

G. Nenadić, I. Spasić & S. Ananiadou, Salford, UK

Reducing Lexical Ambiguity in Serbo-Croatian by Using Genetic Algorithms

Abstract. This paper presents an approach to acquisition of some lexical and grammatical constraints from large corpora using genetic algorithms. The main aim is to use these constraints to automatically define local grammars that can be used to reduce lexical ambiguity usually found in an initially tagged text. A genetic algorithm for computation of the minimal representation of grammatical features of textual constituents is suggested. The algorithm incorporates two types of genes, dominant and recessive, which are specific for the features that are analysed. The resulting genetic structure describes the constraints that have to be fulfilled in order to form a correct utterance. As a case study, the suggested algorithm is applied on contexts of prepositional phrases, and features of corresponding noun phrases are obtained. The results obtained coincide with (theoretical) grammars that define the constraints for such noun phrases.

1 Introduction

Increasing usage of electronic data has given rise to a need for efficient and effective processing of texts. Every natural language processing system needs to incorporate a certain amount of relevant linguistic knowledge, usually acquired from the theory and/or from corpora. The basic step in processing a text in a morphologically rich language (such as Serbo-Croatian) involves *initial part-of-speech (POS) tagging*. However, the initially tagged text is intrinsically ambiguous in the sense that every word occurrence is associated with lexical information (e.g. POS tags) that corresponds to every possible lexical interpretation of the word in question, of which only few are applicable in the given context. This is the *lexical ambiguity* that has to be reduced as much as possible prior to any deeper analysis of the text (e.g. parsing, information extraction, machine translation etc). For this purpose, local grammars have been widely used [1, 2]. In this paper we will suggest a novel method for automatic generation of local grammars using the concept of genetic algorithms.

This paper is organized as follows: in Section 2 we briefly present the background of the problem, including a survey of lexical ambiguities and some existing methods for their reduction, as well as a brief overview of general concepts of genetic algorithms. In Section 3 we present a novel approach to acquisition of local grammars for reducing lexical ambiguities, while Section 4 provides a case-study on prepositional phrases in Serbo-Croatian. Finally, Section 5 concludes the paper.

2 Background

2.1 Lexical Ambiguities

The process of initial POS tagging for a morphologically rich language such as Serbo-Croatian (SC) can be fully automatized by using a system of electronic dictionaries [13, 14]. Electronic dictionaries provide grammatical information for simple and compound words that are part of an

electronic text. However, as we have stated above, the initially tagged text is intrinsically ambiguous [1, 10] in the sense that every word occurrence is associated with all of its possible lexical interpretations (e.g. POS tags). Consider the following example: the lexical string 'gospodja je dolazila' (eng. *the lady was coming*) should be tagged as:

{gospodja,.N:fsn+} {je,jesam.V:VP3s} {dolazila,dolaziti.V:PPsf}

but it is initially tagged as:

{gospodja,.N:fsn+:fpg+} {je,jesam.V:VP3s} {dolazila,dolaziti.V:PPsf:PPpn}

meaning that the word *gospodja* (eng. *lady*) can be interpreted either as a singular (*s*) noun (*N*) in nominative (*n*) case or a plural (*p*) noun (*N*) in genitive (*g*) case; in both cases the word is of feminine (*f*) gender. Similarly, the word *dolazila* (eng. *to come*) is either a verb form in preterit tense (singular, feminine gender) or a verb form in preterit (plural, neutral gender). The word *je* (eng. *to be*) has only one interpretation (auxiliary verb *jesam* (eng. *to be*) in present tense, 3rd person, singular).

As a result of initial tagging, a lot of lexical ambiguities arise resulting in highly ambiguous word sequences. Therefore, there is a genuine need for the development of automated disambiguation methods. *Local grammars* have been intensively and widely used for this purpose. A local grammar is a finite-state transducer [2, 10, 11] able to recognize "well formed" word sequences in a text and to choose the proper tags for them. It describes a set of related strings that have a similar structure and that can be processed in a uniform way. It does not describe a sentence as a whole. However, it prescribes lexical constraints that are applied to a local context consisting of a sequence of words.

