A Hybrid System for MorphoSyntactic Disambiguation in Bulgarian*

Kiril Iv. Simov and Petya N. Osenova†
BulTreeBank Project
http://www.BulTreeBank.org
Linguistic Modelling Laboratory
Bulgarian Academy of Sciences
Acad. G. Bonchev St. 25A
1113 Sofia, Bulgaria
kivs@bgcict.acad.bg, osenova@slav.uni-sofia.bg

August 3, 2001

1 Introduction

The MorphoSyntactic Disambiguation Problem (MSDP) similarly to the Part-Of-Speech (POS) disambiguation problem attracts researchers' attention in NLP since it became obvious that certain partial analyses can be useful in practice for such tasks as document indexing, reducing the ambiguity in subsequent parsing stages and others. The interest in this problem is also supported by the hope that it is decidable with a high percentage of certainty without deep syntactic analysis to be involved. In languages with rich morphology, like Bulgarian, the tagset is likely to increase in size (for example, for one of the Spanish tagsets [...,] the number of the tags is as high as 475. (Garside, Leech and McEnery 1997)). Hence defining an adequate tagging scheme becomes a question of importance. The main problem with having so many tags is the well-known problem of the sparseness of a corpus, i.e. from a set of linguistic descriptions only a few are frequent in the corpus. Thus representativeness with respect to all grammatical features relies on the very large size of the corpus. This phenomenon motivated us to choose compositional tags instead of atomic ones. As each word in our corpus is connected with a bunch of grammatical features, it happens that less amount of text demonstrates more dependencies between these features.

The work reported in this paper is an improvement over the work done on POS disambiguation for Bulgarian via Neural Networks (Vlasseva 1999). Our improvements are in several directions: (1) we extended the range of grammatical features predicted by the system to cover almost all paradigmatic members of Bulgarian words, (2) we changed the encoding schemata for grammatical features in order to minimize the computation and to use more extensively the context layer of the network, (3) we changed the evaluation of the network output in order to minimize the side effects from evaluating cases that are not relevant in a particular instance of ambiguity. Besides the improvements when using neural networks, we did some improvements on the choice of the training corpus and we added a rule-based preprocessing component in order to disambiguate the cases for which there are rules ensuring 100% correct results.

In general, a disambiguation problem is to attach the right category to a textual element from a set of possible categories for this element. In the case of MorphoSyntactic disambiguation we have to choose the right morphosyntactic features of a word in a text from the morphosyntactic features connected with the word in the lexicon. In our work we concentrate on the morpho-syntactic disambiguation within the sentence although in some cases the

*This work is funded by the Volkswagen Stiftung, Federal Republic of Germany under the Programme "Cooperation with Natural and Engineering Scientists in Central and Eastern Europe" contract I/76 887. It is also partially supported by Center of Excellence BIS-21 grant ICA1-2000-70016.

† Also at the Bulgarian Language Division, Faculty of Slavonic languages, St. Kl. Ohridsky University, Sofia, Bulgaria

right choice depends on data from the surrounding text. Such cases are described in our lexicon and they receive a special marking before the work of the neural network. The output of the system contains this special marking and information about the most probable set of features predicted by the net for the given element.

The structure of the paper is as follows: in the next section we present Simple Recurrent Neural Networks as they are described in (Vlasseva 1999), the third section gives an overview of the morphosyntactic ambiguities in Bulgarian, then we present the encoding schemes for the grammar features and the evaluation of the output, next section discusses the rule-based component of the system, section 6 outlines the principle according to which we chose the sentences in the training corpus, at the end we conclude with some results and possible future work.

