Linguistic Text Compression

Ondřej Kazík
Charles University
Faculty of Mathematics and Physics
Prague, Czech Republic
Email: kazik.ondrej@gmail.com

Jan Lánský
The University of Finance and Administration
Department of Computer Science
Prague, Czech Republic
Email: zizelevak@gmail.com

***Abstract*—Compression of texts written in natural language can exploit information about its linguistic structure. We show that separation of coding of part-of-speech tags of a sentence (so called sentence types) from the text and coding this sentence types separately can improve resulting compression ratio. For this purpose the tagging method NNTagger based on neural networks is designed. This article is focused on a specification and formalization of a compression model of texts written in Czech. Language with such a complicated morphology contains a great amount of implicit grammatical information of a sentence and it is thus suitable for this approach. We propose methods of constructing of initial dictionaries and test their influence on resulting compression ratio.**

***Keywords*-text compression; part-of-speech tagging; neural networks**

I. Introduction

It was pointed out in several works concerning text compression (e.g., [22]) that knowledge of the structure of a coded message can be exploited for more successful compression of this message. It turned out that a set of basic elements (words, syllables, characters) as well as a structure of documents is strongly dependent on the used language. This motivates us to exploit implicit linguistic information contained in an input text.

We use during the compression knowledge about structures larger than words, so called sentence types – a sequence of words' tags in a sentence. We assume that the separate coding of a sentence type and words of a sentence with knowledge of their tag is a suitable supplement of common compression methods based on words as the elements.

A tagging method has to precede the compression itself. Since compression is not sensitive to small errors, we will seek for approximative method of assigning tags to the words in a sentence.

Our goal it then to propose such model of sentences which comprehends the sentence type. This model will serve for an effective coding of document's sentences.

In Section II, we provide a brief overview of tagging and compression methods, other than the approaches described in this article. In Section III we introduce the NNTagger, an approximative part of speech tagger based on neural networks. In Section IV we describe the model of a text, sentence and other subelements and coding and decoding algorithms which are based on them. In Section V we compare results of our method with other compression techniques. Finally, in Section VI we summarize our findings and discuss possible future work.

II. Related Work

Many nature language processing applications, e.g., syntactic analysis, require as a preprocessing phase assigning definite morphological information to each word in an input text. This procedure is tagging and consists of a morphological analysis, i.e., assigning of all possible tags to a word alone, and disambiguation; i.e., resolving tag ambiguity by the context of a word. Both these phases are chosen with regard to the used language and tag set, which is the range of the tagging method. The Czech language in comparison with the English language is a highly inflecting language, which needs a more sophisticated processing of suffixes and in many applications it requires a tag set, which contains more detailed morphological information. Also it is highly ambiguous. In the English language, the relative small tag sets are used (e.g., Brown, Penn, CLAWS, London-Lund [23]). By contrast, the Czech tag sets (as the Prague Dependency Treebank positional tag set [11] or the set used in morphological analyzer Ajka [31]) represent broad morphological information containing several thousands of possible tags. For the Czech language there are correspondent morphological analyzers, e.g., the tools Free Morphology [9] and Ajka [31]. There is a large number of disambiguation methods suitable for different languages and tagging sets, for instance visible [23] or hidden Markov models [6], taggers based on a decision tree TreeTagger [30], transformations [3], maximum entropy model [26], exponential tagger [10], averaged perceptron-based model ([5] or MORČE [40] for the Czech language) and artificial neural networks based taggers (NetTagger [29]) which is close to our approach, especially the context net of NNTagger described later.

The text file compression is usually employed to save hard drive space or bandwidth during their transfer. The compression of texts can be classified according to elements of coding, e.g., letters, words or syllables. The methods

 ISBN: 978-1-61208-127-4

based on words require dividing the input document into a stream of words and non-words [24], [25]. Overview of word methods is provided by [36]. There are word-based variants of all main groups of compression algorithms: e.g., word-based Huffman encoding [13], [28], word-based LZW [7], word-based Burrows-Wheeler Transformation [7], [15], and word-based PPM [2], [24]. Syllables as source units for compression can be successfully used in languages, where words are naturally divided into syllables (e.g., Czech, Russian, and German). This approach was firstly used in [39]. Knowledge of regularities of the language, in which an encoded message is written, can be exploited to improve text compression. For English texts, there are several methods of such improvement: e.g., specific order of letters in the alphabet for lexicographic sorting [1], [4], replacing common clusters (n-grams) of letters with one symbol [38], static initialization of PPM method [35]. In order to manage multilingual text files, the methods need to recognize the language of a message and its encoding, like in modifications of Word Replacing Transformation, TWRT [36]. Separation of the parts of speech of each word is used as additional information for compression of English texts and compared with character and word based compression programs also in [38]. Compression of small files is a separate subject of research. One of such approaches is focused on compression of short text files for mobile phones. The low-complexity static partial matching model (PPM) is described in [27]. A sequence of pruned suffix trees is used in statistic model in [17].