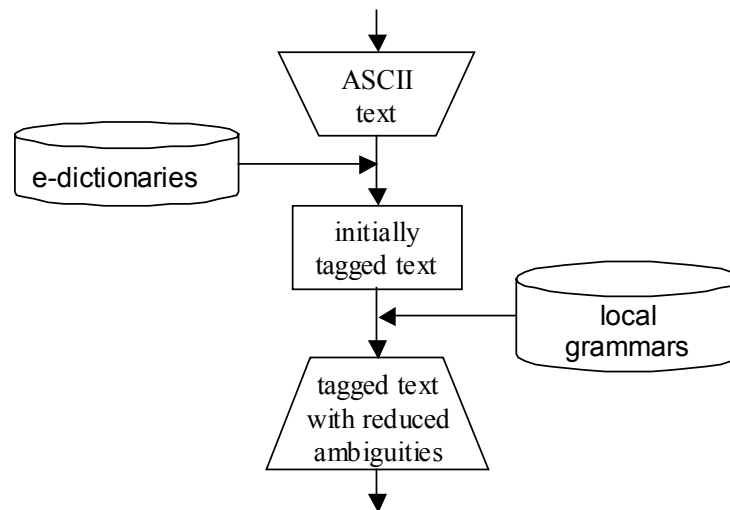


Figure 1: Processing a text using local grammars for disambiguation

Local grammars can be defined in accordance to the rules found in traditional grammar books, but some rules can be added after a corpus analysis [5]. For example, a local grammar that models noun phrases is described in [3], and an example for agreement between a noun phrase's

constituents is presented in [4]. For the sake of illustration, we present a local grammar that describes a local context of the preposition *na* (eng. *on*). According to the local grammar, the preposition in question has to be followed by a noun phrase (*NP*) either in accusative (*a*) or in locative (*L*) case [12]:

$$na.PREP \langle NP:a;NP:l \rangle$$

This way, other tags, if any, associated with an NP can be discarded, thus reducing lexical ambiguities. Unfortunately, it is not always the case that we get fully disambiguated sequence – that is, in general, local grammars are used only to reduce ambiguities.

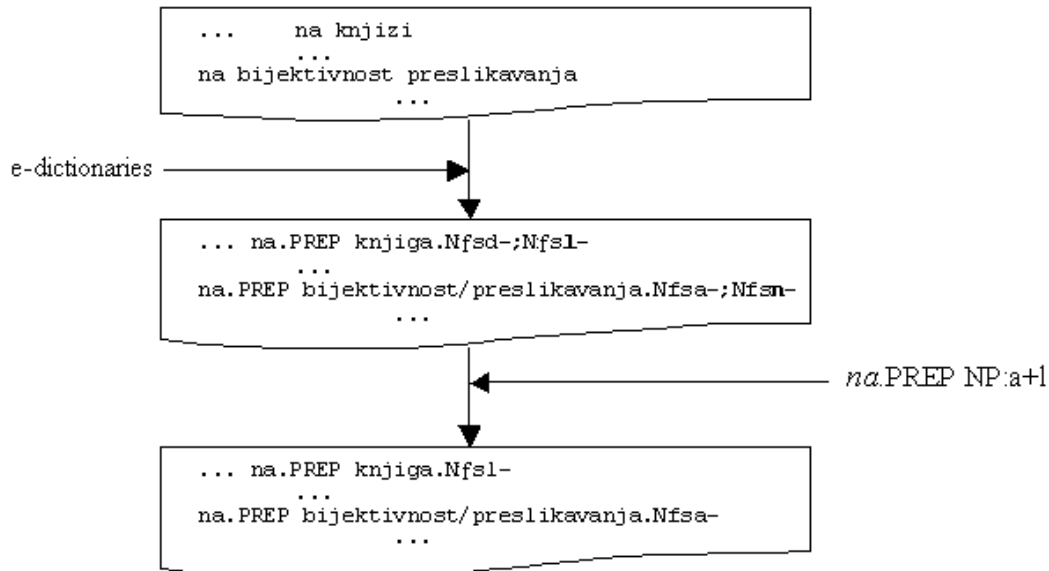


Figure 2: An example of disambiguation for PPs containing the preposition *na* (eng. *on*)

It is essential for a local grammar neither to give incorrect tags nor to remove any possible lexical interpretation that can be applied in the local environment. Local grammars can be used to model complex lexical and syntactical phenomena. For example, they have been used:

- for lemmatization and lexical disambiguation of textual words (e.g. discarding morpho-syntactic information that is not applicable in a specific local context [5]);
- for recognition of syntactical units (e.g. noun phrases [3, 4], coordination of noun phrases [8], compound names [6], adverbial phrases, dates, compound tenses [7]);
- as transducers (e.g. translation from one dialect of a language into another).