2 Simple Recurrent Neural Networks

Natural language and the mechanisms that people use for understanding it are rather complex. When reading a text one can go back to the beginning of a given sentence (even to the words in previous sentences) or look at the words that follow and on the basis of the data the correct information (including morphosyntactic features) for a given word in that sentence can be deduced. In our case such mechanisms are unfeasible so we concentrated on one-sentence-level, marking word elements with outside sentence dependency as overspecified with respect to the corresponding features. To determine the morphosyntactic features for any given word in a sentence, the succeeding and/or the preceding words can help. These characteristics of a natural language suggest that in order to model successfully the problem of morphosyntactic disambiguation, a paradigm capable of dealing with sequential data should be followed. That is why the Simple Recurrent Network (SRN) model has been considered here as the best candidate for our purposes. SRNs have been developed as an extension to the popular feed-forward supervised neural network model—the Multi-layered Perceptron—and it does the same basic computations. The core element in the SRNs (as in the all kinds of neural networks) is the neuron. Each neuron has one output, which is related to the state of the neuron (its functional signal), and the neuron's input may be connected to several other neurons. Its inputs arrive over these connections. They are the activations of the incoming neurons, multiplied by the weights of the corresponding connections. The neurons are organized in layers.

The difference between Multi-layered networks and SRNs is the recurrent connection from hidden to context layer, which can be considered as part of the input layer, but fed with internally produced data. The context layer remembers the activation of hidden layer neurons from the previous moment and thus it serves as temporary memory in the SRN. Therefore, at each moment the response of an SRN depends both on its current input and on its contextual memory. During the learning, SRNs build an internal representation of the sequential input data in their context layer. The activation of neurons from the context layer is a (modified) copy of the activation of hidden neurons. On the next iteration, when the functional signal propagates forward through the network, the context neurons take part in computing the activation of hidden layer. Internal activation for hidden neuron j is computed as:

$$v_j(n) = \sum_i w_{ji}(n) x_i(n), \tag{1.1}$$

where i varies over all neurons from both the input and context layers; $w_{ji}(n)$ denotes the weight connecting the output of neuron i to the input of neuron j on iteration n; and $x_i(n)$ denotes the i^{th} neuron from the input or context layer on the same iteration, respectively.

This copying process can be implemented in two different ways:

- Direct copying—the activation of a hidden neuron is copied (without any changes) into the corresponding context neuron:

  $$c_i(n) = h_i(n), \tag{1.2}$$

  where $c_i(n)$ denotes the i^{th} context neuron at moment n; and $h_i(n)$ represents the i^{th} hidden neuron at moment n.

- Using both the current activation of hidden neurons and the context activation from the moment before:

  $$c_i(n) = (1 - \beta)c_i(n-1) + \beta h_i(n), \tag{1.3}$$

  where β is a constant from the interval [0.2,1]; c_i denotes the i^{th} context neuron at moment n (or $n-1$, respectively); and $h_i(n)$ represents the i^{th} hidden neuron at moment n.

More precisely, when previous context is used the network will store histories of past activations in rather broad boundaries. The parameter β provides flexibility, since its value alters the weight (importance) of the participating hidden and context layer neurons in the computation of new context neuron activations. On the downside, the optimal value for this constant can not be determined in advance, because it is problem specific. Many experiments and observations are necessary in order to narrow down its correct value.

The next step is to compute the functional signal. It represents the contribution of each neuron in the activation of neurons from the next layer. The functional signal in the output of a neuron, which is the result of applying the activation function on the internal activation, is represented as:

$$\phi(v_j(n)) = \frac{1}{1+e^{-v_j(n)}} \tag{1.4}$$

In our implementation, the logistic function has been chosen as the most commonly used activation function in implementations of neural nets and in particular in the field of natural language processing with NNs, where the results have been promising.

To adjust the weights in the network we use the so called delta rule:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n), \tag{1.5}$$

where η is the learning rate, $\delta_j(n)$ denotes the local gradient for the j^{th} neuron at moment n, $y_i(n)$ represents the functional signal in the output of the i^{th} neuron at moment n. The local gradient is defined as follows:

- If the j^{th} neuron is from the output layer, then:

  $$\delta_j(n) = e_j(n)\phi'_j(v_j(n)), \tag{1.6}$$

  where $e_j(n)$ refers to the error signal at the output of neuron j for iteration n, and is defined as:

  $$e_j(n) = d_j(n) - y_j(n), \tag{1.7}$$

  where d_j determines the expected response of the neuron j, and y_j represents the actual response.