Lansky and Zemlicka [22] propose theoretical basis of syllable-based compression, LZWL and HuffSyll algorithms. Right configuration of the characteristic words and syllables dictionaries used by LZWL and HuffSyll algorithms for initialization is emphasized in [21]. Various approaches to small and medium text file compression by means of Burrows-Wheeler Transformation (BWT) are compared in [19]. Suitable inflation of the characteristic syllables and words sets, which are used to initialize the HuffSyll and LZWL methods is examined in [18]. XBW Project [32], [33] studies the compression of large, non-well formed XML files. Its basis is BWT and it implements many other coding algorithms, which we utilize in this paper [20].

III. NNTagger (Approximative Part of Speech Tagging)

Linguistics as a science investigates a language from many viewpoints, e.g., syntax, morphology, lexicology etc. The language we are interested in, the Czech language can be described from these perspectives. The Czech language has a free word order, a rich inflective morphology which lays grammatical morphemes at the end of the word. Also the sets of words belonging to different parts of speech have different properties. For example, the inflective and autosemantic parts of speech (e.g., nouns, verbs etc.) tend to have longer words and greater variation both in a document and among different documents than the non-inflective and synsemantic ones (prepositions, conjunctions etc.). These characteristics can be exploited in our approximative tagging method NNTagger.

Every tagging algorithm has to solve three main problems: tokenization, i.e., division of the input text into basic elements (words, sentences), morphological analysis, i.e., assigning of possible tags to a word alone, and disambiguation which determines the tag according to a particular use of the word in a sentence.

A. Pre-processing

Two questions must be solved before the tagging. First, we focus on a definition of words, which will be the basic element of processing. Second, we should consider a width of the sequence of words, which will be available for the tagging procedure, i.e., how to separate single sentences from an input text.

1) Words: We are extending the definition of words from [22], where authors, with regard to separability, introduced these classes of words: words consisting of small letters (small), those consisting of capital letters (capital), with first letter capital (mixed), numeric and other words. We have taken into account characteristics of training data from the PDT corpus [12]. There not only numeric and alpha words but also punctuation marks are considered as the basic elements on the word layer. This is consistent with the Manning and Schütze [23], where the information about macrostructure of sentence contained in the punctuation is emphasized.

2) Sentences: Now we need to decide, how long blocks of words will be passed on to a tagging algorithm. Annotated available training data (the PDT corpus [12]) contain texts tagged divided into whole sentences. We gather an inspiration from description of the algorithm in [23].

We are passing the sequence of non-special words from left to right and inspecting them as candidates on a sentence boundary. We use this heuristics:

- If the inspected word is ";" or "...", it is marked as the last word in the sentence.
- In the case of "?" or "!", it is marked, if in the same time is not followed by another "?" or "!".
- The inspected word is ".". It is marked only, if the following word is capitalized but the preceding word does not consist of one capital letter (abbreviation of a first name) or this word is not in a list of abbreviations which usually is not placed at the end of a sentence ("mgr", "např", "tzn" etc.).
- If after the inspected word is end of line and next word is not small or if after this word is more than one end of line, we put there the sentence boundary. This is the case of e.g., a heading.

 ISBN: 978-1-61208-127-4

- If immediately after the supposed sentence boundary is a quotation mark, shift the boundary after it.

In the next section we will show that it is unsuitable to pass these large segments to the compression algorithm directly and thus they will be segmented to shorter sequences.