Traditional grammar books, viewed as sources for the development of formalism such as local grammar, usually lack the needed exhaustiveness and/or preciseness. Particularly, they are not suitable for implementation of computer procedures, as they are created for human use, and not for automatic processing. Beside the traditional approach, there are some methods developed for automatic local grammar generation. One such method for automatic generation of local grammars for PP is presented in [5]. The approach is based on the extraction of contexts of a

preposition from a corpus, which are then initially tagged using the system of electronic dictionaries, containing all lexical interpretations of individual context constituents. An algorithm for computation of the minimal representation of grammatical features associated with the corresponding noun phrases is suggested. This minimal representation was defined as a generalized intersection of features that were present in the corpus and that were (usually) ambiguous. The resulting set of features describes the constraints, in the form of a local grammar that an NP has to fulfil in order to form a correct prepositional phrase with a given preposition.

2.2 Genetic Algorithms

Genetic algorithms (GAs) are computational meta-heuristics incorporating the principles of natural evolution and the idea of “survival of the fittest” [9]. Let us introduce some of the basic notions used in the GA terminology. A GA is an iterative procedure that operates on a set of individuals. An *individual* is represented by a finite set of *genes*, collectively referred to as the *genotype*. Genotype is a code of a potential solution for a given problem. A group of individuals processed at each iteration of the algorithm is called a *population*.

In the initial phase of a GA a number of solutions (i.e. initial population) is generated, usually at random. Operators typically used in GAs are *selection*, *crossover*, *mutation*, and *replacement*, and they are applied in this order in each iteration of a GA. Namely, after being selected, individuals are recombined by applying crossover between pairs of individuals. Some of the resulting individuals are singled out with a small probability to undergo mutation. Newly created individuals finally replace some of the less fit old individuals, and a new generation of individuals is formed this way. The four operators are repeatedly applied until a stopping criterion is met. We will briefly discuss each of the operators.

Selection is usually defined probabilistically: the better the solution, the higher the probability for that solution to be selected as a parent. *Crossover* is applied to a pair of parents. It results in their recombination, called children. Most widely used crossover operators are: one-point crossover, multipoint crossover, and uniform crossover [9]. In one-point crossover, a crossover point is chosen (usually by random) and two sub-sequences of genes result from breaking the original sequence at the crossover point in each parent. Two heading sub-sequences are swapped to form two children (see Figure 3). Multipoint crossover is similar to one-point crossover, with the difference that more than one crossover points are used (see Figure 4). Uniform crossover further generalizes multipoint crossover making each gene combination possible. Each gene position is chosen with the probability of 50% for the genes at that position to be swapped. In Section 3.2 we will introduce a novel crossover operator that will be suitable for solving our problem.