- If the j^{th} neuron is from the hidden layer, then $\delta_j(n)$ equals the product of the associated derivative $\phi'_j(v_j(n))$ and the weight sum of the δ's, computed over all neurons in the next layer that are connected to neuron j:

  $$\delta_j(n) = \phi'_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(n), \tag{1.8}$$

For the learning process, the BackPropagation algorithm is used (see (Tveter 1999)). The current implementation used pattern-by-pattern updating of the weights. In this mode of operation, the algorithm cycles through the training data as follows:

1. **Initialization**—after a suitable neural network architecture is selected for the particular problem, all weights in the net are set to small random numbers.

2. **Presenting the training examples**—as input to the network either a single pattern or a whole training set are presented (depending on the mode of training—pattern or batch). Then, for each example the sequence of forward and backward computations in steps 3 and 4 are performed.

3. **Forward computations**—each training example is an ordered tuple $[x(n), d(n)]$ of an input and output vector representing the expected result for a given input. For each input pattern the internal activations are computed and the activation function is applied to the results ((1.1) and (1.4)). This process continues through to the output layer where the functional signals for the output neurons are computed. This in turn allows for the error signals to be computed (see (1.7)).

4. **Backward computations**—in this stage of the algorithm the local gradients are computed and the weights are adjusted (see formulas (1.5) to (1.8)).

5. **Iteration**—new epochs of training examples (an epoch is defined as one complete presentation of the entire training set during the learning process) are repeatedly presented to the network until its behavior stabilizes.

Having been motivated by our previous experience in using SRNs for POS disambiguation we use the sliding-window technique for preparing input-output pairs of examples for the training and testing phases. This technique is used, of course, during exploitation of the trained network. A window is a fixed-width representation of words that forms the complete input pattern for the network. We change the size and the shape of the window in order to demonstrate better results for MSDP—we use a seven word asymmetric window. Thus at each moment the input of the network represents the encoding of morphosyntactic features of seven words in the sentence. The output of the network is the prediction for the characteristics of the third of these words. In this way we use a longer context after the current word and we rely on the context layer to memorize information about the left context of the word. This window moves along the sentence and the network assigns morphosyntactic features to each word in the sentence. When the window exceeds the sentence (at the beginning and the end of the sentence) "blank words" are added.

3 MorphoSyntactic Ambiguity in Bulgarian

Bulgarian belongs to the group of the highly inflected languages. Suffice it to mention some of its morphological features as: gender, number, definite article (concerning nominals) and person, number, nine tenses, voice, moods, evidentials (concerning verbals).

Main ambiguity-bearing cases are divided into two groups: lexical and structural ones. Lexical ambiguity interacts preferably with homonymy word relations and structural ambiguity is supposed to be triggered by the word linear distribution.

Instances of lexical ambiguity:
Wordforms of the same lexeme:

- non-person singular masculine with attached definite article in its short variant vs. its numeral plural:

  stola vs. (dwa) stola
  chair-the vs. (two) chair-pl
  the chair vs. two chairs

- second person of the verb of third conjugation in Aorist (Aor) vs. third person of the verb of third conjugation in Aorist:

  (ti) vidya vs. (toj) vidya
  (you) see-Aor vs. (he) see-Aor
  (you) saw vs. (he) saw

- all verbs of third conjugation in Aorist vs. all verbs of third conjugation in Imperfect (Imper) except for the forms for second and third person singular:

  (te) risuvaha vs. (te) risuvaha
  (they) paint-Aor vs. (they) paint-Imperf
  (they) painted vs. (they) were painting

Wordforms of different lexemes:

In Bulgarian there is a wide variety of cases, but we'll mention just some of them. It is worth saying that the stress differences of the wordforms (the so called homographs) which appear in some cases are ignored as their disambiguating force is suppressed by the written text form:

- short forms of Possessive Pronouns (Poss) vs. short forms of Dative Pronouns (Dat):

  majka mu vs. kazhi mu
  mother he-Poss vs. tell he-Dat
  his mother vs. tell him

- some plural feminine nouns vs. third person singular of the verb in Present Tense or in Aorist:

  iskri vs. iskri
  sparkle-pl vs. sparkle-3person,sg
  sparkles vs. sparkles / is sparkling

