligence. Intelligent behaviour is conditioned by knowledge – human act based on what they know (or believe) about the world (Brachman & Levesque, 2004; Jordan & Russell, 2001). One of the fields of interest of AI is knowledge representation and reasoning, which is suitable for reaching artificially intelligent behaviour.

Knowledge is distributed in different manners among human beings (e.g. text, figure, sound or anything else that can carry knowledge) (Gottschalk-Mazouz, 2007). During the process of learning, it becomes personalized, embedded and stored in the human mind. Various cognitive functions (e.g. thinking, learning, motivation, emotions, etc.) trigger appropriate stored knowledge that is afterwards used for updating previous knowledge, creating new knowledge, reasoning making decisions about starting an activity etc.

An AI system should behave in a similar fashion. It should also be capable of learning knowledge that is stored on a knowledge carrier. The learned knowledge should then become "personalized", embedded and stored in some manner, and available to artificial cognitive processes.

The authors of this paper focus on text-based knowledge (verbalized knowledge) and its embedding in an AI system. A text consists of several mutually connected sentences, but it can also consist of a single sentence, or just a single word. *S*top! and *M*arko has finished college. are examples of a text that contains only a single sentence. In order to understand such a text context is important, i.e. former knowledge about the meaning of the word *Stop*, about the individual named *Marko*, about the meaning of the verb *has finished* and the noun *c*ollege etc. Therefore, although the text consists of a single word or sentence, an implicit relationship with its context exists. This rule applies also to a text that consists of several mutually connected sentences (Quirk, Greenbaum, Leech, & Svartvik, 1985). Therefore, the process of embedding and storing text-based knowledge into an AI system should preserve relationships between words that form a sentence inside a text and support its connection with other contextual knowledge.

* Corresponding author. Tel.: +385 91 597 1002; fax: +385 51 211 270.
E-mail addresses: alen.jakupovic@veleri.hr (A. Jakupović), mile.pavlic@ris.hr (M. Pavlić), zdovedan@hotmail.com (Z.D. Han).
[1] Tel.: +385 98 208 757; fax: +385 51 584 749.
[2] Tel.: +385 95 525 6549.

http://dx.doi.org/10.1016/j.eswa.2014.03.006
0957-4174/

2. Basic definitions and terminology

This chapter presents the basic definitions and terminology used in formal language theories that are discussed in this paper (based on Aho & Ullman (1972), Dovedan (1982) and Dovedan (2012)).

A character is a unique, undividable element. An *alphabet* is a finite set of characters, $\mathcal{A} = \{a_1, \ldots, a_n\}$.

A string is formed by placing the characters of an alphabet $\mathcal{A}$ one next to the other. An empty string is defined as a string that does not contain any character. We mark it with ε. The set of all strings over the alphabet $\mathcal{A}$, of unlimited length, plus the empty string, is marked with $\mathcal{A}^*$. $\mathcal{A}^+$ marks the set $\mathcal{A}^* \setminus \{\varepsilon\}$. *Language*, $\mathcal{L}$, is defined as any subset of $\mathcal{A}^*$. According to Chomsky (1957), languages can be classified into four categories (or types):

Type	Name
0	unrestricted
1	context-sensitive
2	context-free
3	regular

Natural languages are type 0.

2.1. Symbols and sequence of symbols

Machine processing of natural languages focuses on strings of finite length that can be considered as a unique, undividable entity. Such character sequences are called *words* or *symbols*.

A *dictionary* $\mathcal{V}$ is the set of all symbols in an alphabet $\mathcal{A}$. The *language* $\mathcal{L}_\mathcal{V}$ over the dictionary $\mathcal{V}$ is any subset of symbol sequences (sentences) from $\mathcal{V}^+$.

2.2. Grammars

There are four grammar types: type 0, 1, 2, and 3, and each generates languages of the same type. For the purpose of this paper we will use a context free grammar which consists of:

N a finite set of nonterminal symbols or variables
V a finite set of terminal symbols (dictionary) disjoint from N, $V \cap N = \varnothing$
P a finite set $\{(\alpha, \beta)$: $\alpha \in N$, $\beta \in (N \cup V)^*\}$
S a distinguished symbol in N, $S \in N$, called starting symbol.

