SENTENCE BOUNDARY DETECTION IN CZECH TTS SYSTEM USING NEURAL NETWORKS

Jan Romportl, Daniel Tihelka, Jindřich Matoušek

Department of Cybernetics, University of West Bohemia in Pilsen
Univerzitní 22
306 14 Plzeň, Czech Republic

ABSTRACT

This paper[1] proposes results of an application of a neural network on the problem of deciding whether a certain punctuation mark in Czech text is or is not the end of a sentence. It also discusses possibilities of using methods for relevant parameters extraction and compares a neural network based method with a Bayes classifier and a heuristic classifier.

INTRODUCTION

Probably everyone concerned with text processing agrees that segmenting a text into sentences is certainly not a trivial task. Mainly for the sake of the text-to-speech system ARTIC [1] we have started the development of a sentence boundary disambiguator. Further in this text we will show why we have decided to use a neural network in the task of processing signal determined by the lexical context of ambiguous punctuation marks and we will present a comparison of a neural network classifier with a Bayes classifier and a heuristic classifier.

FORMALIZATION

Indeed, when speaking about sentence boundary detection we actually mean designing an appropriate classifier where the essential task is to set up its internal parameters.

Be S such a set whose elements are ordered n-tuples (the value of n is not important so far) of word forms (technically including also punctuation marks) such that they can surround a text token which can potentially end a sentence ("context" of an ambiguous text token). It is clear that S is a potentially infinite set, thus be $S' \subset S$ such a finite subset which can be created as training data. Be C a set representing two propositions: "ambiguous token is the end of a sentence" and "ambiguous token is not the end of a sentence" (let us say $C = \{1;0\}$).

The ideal classifier is then a function F, e.g. a set whose elements are 2-tuples (c, s), where $c \in C$ and $s \in S$. The partialization of F according to S' is a set F' which can be called a real classifier. The process of classification actually reclines upon creating a model of F' followed by its extrapolation on arbitrary subsets of S. In order to create a model of F' it is useful to have a mapping function $M: S \rightarrow R^n$, e.g. a set of 2-tuples (r, s), where $r \in R^n$, numerically representing possible contexts of ambiguous tokens. The model is then a set F_m of 2-tuples $(c, (r, s))$. If we treat vectors $c_1 \ldots c_j$ and $r_1 \ldots r_j$ (respective to $s_1 \ldots s_j$) as signal, we may model F' as a transformation explicated by the set of equations:

$$c_i = g(W, b, r_i), \text{ for } i = 1..j$$

W and b are parameters of the model which are to be aproximated by a (for instance) gradient method. Since particular neural networks are trained by methods that are actually based on refining the parameters in the direction of a gradient, we can create F_m using a feed forward neural network with non-linear activation functions and back-propagation based learning algorithms.

LINGUISTIC DATA

The first thing to be solved is suitable linguistic data representation, e.g. the mapping function M. In the following text we will call "punctuation mark" such a text token, which can potentially end up a sentence (we considered these punctuation marks: .;?!). For each puctuation mark the neural network is supplied by a pattern determined by the lexical context of this mark and the output is required to generate the value 1 (or as close to 1 as possible) if this punctuation mark is the end of a sentence and 0 if it is not.

We have undertaken experiments with several types of the context of punctuation marks and we realised that best results gives the "2+2 context", e.g. two text tokens left from the punctuation mark and two tokens right, as the following example shows:

<daně> <celkem> . <Doklady> <,>
(lit.: <tax> <in_total> . <Vouchers> <,>)

The pattern determined by the aforementioned context is a vector given by the juxtaposition of 2+2 so called "descriptor arrays" (DA, as in [2]). Each DA belongs to one token from the context and it is a vector whose values are estimations of probabilities of the analytical functions (AFUN) the word (represented by a particular token) appears in. The AFUNs are chosen and estimated from the

1 This research is supported by the Grant Agency of Czech Republic no. 102/02/0124 and the Ministry of Education of Czech Republic, project no. MSM235200004.

Prague Dependency Treebank 1.0 (PDT), see [3], where also all training and testing texts were taken from. DA is then extended by two more values (Abr, Cap) not presented in PDT, thus we can show this example:

DA for word "to" (Czech "it"):

AFUN	*value*	*description*
Pred	0	predicate
Sb	0.533	subject
Obj	0.249	object
Adv	0.031	adverbial
Other	0.046	AFUNs not distinguished
Atr	0.023	attribute
AuxOther	0.081	auxiliary AFUNs not distinguished
Coord	0	coordination
AuxR	0	reflexive particle "se" which is neither Obj nor lexically bound to its verb
AuxP	0.001	preposition
AuxX	0	comma
AuxG	0	other graphical symbols not classified as AuxK
AuxK	0	sentence end punctuation
ExD	0.014	ellipsis (ex-dependency)
Abr	0	abbreviation
Cap	0.112	written with capitals

The values of DA are assigned from a lexicon. The nature of this task does not necessarily need any sort of probability estimation smoothing. There is one more attribute being added to the input pattern – an indication of digits in a token preceding the punctuation mark and an indication of capital letters in a token following. In our first experiments we also used the attribute "token counter" indicating the number of tokens between the punctuation mark and the last recognised sentence end but we surprisingly realised that this attribute slightly deteriorates the classifier performance. The correlation between this attribute and sentence ends is quite high; however, in this case it often causes the classifier to ignore sentence ends where the sentences are short (e.g. 1 or 2 tokens).