B. Tag Set

The decision about a tag set was limited by available data. In our case it is the annotated corpus PDT with a positional morphological tag system, where each position corresponds to some morphological category. So the tag set should be equivalent to a subset of these positions. The decision was based on a presupposition that the tag should carry an information about characteristics of the given word form, its statistical features and its relation to other words in a sentence. In the same time the sets of word forms belonging to these tags should have minimal overlapping, which disqualify some detailed categories (as the cases). Hence we can focus on the position *Part of Speech* (POS, 12 values) and the position *Detailed Part of Speech* (SUBPOS, 75 values), from which we can also derive the part of speech category. Due to various criteria for subsumption under a part of speech, words from the same set can play various roles in a sentence (e.g., Numerals or Pronouns). The detailed part of speech category would partially reduce this ambiguity. But we decided only for the first category (Part of Speech) because of the following reasons:

- The larger is the tag set, the more difficult is to determine the correct value of the tag. This is due to ambiguity of the division criteria and to shortcomings of the chosen tagging method.
- With an amount of information extracted from the text complexity of the model of language and number of its parameters increase.
- The larger alphabet of symbols, i.e., tag set, corresponds to the larger set of words, which are compound of this alphabet, i.e., sentence types, and also the higher probability of occurrence of a new sentence type. This would complicate the compression.

The selected set of part of speech tags contains thus 11 values: nouns (also marked by `N`), adjectives (`A`), pronouns (`P`), numerals (`C`), verbs (`V`), adverbs (`D`), prepositions (`R`), conjunctions (`J`), particles (`T`), interjections (`I`) and punctuations (`Z`). This category carries according to the entropy estimation about 25 % of the information of a word form.[1]

C. Morphological Phase

The input of this phase is a non-special word w with the length m:

$$w = \alpha_1 \alpha_2 \dots \alpha_m$$

[1]The entropy was measured on the PDT corpus. Shannons entropy of the part of speech is 2.9 bits and of the whole word forms 11.9 bits.

The output is then the vector $\mathbf{y}^m(w)$ with the length equal to the size of the tag set $n = |\mathcal{T}|$. It holds for elements of this vector:

$$y_t^m(w) \in [0, 1]$$

This value can be interpreted as the likelihood that the word w has the tag $t \in \mathcal{T}$ on this level.

We define the auxiliary vector $\hat{\mathbf{y}}(t)$ which can be considered as a certainty that the word has the tag t.

$$\begin{aligned} \hat{y}_t(t') &= 1 \quad \text{if } t = t' \\ &= 0 \quad \text{otherwise} \end{aligned}$$

First, it is possible to tag the non-alpha words. If w is a word containing a symbol from Σ_Z, then w is a punctuation: $\mathbf{y}^m(w) = \hat{\mathbf{y}}(\mathtt{Z})$. The procedure is then terminated.

Similarly, if the word w contains a symbol from Σ_C, it is a number and thus:

$$\mathbf{y}^m(w) = \hat{\mathbf{y}}(\mathtt{C})$$

Next only alpha words are processed, so the small, the capital and the mixed are distinguished. Lets the function $\psi : \Sigma_M \cup \Sigma_V \rightarrow \{1, 2, \dots, 41\}$ assigns to every letter a natural number which corresponds to some basic Czech small letter. [2] A size of letters is irrelevant for this mapping, thus:

$$\forall \alpha \in \Sigma_V : \psi(\alpha) = \psi(\mu(\alpha))$$

For the letters which are not included to the basic Czech alphabet (which have small probability in Czech texts) it assigns some value from this alphabet. For example: $\psi('\ddot{\mathtt{a}}') = \psi('\mathtt{a}')$

We decided to use neural nets for the next processing. This decision was motivated by the amount of information, which contains regular structures of word forms in such inflecting languages as is the Czech language. Another advantage is the absence of any dictionary. On the other side, we expect a lower accuracy than by other methods of the morphological analysis. This should not have a major influence on the compression.

Due to a speed of learning and a number of parameters, we decided to limit relevant letters for a morphological analysis to a fixed window. Symbols of the word w will be inserted to this window justified to the right. This reflects characteristic of the Czech language, where the major part of morphological information carries an end of a word, i.e., suffix and ending. We choose the size of the window to eight characters.

The morphological net is based on the Back-propagation net [37]. It includes input layer units, which form vector $\mathbf{x}^m$ of the length 331.

[2]We consider the basic Czech alphabet as a set of small letters together with their possible variants (with the accent, wedge or circle), total 41 letters. The mapping μ is a bijection after a restriction on this set.