- some feminine singular adjectives vs. some impersonal verbs:

  nyama vs. nyama
  dumb-[fem,sg] vs. 'there is'-negation
  dumb vs. there is not

- neuter singular adjectives vs. adverbs:

  1. ending in '-o':

     tajno vs. tajno
     secret-Adj[neut,sg] vs. secret-Adv
     secret vs. secretly

  2. ending in '-ski':

     istinski vs. istinski
     true-Adj[neut,sg] vs. true-Adv
     true vs. truely

- Past Participle (PP) in -n vs. Adverb vs. neuter singular adjective:

  vyzhiteno vs. vyzhiteno vs. vyzhiteno
  delighted-PP vs. delighted-Adj vs. delighted-Adv
  delighted vs. delighted vs. delightedly

- some feminine singular nouns vs. masculine singular nouns with attached to them short variant of the definite article:

  botanika vs. botanika
  botany vs. botanist

Instances of structural ambiguities:

1. Regular ambiguities (adjectives and participles, prepositions and conjunctions, adverbs and pronominal adverbs)

   These could be defined as a functional type as well. Generally speaking, this kind of ambiguity can be solved pre-theoretically. This step is facilitated by the existence of some 'mapping effect' between the parts of the speech. For instance, demonstrative pronominal adverbs are included in the more general group of adverbs. Therefore both labels are correct, but only the latter is the most specific.

   At the same time a lot of participles function attributively in phrases and as a consequence of this some of them are considered to be adjectives in dictionaries. In fact, we could recognize the original participles through their endings (-l, -en, -t, -st), but when in NP, they have adjective-like behaviour. For instance, in (Bresnan 1995) a morphological process of participle-adjective conversion is assumed with evidence from English and Bantu. So, the problems here have to do preferably with conventionality and can be solved more precisely on chunk level.

2. Local ambiguities (masculine vs feminine nouns)

   If we ignore the achievements of the trained corpus, some of the local ambiguities could be solved on chunk level only, because we need to resolve the local dependencies (agreement, government and others) which are crucial for disambiguation. In this respect some preliminary rules, based on such dependencies would be useful, too.

3. Non-local ambiguities (masculine vs neuter pronouns, 2 person vs 3 person verb forms)

   These can be solved on discourse level, because not only pure grammatical information is required, but referential one as well. Difficulties in this respect arise from the fact that Bulgarian language belongs to the group of null subject languages, i.e. languages in which the pronominal subject is usually dropped.

4 Modelling of MorphoSyntactic Disambiguation in SRNs

After we chose the neural network architecture for the solution of MSDP we had to answer the following questions: how to encode the morphosyntactic features carried by the words in the sentences and how to evaluate the output of the network.

The encoding of the grammatical features has to be done in terms of number of neurons and their values. If we chose to represent grammatical features as tags where each tag is a mnemonic name for a bundle of features that can be assigned to a lexical item, then one possible encoding would be the following: when each tag is represented by one neuron and if the tag is a possible description for the word then this neuron has value 1, otherwise its value is 0. Such encoding was accepted for the case of POS disambiguation in (Vlasseva 1999). A drawback of this kind of encoding is that it requires a very large number of neurons for a larger tagset. So we decided to choose a compromise and represent bundles of grammatical features instead of atomic tags. Thus, instead a tag to be represented as one neuron in the input and output layers, it is represented as several neurons depending on the grammatical features the tag stands for.