Following [2] we have also tried to construct DA using part-of-speech paradigmas instead of analytical functions. However, due to the free word order in the Czech language such an approach is not as effective as [2] reports. We believe higher efficiency can be achieved by concerning syntagmatical configurations because from such a point of view each sentence is a more or less compact entity embodying more regularities. The results of comparing these two approaches will be shown further.

Figure 1 demonstrates the covariance of those patterns from the training set, which represent the cases with sentence ends. It can be easily seen that many attributes do not give any considerable information for classification, thus it would be useful to suitably decrease the pattern dimension.We have carried out some experiments with performance improvement attained by suitable attribute extraction based on higher-to-lower dimensional pattern space projection using Karhunen-Loeve method:

Each pattern is mapped into a space with lower dimension so as a specific criterial function representing an error caused by this projection is minimal. The mapping is represented in the form (highly approximated, though often used) of a tranformation matrix given by the formula:

$$T = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{M_l}\sum_{m=1}^{M_l}(r_{lm}-\bar{r})(r_{lm}-\bar{r})^T$$

where L is the number of classes to be distinguished (2 in our case), M_l the number of training patterns in the class l, r_{lm} the m-th pattern in the l-th class and

$$\bar{r} = \frac{1}{L}\sum_{l=1}^{L}\frac{1}{M_l}\sum_{m=1}^{M_l} r_{lm}$$

The attribute extraction works simply in such a way that each vector r_m from the training set is multiplied by the matrix derived from T and the resulting vector is treated as a new pattern representing particular $s \in S$.

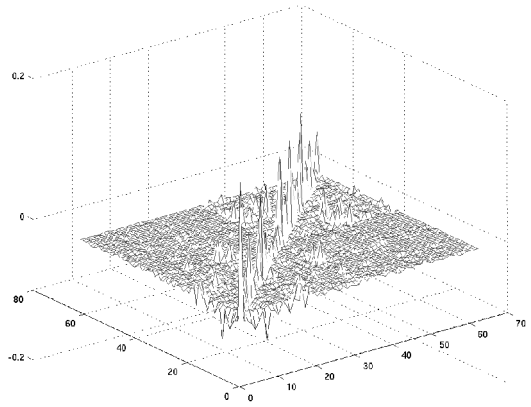

Figure 1. Patterns covariance

NEURAL NETWORK

The structure of the neural network model of the classifier is based on [2]. It is a fully connected feed-foward two-layer neural network (three-layer respectively; however, input layer – hidden layer – output layer networks are often called two-layer) with *a* input units, *b* hidden units and one output unit. The output of the network is a real number from the interval <0; 1>, all units have "sigmoidal" activation functions. The numbers *a* and *b* depend on the choice of input attributes. For the input patterns constructed the way described above it means the number of the input units is 65 (e.g. $a = 65$). The ability of the network to learn the train set without mistakes as well as the speed of the learning depends on the number of the hidden units. We experimentally set this number to 33 (e.g. $b = 33$).

HEURISTICS

For the sake of performance measuring we have also created a Bayes classifier and a simple heuristic classifier which uses a set of rules to disambiguate a punctuation mark. The rules are defined as follows (the order of the actions is important and was made up to minimise the error on the training set):

1. result := T (true, punctuation mark is the sentence end)
2. if the preceding token is a number, result := F (false)
3. if the following token is with a capital letter and followed by a space, result := T
4. if the preceding token is an abbreviation, result := F
5. if the preceding token has only one letter and this letter is capital, result := F
6. if the preceding token is a sentence end, result := T
7. if the preceding token is "(" , result := F
8. if the following token is ")" and is not followed by a token with a capital letter, result := F
9. return the actual value of result

BAYES CLASSIFIER

The Bayes classifier inserts input patterns to different classes so as to maximise the probability that a certain pattern belongs to a respective class according to the Bayes theorem. We assumed the patterns abide the normal (Gaussian) probabilistic distribution. This classifier is supplied by the same input data as the neural network.

TESTING AND RESULTS

All classifiers (neural network, NN; Bayes classifier, BC; heuristics, H) were set up on the training set of 3,667 ambiguous cases with a cross-validation set of 507 ambiguous cases. The overall tests were run on the separate set of 6,772 ambiguous test cases. All test cases were taken from the PDT and come from Czech newspapers and journals. The threshold for the NN output was experimentally set to 0.7 (e.g. if the NN output value is higher than the threshold, the input pattern is recognised as representing the sentence end).