 ISBN: 978-1-61208-127-4

- In a first part of the layer, every unit corresponds to a possible symbol of the input window. It thus consists of $8 \cdot 41 = 328$ symbols and these are set as follows:

$$\begin{aligned}&\forall i \in \{1,\ldots,8\}, j \in \{1,\ldots,41\}:\\ &x^m_{41\cdot(i-1)+j} = 1 \quad \text{if } j < |w| \text{ and } \psi(\alpha_{|w|-i+1}) = j\\ &x^m_{41\cdot(i-1)+j} = 0 \quad \text{otherwise}\end{aligned}$$

- The next two units are coding a size of the word w:

$$\begin{aligned}x^m_{359} = 0 \text{ and } x^m_{360} = 0 &\text{ if } w \text{ is small}\\ x^m_{359} = 1 \text{ and } x^m_{360} = 0 &\text{ if } w \text{ is mixed}\\ x^m_{359} = 1 \text{ and } x^m_{360} = 1 &\text{ if } w \text{ is capital}\end{aligned}$$

- The last symbol contains a length of the word:

$$x^m_{361} = |w|$$

The output vector has the length 10, which is a length of $\mathbf{y}^m$ shortened by the position for punctuation (Z) handled before. On an output of the morphological phase value 0 is being appended on this position. During the training, an input word from the corpus is transformed according the previous definition to the input vector, which is presented to the morphological net and a result is computed. Next, this result is compared to the desired output, i.e., vector $\hat{\mathbf{y}}(t)$, where t is the right tag assigned to the word in the corpus. Weights of the net are subsequently adjusted according to the Backpropagation algorithm. [37]

We transformed the data from PDT [12] to a suitable form, which includes beside word forms and their part of speech tags the information about ends of sentences. The data containing 670528 words were split to training, with which the net was trained, and testing, on which an error rate was measured in an approximate ratio $2:1$.

We set in all cases the learning rate of the algorithm $\alpha = 0.2$. Momentum was not used, because there was no or negative influence on a learning. Training data were passed in six cycles by the net. This procedure was performed on nets with different configurations. For nets with one hidden layer it were 50 (i.e., 331 in the input, 50 in the hidden and 10 in the output layer), 100, 150, for nets with two hidden layers then 50-25, 100-25, 100-50, 150-100.

On the testing data is then with estimated the tag of each word as a maximal value in the output vector of the morphological phase:

$$\hat{t}^m(w) \quad = \quad \arg\max_t y^m_t(w)$$

We see that larger nets do not automatically lead to better performance. Greater amount of parameters makes probably the learning harder and longer. The relative simple net has the best result with only one hidden layer containing 100 units.

D. Context Phase

The goal of the context phase is a processing of information about the surrounding words, the actual context of a word, in order to estimate its real tag. We took up a neural net-based disambiguation method Net-Tagger, described in [29]. Nevertheless there are several differences between this and our approach. It is namely:

- Inputs of context net are not probabilities of tags assigned to the word, which was looked up in a dictionary of word forms, but an output vector of the morphological phase.
- Context net contains a hidden layer. Computational power of the nets without this layer is limited. [37] On the other side, multilayer nets can model more complicated functions.

We take into account for the actual word its left context with size p and right context with size f. Every word in this context is represented by $|\mathcal{T}|$ (i.e., 11) units expressing likelihood that the word has a corresponding tag, based on all available information. This means for the actual word w_i and words in the right context, for which context phase was not yet processed, the outputs of the morphological phase, i.e., vectors $w_{i+1}\ldots w_{i+f}$. For words in the left context, previous outputs of the context layer are available. The input layer thus contains $11\cdot(p+1+f)$ neurons. If $|S|$ is a length of the sentence S, we set the input vector $\mathbf{x}^c$ for the word w_i as follows:

$$\begin{aligned}&\forall j \in \{-p,\ldots,f\}, t \in \{1,\ldots,11\}:\\ &x^c_{11\cdot(j+p)+t} = y^c_t(w_{i+j}) \quad \text{if } j < 0 \text{ and } 1 \le i+j\\ &x^c_{11\cdot(j+p)+t} = y^m_t(w_{i+j}) \quad \text{if } 0 \le j \text{ and } i+j \le |S|\\ &x^c_{11\cdot(j+p)+t} = 0 \quad \text{otherwise}\end{aligned}$$

The output of the net is the vector $\mathbf{y}^c(w_i)$ with a length $|\mathcal{T}|$.

The input of the words from the left context brings a recursion into the computation. This can complicate the learning procedure, especially in the moment, when the outputs are not yet correct. Hence in the time of learning we replace the vector $\mathbf{y}^c(w_{i-j})$, analogously to [29], by weighted average of the previous output vector $\mathbf{y}^c(w_{i-j})$ of the context net and the output of the morphological phase $\mathbf{y}^m(w_{i-j})$:

$$(1-e_\tau)\cdot\mathbf{y}^c(w_{i-j}) + e_\tau\cdot\mathbf{y}^m(w_{i-j})$$

Here e_τ is a coefficient, which in the beginning, when the error of the net is high, is near to 1 and with successful learning falls. We used there for this value an exponential moving average of an error of the net, i.e., ratio of wrong assigned tags, if the error is greater than 0.1. In the other case 0 is assigned to e_τ.