In our experiments the morphosyntactic information for each word in sentences is encoded as 36 neurons where the first 15 neurons represent all parts-of-speech in Bulgarian with participles, verbal adverbs, personal and possessive pronouns, and cardinal numerals separated as independent parts of speech, next three neurons stand for genders (masculine, feminine, neuter), then 3 neurons for number and so on for all morphosyntactic features in Bulgarian. For instance, the wordform 'vyzhiteno' from examples given in the previous section will be encoded as the following vector:

[0,**1**,0,0,**1**,0,0,0,**1**,0,0,0,0,0,0,0,0,**1**,**1**,0,0,**1**,0,0,0,0,0,0,0,**1**,0,0,0,0,0,0]

where the value 1 in the second position encodes the fact that this wordform can be a participle, the value 1 in the fifth position is for adjective, next 1 is for adverb, then the value 1 in the eighteenth represents the fact that the wordform is neuter gender if it is a participle or a noun. The next values 1 are for singular, indefinite and past passive. Thus this encoding incorporates the three possibilities of grammatical features for 'vyzhiteno':

Participle, past passive, neuter, singular, indefinite

[0,**1**,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,**1**,**1**,0,0,**1**,0,0,0,0,0,0,0,**1**,0,0,0,0,0,0]

Adjective, neuter, singular, indefinite

[0,0,0,0,**1**,0,0,0,0,0,0,0,0,0,0,0,0,**1**,**1**,0,0,**1**,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

Adverb

[0,0,0,0,0,0,0,**1**,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

This encoding of the morphosyntactic features for lexical items can be considered as several neural networks that work in parallel: one for part-of-speech disambiguation, one for gender, one for number and so on. In our case these networks are incorporated in one network. One advantage of this is the following: having all features represented simultaneously allows the network to learn not only dependency between values for one grammatical category but also between the values of several grammatical categories. For instance the network can disambiguate between a verb and a noun on the base of the presence of an appropriate gender value for the next word.

The above encoding of grammatical features determines the evaluation of the network output. We proceed in the following order. First we determine the part-of-speech predicted by the network. It means that we consider the

first fifteen neurons and choose the one with the highest value. In fact we decrease the range of the neurons among which we choose to represent the possible parts of speech for this word. In the above example, we consider only the second, the fifth and ninth neurons (we choose between participle, adjective and adverb). Then depending on the result for the part of speech disambiguation we check the values for other features in turn. For instance, if an adjective is chosen then we look for gender, number and definiteness (if they are ambiguous). In the above example this is not the case because 'vyzhiteno' is not ambiguous with respect to gender, number and definiteness, but for other wordforms such ambiguity is possible (see the examples in the previous section).

The actual use of the network includes the suggested encoding and the window-slide technique for prediction of the right features for each word in the text. As to the non-ambiguous words, the system copies their morphosyntactic features to the output. Actually the neural network gets activated, but the prediction is not taken into account and thus its only purpose is the adjustment of the neuron values for the next words in the sentences.

5 Rule-based Disambiguation

In the introduction it was mentioned that our system has also a rule-based component. We applied rules to disambiguate as many ambiguities on morphosyntactic level as possible before applying neural network disambiguator. The general idea was to minimize the input ambiguities. That is why the rules that we invented are applicable in very specific contexts which determine the disambiguation with very high level of certainty—100%. Sometimes when we can not ensure such perfect rules, we rely on the neural network which will work after the rule-based component finishes its job. We added some rules with less certainty but in this case we only modified the encoding of some morphosyntactic features without excluding any possibilities. For instance in the case of 'vyzhiteno' if we have a rule that says to us that it is very probable for the wordform in this use to be marked for gender and number then we reduce the value of the possibility for it to be an adverb. The encoding in this case could be something like this:
[0,**1**,0,0,**1**,0,0,0,**0.6**,0,0,0,0,0,0,0,0,**1**,**1**,0,0,**1**,0,0,0,0,0,0,0,**1**,0,0,0,0,0,0]
where the value for adverb is reduced to 0.6.

Another guideline for the development of rules is for them to solve ambiguities depending on lexical items that are far in the sentence. Such cases are hard to be solved by the window based approach of the neural network because the window couldn't cover the depending lexical items simultaneously.

The rules themselves are developed along the lines of local grammars and they are compiled as finite state automata. Some of the rules use not only morphosyntactic features but also other characteristics of the words like their spelling.