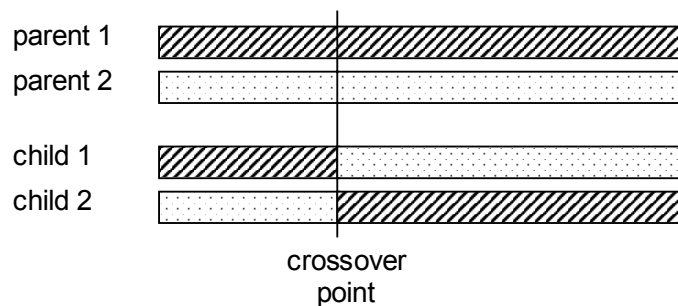


Figure 3: One-point crossover

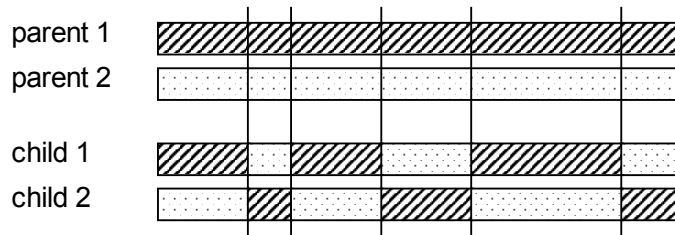


Figure 4: Multipoint crossover

Mutation operator introduces diversity into a population. It modifies a newly formed solution in a random manner. Only a small portion of solutions might undergo mutation. In our approach we do not use mutation at all.

Once a sufficient number of new solutions have been created by applying the three genetic operators, they are evaluated and the fittest ones *replace* the appropriate number of less fit old solutions, thus forming a new population. Predefined quality criterion is referred to as *fitness* and is evaluated by a *fitness function*. This process is repeated from one generation to another until a stopping condition is fulfilled.

In the following we will define GA operators especially tailored to tackle the problem of lexical ambiguities.

3 A GA Approach to Automatic Generation of Local Grammars

3.1 Motivation

By looking at the ambiguous lexical interpretation of a single word taken out of the context it is impossible to automatically reduce the ambiguity. On the other hand, by looking at the whole population of related words it is likely to come to a certain conclusion about their behaviour in general. This was the primary motivation for using a GA for this purpose, as it uses a population of individuals in order to find a near-optimal solution as opposed to using a single individual for the same purpose. The initially tagged corpus provides us with various “populations” which are not random, but instead capture some properties of the individuals. Our goal is to automatically refine these properties in order to find an optimal lexical interpretation of a population as a whole. We relied on the implicit assumption of GAs that it is sensible to re-combine the small pieces of the genotype into bigger pieces that should eventually result in an optimal solution [9].

The initial population will be determined as a set of all initial tags assigned by the system of electronic dictionaries and/or existing local grammars. The suggested GA implements two types of genes, *dominant* and *recessive*, which will ensure that we get desirable features inherited in the next population. In each iteration we will reduce the number of individuals in the population, by replacing two parents with a child until we have just one individual in the population. The resulting “genetic structure” will describe the constraints that the given type of context has to fulfil in order to form a correct utterance in SC. A corresponding local grammar can be derived directly from this genotype.

3.2 Method

When repeatedly applied to a series of populations, selection and crossover operators results in a convergence of the successive populations with respect to the fitness of their members. Namely, selection operator sets the direction of convergence by discarding the individuals that do not satisfy the criteria incorporated into the selection operator, while the crossover operator “evens out” the genetic content across the population by repeated exchange of the genetic material between the population members. In other words, selection operator implies *what* should characterise the population members, and crossover operator determines *how* to produce such individuals. The rate of convergence depends largely on how well these two operators are tailored to a specific problem.

We base our GA approach on the concept of dominant/recessive genes. A *dominant gene* is the one that - when combined with another gene of the same type - will be always reflected in an individual. On the other side, characteristics corresponding to *recessive genes* will be reflected in an individual, only in the absence of a dominant gene of the same type. We will use blood types as a real-world example in order to clarify these notions. There are three blood types (**0**, **A**, and **B**) that can result from a combination of the corresponding genes, **0** being the only one corresponding to a recessive gene. There are six ways in which these genes may be combined. Note that these combinations may include the same gene twice, as they are coming from two parents. The combinations include **00**, **0A**, **0B**, **AA**, **AB**, and **BB**, that respectively result in the following blood types: **0**, **A**, **B**, **A**, **AB**, and **B**. Obviously, characteristics determined by recessive genes are always ‘hidden’ by their dominant counterparts, while characteristics corresponding to two different dominant genes co-exists (e.g. **AB** blood type).