The inputs from the training data were put to the net during the learning according to the previous definitions and

 ISBN: 978-1-61208-127-4

Table I
ERROR RATE OF THE NNTAGGER DEPENDING ON A CONTEXT SIZE. IN THE ROWS ARE SIZES OF A LEFT CONTEXT p, IN THE COLUMNS ARE SIZES OF A RIGHT CONTEXT f.

	0	1	2	3
0	4.72 %	4.74 %	4.67 %	4.61 %
1	4.72 %	4.53 %	4.42 %	4.45 %
2	4.72 %	4.51 %	4.40 %	4.41 %
3	4.72 %	4.48 %	4.43 %	4.37 %

the outputs were vectors $\hat{\mathbf{y}}(t)$, where t is a correct tag of the word. The output of the best morphological net was used as the input of the context net, i.e., the net 331-100-10. The same data source as by the morphological phase was used for the training. The training was processed six times with a learning rate equal to 0.2 and without a momentum.

The output of the NNTagger method is thus a sequence of tags induced from values of the vectors $\mathbf{y}^c(w)$. Formally speaking, the tag belonging to a word w is:

$$\hat{t}(w) \quad = \quad \arg\max_t y_t^c(w)$$

We show in the table I achieved error rates on the remaining test data depending on sizes of the left (p) and the right context (f).

The lowest relative accuracy of the tagger was by interjections which were never tagged. This part of speech occurs in Czech texts only rarely. Greater error rate was also by particles which have disputable morphological characteristics. The highest absolute number of mistakes was between nouns and adjectives in both directions.

IV. COMPRESSION OF SENTENCES

Any natural language utterance contains a great amount of implicit information. Now we will show use of the information about parts-of-speech in an input text for its compression.

We define a *type of a sentence* (or of a sentence part), which is determined by its non-special words $w_1, \ldots, w_n$, as a sequence of part-of-speech tags of this sentence. ($T(S) = t_{1,n}^S$)

A. XBW Project

We adopt the platform XBW [20] and exploit some of its features in our work. The goal of the XBW project is creation of an interface to test of different compression methods with an accent on compression of texts in the XML-format. It implements some of the elementary algorithms for a number coding and decoding (Elias, arithmetic, Huffman coding), string compression (Burrows-Wheeler, PPM, MTF, RLE, LZ), XML files processing, dictionary compression, some of auxiliary data structures (trie) and suggests their collaboration.

B. Sentence Parts

We performed an experiment in order to find out duplicities of sentence types. We split a language corpus between training and testing data in an approximate proportion 2:1. A dictionary of the sentence types was created from the training data. Then there were only 16 % overlapping of the sentence types from the test data and those in the dictionary. Variation of types of whole sentences is thus too high for our purpose. Hence, we propose using shorter segments of a sentence, called *sentence parts*. Due to a heuristic character of the definition, sentence parts do not have to match with sentence structure.

The parsing of an input sentence proceeds as follows. After skipping of an initial sequence of punctuations, the sentence is being passed from the left to the right and divided in these cases:

- Put a boundary of a sentence part after occurrence of punctuation marks ",", ":", """, "(" or ")". If it is immediately followed by some other punctuation marks, move the boundary after them. If there is not a whitespace between the punctuation and a next word (so this word immediately follows), move boundary of sentence part before the punctuation.
- Put the boundary after "-", if between it and the next word is a white space (unlike the case of a composed word or the Czech particle -li).
- Put the boundary before the Czech conjunctions "a", "i", "ani", "nebo" or "či". These conjunctions usually have no commas before them.

After division of the sentence part the parsing continues on the rest of the sentence until the end of sentence is reached.

Types of above defined sentence parts have the duplicity in the dictionary higher, approximately 37 %. Nevertheless the number of new emerging types is relatively high. Thus we will seek for methods of an effective coding of new sentence types.

After the pre-processing we have input text decomposed in sentence parts in this form:

$$S^P = w_1 \cdot s_1 \cdot w_2 \cdot s_2 \cdot \cdots \cdot w_n \cdot s_n$$

where n is length of the sentence, w_i (for $i = 1, \ldots, n$) is a non-special word (alpha word, numeric word or punctuation) and s_i is a special word (even with zero length), which always follows w_i. There is also an sentence type ($T(S_P) = t_{1,n}^{S_P}$) for every sentence part S_P.