The test set consists of 984 (14.5%) punctuation marks which are not ends of sentences (e.g. 984 negatives) and 5,791 which are (e.g. 5,791 positives). This means that a trivial classifier assigning each punctuation mark the sentence end would achieve 14.5% error rate. The following table shows the error rates of the aforementioned classifiers when disambiguating the test set of 6,772 test cases. Errors can be divided into two categories: false positives (punctuation mark is not the end of a sentence while the classifier says it is) and false negatives (vice versa).

Figure 2 shows the evolution of the mean square error of the training and the validation set during training epochs.

Table 1 shows the error rates achieved in the validation set and briefly summarizes the performance of the neural network in comparison with two other classifiers.

Table 2 shows the error rates of NN with 3+2 context when relevant attribute extraction (projection using Karhunen-Loeve method) was applied. The initial pattern space is of 81 dimensions and this method finds such approximations of the patterns from this space in spaces of lower dimensions that the mean square error is minimal. Again the test set is of 6772 test cases.

CONCLUSION

In comparison with [2] the best results we achieved seem to be notably worse ([2] alleges about 1% error rate). However, one must note that we are quite handicapped partly by the free word order and partly by extensive flexion of the Czech language (we must also recall the difficult nature of the testing text). The impacts of the first reason we try to minimise by using AFUNs instead of part-of-speech paradigm since the patterns constructed this way can better represent a sentence in terms of more or less syntactically and semantically closed language unit. The second reason causes problems when creating a suitable lexicon with probability estimations and this will be in focus of the further research as well as more elaborate exploration of DA and input pattern features (e.g. morphological analyser, pattern attribute pruning, etc.) which means we expect further improvement of the NN performance.

BC has shown not to be very efficient, which is mostly by the reason of normal probabilistic distribution used. It supports the assumption that no language phenomena can be appropriately described by this distribution. On the contrary, H achieved almost the same results as NN. The simplicity and the performance of H thus would speak in favour of H. However, we must be aware of the fact that further improvement of H is very difficult and far from being significantly efficient.

Moreover, the expected improvement and evident universality of NN (it is easily adaptable to be used in other languages) and, last but not least, the possible using of this method for phonemic clause detection (for the sake of TTS system prosody implementation) speak in favour of NN.

REFERENCE

[1] Matoušek, J. - Psutka, J. "ARTIC: A New Czech Text-to-Speech System Using Statistical Approach to Speech Segment Database Construction", In The Proceedings of the 6th International Conference on Spoken Language Processing ICSLP2000, vol. IV. Beijing, China, 2000, pp. 612-615.

[2] Palmer, D. - Hearst, M. "Adaptive multilingual sentence boundary disambiguation", Computational Linguistics 23, 1997, pp. 241-267.

[3] Bohmova, A. - Hajic, J. - Hajicova, E. - Hladka, B. "The Prague Dependency Treebank: Three-Level Annotation Scenario", In Treebanks: Building and Using Syntactically Annotated Corpora, ed. Anne Abeille. Kluwer Academic Publishers.

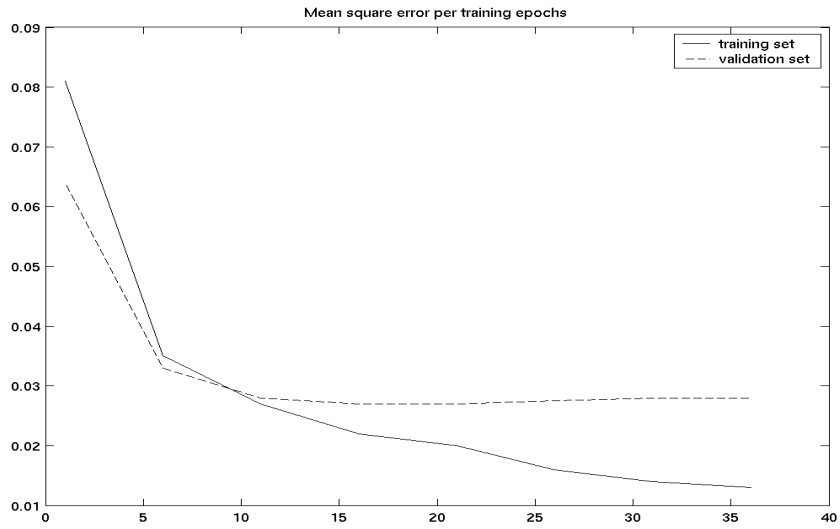

Figure 2. The evolution of the mean square error during training

	NN	***BC***	***H***
errors (per cent)	183 (2.7%)	548 (8.1%)	196 (2.9%)
false positives (per cent)	81 (1.2%)	99 (1.5%)	101 (1.5%)
false negatives (per cent)	102 (1.5%)	449 (6.6%)	95 (1.4%)

Table 1. The error rates of the classifiers with the 2+2 context

pattern dimension	81	60	37	22
errors (per cent)	203 (3.0%)	190 (2.8%)	237 (3.5%)	284 (4.2%)

Table 2. The error rates achieved when using the Karhunen-Loeve transformation on the 3+2 context