It is logical to choose desirable characteristics to be determined by dominant genes, as we wish them to “override” the less desirable ones when performing crossover between two individuals. It is important to emphasise here that this way it is guaranteed for a child not to be less fit than any if its parents. Thanks to this fact it is not necessary to evaluate the fitness of a child (i.e. there is no need for a fitness function typically used in a GA approach) as the fitness can be viewed as a monotonic function along an ancestry. This fact is used for replacement as well, that is – a newly produced child replaces both of its parents in the previous population. Note that the result of the crossover based on the concept of recessive/dominant genes is deterministic: for a given pair of parents all children will be identical. Therefore, only one child is produced per a pair of parents. Hence, the number of population members is halved in each iteration of a GA, finally being reduced to a single individual that comprises all features determined by dominant genes present in the initial population.

4 Case Study: Prepositional Phrases in SC

For the sake of clear illustration of the method, we will use examples related to the reduction of lexical ambiguities in initially tagged prepositional phrases (PPs) in SC. For simplicity, we will consider only features that relate to the problem of ambiguous case assigned to an NP when a specific preposition (PREP) is present. In this case, a genotype consists of a set of genes, where each gene takes on a value from the set $C = \{n, g, d, a, v, i, l\}$, whose elements represent the

corresponding cases ($n = \text{nominative}$, $g = \text{genitive}$, ..., $l = \text{locative}$). The number of cases in Serbo-Croatian characterizes the problem size and difficulty. Precisely, there are in total

$$\sum_{k=1}^7 \binom{7}{k} = 127$$

hypotheses of the type “the case is either $case_1$ or ... or $case_k$ ” (where $1 \leq k \leq 7$) that can be formulated about a case of a particular NP occurrence. There is always at least one correct hypothesis that can be formulated, i.e. a trivial one: “the case of this NP is either nominative or genitive or dative or accusative or vocative or instrumental or locative”. However, we are not interested only in the correctness of such hypotheses but also in their restrictiveness. The more restrictive a hypothesis is, the better.

Let us now define precisely the genotype representation that we will be using henceforth. A hypothesis “the case is either $case_1$ or ... or $case_k$ ” (where $1 \leq k \leq 7$) will be represented as a sequence $c_1 \dots c_k$, where $c_i \in C$ stands for the corresponding case $case_i$ (for $i = 1, \dots, k$). Note also that a hypothesis concerning the case of a particular NP occurrence is an **exclusive** disjunction and that a string used as a genotype is to be interpreted likewise. As the operator of exclusive disjunction is commutative, the order in which the cases are written is of no importance. We now introduce a hypothesis that combines the hypotheses of the basic type defined afore by using the operator of **inclusive** disjunction. More precisely, if h_1, \dots, h_n is a sequence of n hypotheses of the basic type, represented by the corresponding sequences g_1, \dots, g_n , then the hypothesis “ h_1 or ... or h_n ” will be presented as $g_1 + \dots + g_n$. For example, if the case for an NP can be accusative or either dative or locative, we will represent this hypothesis as $a+dl$. Note that for our problem a hypothesis “ h_1 or ... or h_n ” is a *solution*, its representation $g_1 + \dots + g_n$ is a *genotype*, and disjuncts g_1, \dots, g_n are *genes*.

We use a GA in order to find the most restrictive correct hypothesis about the case, based only on the information contained in a corpus. In other words, we are interested in a **minimal** genotype representation, where this optimality condition is two-dimensional. Namely, we want to minimise both the length of a genotype (i.e. to minimise n in $g_1 + \dots + g_n$) and the length of each individual gene of the genotype (i.e. to minimise k in $c_1 \dots c_k$). In order to achieve this goal we use a selective breeding strategy based on the crossover operator that we now introduce.

Let us discuss a few examples. If we apply the crossover operator on two parents: a (the case is accusative) and an (the case can be either accusative or nominative), we would like their child to have a genotype a (the case is accusative) since it is more restrictive. Therefore, we will say that gene a dominates gene an , i.e. an is a recessive gene in this case. In general, let genes g_1 and g_2 belong to two different parents. We say that gene g_1 dominates gene g_2 (i.e. g_2 is recessive with respect to g_1) iff g_1 is a subset of g_2 . Further, consider what we would like to be a child’s genotype if the parents have genotypes $a+dli$ and $an+dl$. The preferred result should be $a+dl$, as a dominates an (i.e. an is recessive) and dl dominates dli (i.e. dli is recessive). Finally, if the crossover operator should be applied to $a+v$ and $an+dl$ the result should be $a+v+dl$, as an is the only recessive gene (it is dominated by a). Generally, a child is determined by singling out all recessive genes from the union of all genes found in its parents.