C. Models

Lets define a model of text generation which will be used for text compression and decompression.

Four models will be created for these layers of the text: sentence part, sentence type, word, and symbol. Except the first one we can all of them subsume under a template. If we have μ as a basic element (type, word or symbol), then this model (template M) includes:

 ISBN: 978-1-61208-127-4

- *Dictionary of the basic elements* Δ. This is the injective function, whose domain is a set of the known elements and the range is the set of natural numbers.
- *Probability distribution* ρ of the known elements and reserved value esc for elements, which are not found in dictionary Δ. These structures are adaptive. If a new element occurs, new values are added to the Δ and ρ. After the occurrence of an element μ, relative probability $\rho(\Delta(\mu))$ increases. The implementation of r depends on concrete compression method (in this case arithmetic coding). Dictionaries and distributions of elements could be initialized from frequent symbols based on a text corpus. Details about the initialization of dictionaries will be provided in the next section.
- *Model of unknown elements* M^{sub}. If the element is not found in the dictionary, it has to be coded on a lower layer (the models of lengths, symbols, or parts-of-speech).

For coding of the element μ to a bit string with knowledge of the model M we will use this notation:

$$K(\mu|M)$$

In most cases we abstract from the concrete coding algorithm. So the code of symbol with index i and distribution ρ (i.e., $K(i|\rho)$ can be realized by any adaptive algorithm).

We introduce an operator of concatenation:

$$K = K_1 \cdot K_2$$

which means that the bit string K consists of the code K_1 followed by the code K_2.

We do not describe the decoding explicitly. It follows directly from the coding procedure, which is proposed as a bijection between a set of possible texts and a set of possible codes.

For every layer we have thus its own model structure: a model of sentence parts M^S, a model of sentence types M^T, model of words M^W and model of symbols M^C. In addition, we have a model for lengths of words or sentences M^{L_k}, which describes generation of non-zero lengths. Some of these structures can be multiple instantiated, as is the case of the word models for each part-of-speech.

We will describe the compression procedure from above, i.e., from higher structures as files and sentence parts to the single characters.

The input file is decomposed after pre-processing into the sequence of sentence parts $S_1, \ldots, S_m$. These sentence parts are coded one by one and the code of empty sentence λ is appended at the end.

1) Sentence parts:

Model: As we said, a sentence part with length n is determined by a sequence of non-special words $w_1, \ldots, w_n$, a sequence of special words $s_1, \ldots, s_n$, which follow them, and its type, i.e., a sequence of tags, $t_{1,n}^{S_P}$. All of these sequences have the lengths equal to n. Empty sentence part (λ, zero length sequence) has a special meaning and it represents an end of file.

Model of sentence parts consists of these submodels:

- *Model of sentence types* M^T.
- For every part-of-speech one *model of words* $M^{W_N}, \ldots, M^{W_I}$. We have not find any reason for separate models of symbols for each part-of-speech, because the basic distribution of letters differs in the Czech words only slightly. So there is shared model of symbols for each of them. On the other hand, there are differences in average lengths of each part-of-speech, so we proposed independent models of lengths.
- *Model of punctuations* M^{C_Z}.
- *Model of numeric words* $M^{W_{num}}$.
- *Model of special words* M^{W_S}.

The last two have their own models of lengths and symbols.

Coding: Code of a sentence part consists of a code of the sentence type and codes of each word of this sentence part. Thus:

$$\begin{aligned} K(S|M^S) &= K(t_{1,n}^S|M^T) \cdot K(w_1|M_1) \cdot K(s_1|M^{W_S}) \cdot \\ &\quad \cdot \cdots \cdot K(w_n|M_n) \cdot K(s_n|M^{W_S}) \end{aligned}$$

where $K(w_i|M_i) = K(w_i|M^{W_{t_i}})$ if t_i is a tag of the word w_i, different from punctuation or numeral. If the word is numeral, the alpha numeral must be distinguished from numbers. It is accomplished by a one-bit token. If it is an alpha numeral we use its model:

$$K(w_i|M_i) = 0 \cdot K(w_i|M^{W_C})$$

For a number the word is encoded by the model of numbers:

$$K(w_i|M_i) = 1 \cdot K(w_i|M^{W_{num}})$$

On the other hand, a punctuation is a one-character word $w_i = \alpha$, thus the coding is provided directly by the symbol model of punctuations.

$$K(w_i|M_i) = K(\alpha|M^{C_Z})$$

2) Sentence types:

Model: The model of sentence types contains a dictionary Δ^T of sentence types which were in an initialization set or appeared in the input text. The dictionary is represented by a trie (which maps a type to a natural number) over the alphabet of tags and by a table (which maps a value to a type). The next part of this model is a probability distribution ρ^T of items of the dictionary Δ^T, which is used for coding and decoding. The dictionary and the distribution include an element corresponding with the type of an empty sentence λ.