Here are some language-specific rules, designed to resolve the ambiguity that came out after applying the morphological system "Slovnik" (see (Popov, Simov and Chernokozhev 1997), (Popov and Simov 1996) and (Popov, Simov and Vidinska 1998)). In this case many of the possible ambiguities are suppressed and therefore the rules are linguistically simplified:

1. When coming after numerals or some quantity pronouns, masculine singular nouns are put in the special for them numeral plural form.
2. When coming after the particles 'da' or 'ste' the verbs are in Present Tense.
3. We can predetermine the set of compound conjunctions and thus to solve the problem with recognizing the prepositions, contained in them. But the rule doesn't apply to discontinious use of conjunctions.
4. Graphic Principle—when the noun is in capital letter and is not in the beginning of the sentence, it is a Proper name and is distinguished from Common name.

6 Corpus

We paid a special attention to the preparation of the corpus of training and testing sentences. One of the main problems with small corpora is that of sparseness—when the tagset is large then most of co-occurrences of tags miss in the corpus and then the dependency between them cannot be learned. One thing against this problem in our

system is the encoding of grammatical features instead of tags as it was described above. The second suggestion is for the system to be able to solve the most frequent cases of morphosyntactic ambiguity and so we decided to prepare a corpus that explicates them. In this section we describe the preparation of the corpus.

First, we collected a text corpus from Bulgarian texts available on Internet. It contains more than fifteen millions word usages and there are texts from different genres—16% of the texts come from fiction, 76% from newspapers and about 8% from others. The next step was to create a frequency vocabulary of the texts. The vocabulary contains all wordforms in the texts with the number of their usages. Then this vocabulary was analyzed by the program "Slovnik" and a new frequency list was created but this time we counted morphosyntactic features, not words. In parallel to this we developed a small program about automatic extraction of sentences from the text. From the whole corpus we extracted 270 000 sentences. Then we ranked the extracted sentences with respect to frequency of morphosyntactic features of the words in the sentences divided by the length of the sentence. The formula for calculation of the rank of each sentence is:

$$r(S) = \frac{\sum_{w \in S} f(w)}{len(S)},$$

where $r(S)$ is the rank of the sentence S, w is an ambiguous word in S, $f(w)$ is the number of occurrences of the morphosyntactic features of w in the corpus and $len(S)$ is the length of the sentence.

The corpus during the experiments contained 2500 sentences with the largest ranks. We divided the corpus into two parts: training part (1600 sentences) and test part (900 sentences).

7 Conclusion and Future Work

The results of the systems are: 95.17% accuracy for POS disambiguation and 92.87% for all grammatical features. It is important to note that most of the errors concern functional words like prepositions, particles and thus the system can be used for applications which are not concerned with these classes of words, like information retrieval.

The main directions of future work consist of making more experiments, especially with chosen at random sentences. Also we plan to add a post-processing rule-based component in order to repair some of the known errors of the system. When we were choosing which morphosyntactic features to encode in the system we made our choice on two principles—first we chose features that are ambiguous for some wordform and second we chose features that probably play role in the disambiguation of other ambiguous features. We think about reducing the number of neurons in the encoding of the words.

References

Bresnan, J. (1995) *Lexicality and Argument Structure*, In the proc. of the Paris Syntax and Semantics Conference.

Garside, R., Leech G. and McEnery T. (1997) *Corpus Annotation*, Longman, London and New York.

Popov, D., Simov, K. and Chernokozhev, O. (1997) *Slovnik: Morphological processor for Bulgarian*, http://www.diogenes.bg/slovnik/index.html

Popov, D., Simov, K. and Vidinska, S. (1998) *A Dictionary of Writing, Pronunciation and Punctuation of Bulgarian Language*, Atlantis SD, Sofia, Bulgaria.

Simov, K. and Popov, D. (1996) Creating a morphological dictionary of the Bulgarian Language, *in Proceedings of COMPLEX'96 Conference*, Budapest, Hungary.

Tveter, D. R. (1999) *The Backprop Algorithm*, http://dontveter.com/bpr/bptutinps.zip

Vlasseva St. (1999) *Part-Of-Speech Disambiguation in Bulgarian Language via Neural Networks*, Master's Thesis. Faculty of Mathematics and Computer Science, St. Kl. Ohridsky University, Sofia, Bulgaria.