The crossover operator can be formally introduced as follows. Let p_1 and p_2 denote two individuals $g_1^1 + \dots + g_n^1$ and $g_1^2 + \dots + g_m^2$ respectively. A child resulting from a crossover operator applied on the two individuals is obtained as a result of the following pseudo-code:

```

for all genes  $g_i^1$  in  $p_1$ 
  for all genes  $g_j^2$  in  $p_2$ 
    if  $g_i^1$  is a subset of  $g_j^2$ 
      then label  $g_j^2$  as recessive;
    else if  $g_j^2$  is a subset of  $g_i^1$ 
      then label  $g_i^1$  as recessive;
collect all genes not labelled as recessive;
the child of  $p_1$  and  $p_2$  is a union
of the collected genes;

```

Obviously, in our case genes are treated as sets, while the dominance between genes is defined as inclusion between these sets. As inclusion is a transitive relation between sets, the order in which individuals are bred is of no importance, i.e. the crossover operator is associative.

As an example, Figure 5 illustrates a sequence of generations, whose individuals represent hypotheses about possible cases for an NP combined with preposition *na* (eng. *on*). The initial population, extracted from the initially tagged corpus, contains the following single-gene individuals: *dl*, *a*, *an*, *dli*, *agvn*, *dla*. The crossover operator is applied to the pairs of individuals, as illustrated in Figure 5, resulting in the second population: *a+dl*, *an+dli*, *av*, *dla*¹. As we see, the only child whose genotype is not a simple union of all genes found in its parents' genotypes is *av*, which is due to the fact that *av* dominates *agvn*. In the next population, we get *a+dl* from *a+dl* and *an+dli* in the same manner. Since the crossover operator is associative, the order in which we apply the crossover operator will not affect the result.

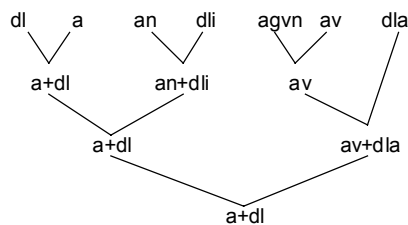


Figure 5: An example of crossover between the case tags for PPs containing preposition *na* (eng. *on*)

We have run the proposed GA for three types of PPs (namely, PPs containing prepositions *prema*, *od*, and *na* respectively). The context for each preposition was extracted from the corpus by using the INTEX corpus processing system [10, 11]. Each initial population corresponds to case properties of NPs that are part of a PP (having a fixed preposition, i.e. either *prema*, *od* or *na*).

¹ As there were an odd number of individuals in the population, one of them is just passed to the next generation without combining with others.

PREP	Initial population sample	Final population	Theoretical local grammar
<i>prema</i>	dl	dl	dative or locative
	dli		
	dlg		
<i>od</i>	ga	g	genitive
	g		
	gnav		
	ng		
	gnv		
<i>na</i>	dl	a+dl	accusative or locative
	a		
	an		
	dli		
	agvn		
	av		
	dla		

Table 1: The results for PPs and comparison with theoretical local grammars

We have compared the results obtained by the GA with theoretically available local grammars [12] (see Table 1) and they were encouraging. For prepositions *prema* and *od*, the final population has exactly the same features as stated in grammar books. However, the final population for the preposition *na* states that a corresponding NP has to be either in accusative case, or in a form that corresponds to either dative or locative. Although this result differs from the grammar books, it is understandable if we bear in mind the fact that locative and dative cases are homographs in SC! In addition, if we compare this corpus-based result to the local grammar mentioned above, we see that our approach did not lose any possible grammatical constraint concerning cases. On the other hand, the "overgeneration" is due to morphological characteristics of Serbo-Croatian, which cannot be resolved without additional linguistic knowledge.

The implementation of the algorithm presented is done in the programming language C. The resulting population can automatically be converted into local grammar and stored for later automatic disambiguation of the corpus.