A model of lengths $M^{L_{30}}$ is then needed because of the coding of new types. Lengths of sentence types are at the

same time the lengths of whole sentence parts, so we do not need to code this information on the higher level. Finally we have to consider a model of generation of the unknown type tags. The basic variant is to have an unconditional probability distribution of tags occurrence ρ^P. However, we decided to include the information about right context of the actual tag. So, the resulting model is a bigram model. Instead of one distribution, there is a distribution $\rho^{P_{t'}}$ for every possible tag t' which is on the right side of the actual position t. Together with them there is a distribution $\rho^{P_\emptyset}$ of the tags at the end of the type, which have no right context. We used the entropy of tags and estimated that the improvement on the new types would be approximately 21 %.

Coding: We have to divide the coding of sentence types to two cases. In the first case, the type $t_{1,n}$ is in the dictionary Δ^T. It is then possible to code corresponding value according to the distribution ρ^T:

$$K(t_{1,n}|M^T) = K(\Delta^T(t_{1,n})|\rho^T)$$

The distribution is subsequently adjusted.

Otherwise, the type is not in the dictionary. Thus, the escape symbol is generated and the new type must be coded. This consists of coding of its length n and all of its tags from the right to the left with knowledge of the right context.

3) Words:

Model: There are several instances of a model of words in the general model of texts. There are models corresponding to all part-of-speeches except punctuations (alpha words: $M^{W_N}, \ldots, M^{W_I}$), model of numbers (numeric words $M^{W_{num}}$) and model of special words (M^{W_S}).

Each model M^W is responsible for coding of a word w (the string of symbols $\alpha_1 \cdot \cdots \cdot \alpha_m$). The knowledge about the position of the alpha word in a sentence part is essential for coding of its size. From a measuring of word sizes on the corpus we determined the most probable sizes of a word at the beginning and on following positions of a sentence part. The implicit value on the first position is a mixed word (i.e., word with capital the first letter and small other letters), on the following positions are implicit small words.

Regarding this, there are again the dictionary Δ^W and the adaptive probability distribution ρ^W, both of them include except the esc symbol for new word these function symbols:

- $esc_{M\to C}$, i.e., the token which means that the word is capital at the beginning of a sentence part.
- $esc_{M\to S}$ for small words at the beginning of the sentence part.
- $esc_{S\to C}$ for capital words on the second or another following position.
- $esc_{S\to C}$ for mixed word which is not at the beginning.

The dictionary and the distribution are moreover able to code the empty word λ, as for instance the non-existent white space between a word and the next punctuation. The dictionary is represented by a pair of a trie over the alphabet of UNICODE characters and a table which is maps from the values to the words. In order to code new words, the M^W contains a model of length $M^{L_{20}}$ and a model of symbols M^C (in the case of alpha words representing small letters).

Coding: In the first place, if the word is an alpha word, we have to compare its size (small, mixed or capital) with the predicted one and, if necessary, to emit an escape symbol. Then the small variant of the word $w' = \mu_g(\alpha_1) \ldots \mu_g(\alpha_n)$ is taken into account.

- If the word is capital on the first position:

$$K(w|M^W) = K(esc_{M\to C}|\rho^W) \cdot K'(w'|M^W)$$

- If it is a small word on the first position:

$$K(w|M^W) = K(esc_{M\to S}|\rho^W) \cdot K'(w'|M^W)$$

- If it is a capital word on next positions:

$$K(w|M^W) = K(esc_{S\to C}|\rho^W) \cdot K'(w'|M^W)$$

- If it is a mixed word on next positions:

$$K(w|M^W) = K(esc_{S\to M}|\rho^W) \cdot K'(w'|M^W)$$

- In all other cases (including the non-letter words) the size of word is known:

$$K(w|M^W) = K'(w'|M^W)$$

Next, the w' is coded. If the w' is in the dictionary Δ^W, the code can be emitted directly:

$$K'(w'|M^W) = K(\Delta^W(w')|\rho^W)$$

.