An element (α, β) in P will be written $\alpha \to \beta$ and called a *production*. The symbol "→" can be read as "produces", "can be replaced with" or "is transformed into". If P in a grammar contains the following production:

$\alpha \to \beta_1 \ldots \ \alpha \to \beta_n$

it can be written as

$\alpha \to \beta_1 \mid \beta_2 \mid_{\ldots} \mid \beta_n$

The character "|" is read as "or". β_i are alternatives for α. If P contains a production rule that has a form of

$\alpha \to \varepsilon \mid \beta \mid \beta\beta \mid \beta\beta\beta \mid \ldots$

it is written as $\alpha \to \{\beta\}$. The curly braces delimit the array that can be either left out or written one time, two times, three times etc. The production of the type:

$\alpha \to \varepsilon \mid \beta$

is written as $\alpha \to [\beta]$.

2.2.1. Grammar as a language generator

The *sentential form* of the grammar $G = (N, V, P, S)$ is defined recursively as follows:

(1) The start symbol S is a sentential form.
(2) If $\alpha\delta\gamma$, where $\alpha, \gamma \in (N \cup V)^*$, $\delta \in N$, is the sentential form and $\delta \to \beta$ is production in P, then $\alpha\beta\gamma$ is also a sentential form. It is written as $\alpha\delta\gamma \Rightarrow \alpha\beta\gamma$. The relation $\Rightarrow$ to be read as *directly derives*.

A sentential form of G containing no nonterminal symbols is called a *sentence* generated by G. If for sentential forms α_0, $\alpha_1, \ldots, \alpha_n$, $\alpha_i \in (N \cup V)^*$, $n \geqslant 1$, the following is true

$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \ldots \Rightarrow \alpha_n$

then $\alpha_0^n \Rightarrow \alpha_n$ is an *derivation sequence* of the length n. Generally, it is written as $\alpha_0^* \Rightarrow \alpha_n, n > 0$ and we say that α_0 derives_α_n. Based on these two definitions mentioned above, *the language generated by the grammar G* can be written as

$L(G) = \{w \in V^* : S^* \Rightarrow w\}$

The statement above can be read as: "The language L that is generated by the grammar G is a set of sentences that are produced by an sequence of all possible derivations, beginning with the start symbol S".

2.2.2. Extended Backus–Naur form (EBNF)

The given formalism of writing production rules (or "replacement rules") is a modification of a formalism that is known as the Backus–Naur form or BNF. It was used for the first time in 1963. for the definition of the programming language ALGOL 60. For a simpler and more comprehensive description of the basic syntactic structure of simple sentences of the English language we will adjust and expand the rules of writing the Backus–Naur form and introduce the following rules:

(1) Nonterminal symbols will be written between the characters "⟨" and "⟩", or the will be written in uppercase italics for readability. If the symbol consists of multiple words, the words will be connected by an underline "_".
(2) Terminal symbols are words (classes) of English language which belong to disjoint subsets $\mathcal{V}_i$ of the dictionary $\mathcal{V}$:

$$V = \bigcup_{i=1}^{n} \mathcal{V}_i \quad \mathcal{V}_i \cap \mathcal{V}_j = \varnothing \text{ for } i \neq j$$

Each $\mathcal{V}_i$ contains words with equal characteristics (nouns, pronouns, adjectives, verbs etc.). Words are written upright (non italic). They consist of lowercase and uppercase letters of the English alphabet. Some words classes can contain the suffix 's. In case that a grammar of a language that has a large number of words is being implemented, those words are not explicitly written in production of the grammar, but we presume that they exist in separate files.
(3) BNF is used in defining a context-free grammar in which all production rules have the form

$A \to \beta$

where A is a nonterminal symbol, and β is a sequence that consists of nonterminal symbols, terminal symbols and special (meta) symbols. Nonterminal and terminal symbols must be separated by a single space. Meta symbols are:

(1) $\alpha \mid \beta$ *or* $(\alpha \mid \beta)$ alternative (α or β). Parentheses ( ) may be used wherever needed to influence the grouping of operators "|".
(2) [α] α left out or written once.
(3) {α} α left out or written once, twice, ...

Those are the rules of the Extended Backus–Naur Form (EBNF).

2.2.3. Context-free grammars that describe natural languages

Context-free grammars that are applied to natural languages can be useful only for the presentation of basic sentence structures, without their meaning (semantics). In a natural language a