5 Conclusion

In this paper we have presented an approach to acquisition of some lexical and grammatical constraints from large corpora using genetic algorithms. The main aim is to use these constraints to automatically define local grammars that can be later used to reduce lexical ambiguity. These constraints are learned by a genetic algorithm. The genetic algorithm computes the minimal representation of grammatical features of entities by incorporating two types of genes, dominant and recessive, which are specific for the features that are analysed. The resulting genetic structure describes the constraints that have to be fulfilled in order to form a correct utterance.

As a case study, we have presented the specific application of the algorithm on contexts of prepositional phrases. Features of corresponding noun phrases are obtained from the initially

tagged corpus. The results obtained coincide with (theoretical) grammars that define the constraints for such noun phrases.

The similar algorithms can be implemented for consideration of constraints related to other lexical phenomena. In addition, we believe that this approach is feasible for acquisition of semantic constraints as well, especially in domain-specific corpora. Therefore, an important area of future research will involve development of genetic algorithms that will support word sense disambiguation and information classification.

References

- [1] Gross, M. & D. Perrin (Eds.) (1989): *Electronic Dictionaries and Automata in Computational Linguistics*. In: *Lecture Notes in Computer Science 377*, Springer-Verlag, Berlin.
- [2] Gross, M. (1997): *The Construction of Local Grammars*. In: Roche, E. & Y. Schabes (eds.): *Finite State Language Processing*. Cambridge, MA, The MIT Press, pp. 329-352.
- [3] Nenadić, G. & D. Vitas (1998): *Formal Model of Noun Phrases in Serbo-Croatian*. In *BULAG 23*, Universite Franche-Compte, Besançon, France.
- [4] Nenadić, G. & D. Vitas (1998): *Using Local Grammars for Agreement Modeling in Highly Inflective Languages*. In: *Proceedings of the First Workshop on Text, Speech and Dialogue - TSD 98*. Masaryk University, Brno, pp. 91-96.
- [5] Nenadić, G. & I. Spasić (1999): *The Acquisition of Some Lexical Constraints from Corpora*. In: *Text, Speech and Dialogue - TSD '99, Lecture Notes in Artificial Intelligence 1692*. Springer Verlag, Berlin.
- [6] Nenadić, G. & I. Spasić (2000): *Recognition and Acquisition of Compound Names from Corpora*. In: *NLP-2000, Lecture Notes in Artificial Intelligence 1835*, Springer Verlag, Berlin.
- [7] Nenadić, G., D. Vitas & C. Krstev (1999): *Local Grammars and Compound Verb Lemmatization in Serbo-Croatian*, In: *Selected papers from the Third European Conference on Formal Description of Slavic Languages (FDSL-99)*, University of Leipzig.
- [8] Nenadić, G. (2000): *Local Grammars and Parsing Coordination of Nouns in Serbo-Croatian*. In: *Text, Speech and Dialogue - TSD 2000, Lecture Notes in Artificial Intelligence 1902*, Springer Verlag.
- [9] Reeves, C. (1996): *Modern Heuristic Techniques*. In: Rayward-Smith, V. et. al (Eds.) *Modern Heuristic Search Methods*. John Wiley & Sons Ltd.
- [10] Silberztein, M. (1993): *Dictionnaires électroniques et analyse automatique de textes: le systeme INTEX*. Masson, Paris.
- [11] Silberztein, M. (1994): *INTEX: A Corpus Processing System*. In: *Proceedings of COLING 94*. ACL, Tokyo.
- [12] Stanojčić, Z. & Lj. Popović (1994): *Gramatika srpskoga jezika*. Zavod za udžbenike i nastavna sredstva, Beograd (in Serbo-Croatian).
- [13] Vitas, D (1993): *Mathematical Model of Serbo-Croatian Morphology (Nominal Inflection)*. PhD thesis. Faculty of Mathematics, University of Belgrade (in Serbo-Croatian).

- [14] Vitas, D., C. Krstev, G. Pavlović-Lažetić & G. Nenadić (1998): Recent results in Serbian Computational Lexicography. In: *Monograph on 125th anniversary of the Faculty of Mathematics*. University of Belgrade, pp. 111-128.