Otherwise, we generate the codes of the escape symbol, the length of the word w' and all of its symbols. Then the dictionary and the distribution are adjusted.

4) Lengths:

Model: Model of lengths (M^{L_k}) contains only a probability distribution ρ^L of $k+1$ values. The values $1, \ldots, k$ are the most frequent lengths. The value $k+1$ is reserved as a special symbol for higher lengths. As usually the distribution is adaptive and after (de)coding of a length l is the relative probability of occurrence of l increased. The range of ρ^L is, on the contrary, fixed and new values are not added into the distribution.

Coding: There are two possibilities. First, if $1 \leq l \leq k$, we code it directly:

$$K(l|M^{L_k}) = K(l|\rho^L)$$

Otherwise $k < l$, the auxiliary overflow symbol is emitted, followed by the alpha code of a surplus value:

$$K(l|M^{L_k}) = K(k+1|\rho^L) \cdot \alpha(l-k)$$

5) Symbols:

Table II
COMPARISON OF VARIOUS METHODS COMPRESSION RATIOS T_B.

Method	Compression Ratio
$\mathbf{A}_1$	4.60
$\mathbf{A}_2$	4.41
$\mathbf{A}_3$	4.19
$\mathbf{A}_4$	3.85
$\mathbf{A}_5$	3.34
gzip	4.43
bzip2	4.38

Model: Model of UNICODE-symbols generation is in the sentence part model included in these instances:

- Model of small letters M^{C_P} shared by models of all alpha part-of-speeches ($M^{W_N}, \ldots, M^{W_I}$)
- Model of characters in numbers $M^{C_{num}}$ (digits, decimal separators, symbol "-") which is part of model of numbers (numeric words).
- Model of punctuations M^{C_Z}
- Model of special words symbols M^{C_W} (e.g., white space)

Each of these models M^C contains a symbol dictionary Δ_C, consisting of two tables mapping frequent symbols to natural numbers and back, and an adaptive probability distribution ρ^C. This distribution includes among others a special symbol esc for new characters.

Coding: The coding procedure of a symbol α has two alternatives. If the symbol is in the dictionary Δ^C, we emit simply the corresponding code:

$$K(\alpha|M^C) = K(\Delta^C(\alpha)|\rho^C)$$

with increase of a relative frequency of $\Delta^C(\alpha)$.

If α is new, we put esc symbol on the output followed by full 32-bit beta code of the character:

$$K(\alpha|M^C) = K(esc|\rho^C) \cdot \beta^{32}(\alpha)$$

.

New element is subsequently added into the Δ^C and the ρ^C.

V. RESULTS

We test there our method in various configurations of the initial dictionaries and the coding algorithm of basic elements.

We tested the mixtures of the initial dictionaries ($\mathbf{A}_1 \ldots \mathbf{A}_5$) on the testing set T_B containing 100 shorter documents (as newspaper articles). We show the results in table II in bpc compared with commonly used methods bzip2 and gzip.

We see the expected improvement of a compression rate for small documents. The variant $\mathbf{A}_3$ already overcame both methods.

Next we present dependency of a compression rate on a size of a compressed file. We split the set T_A by size into three categories and observed results within them. The results are in table III.

Table III
COMPARISON OF COMPRESSION RATIOS FOR VARIOUS FILE SIZES IN T_A.

Method	10–100 kB	100–500 kB	500–1000 kB
$\mathbf{A}_1$	4.21	3.78	3.69
$\mathbf{A}_2$	4.00	3.61	3.51
$\mathbf{A}_3$	3.76	3.42	3.34
$\mathbf{A}_4$	3.47	3.22	3.16
$\mathbf{A}_5$	3.10	2.95	2.93
gzip	3.69	3.46	3.47
bzip2	3.35	2.71	2.57

There is again revealed that variant $\mathbf{A}_5$ achieves best results for all sizes of an input file. Due to this initialization the method overcame successful algorithm bzip2 based on a Burrows-Wheeler transform.

We performed the compression and decompression of the set T_A concatenated into one file in order to find out time demands of the algorithm. The compression of such concatenated file with size 4.76 MB with the variant $\mathbf{A}_5$ took about 82 seconds on the computer with processor unit AMD Sempron 2800+, 1.60 GHz and 960 MB RAM. The decompression took about 4 seconds. Here we can observe the asymmetry between compression and decompression phases.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper we have introduced the method of extraction of information from an input text which would allows better modelling of this text and improve our ability to compress it. Such knowledge has been for us the sequence of tags of words in the text which represent the words' belonging to the parts of speech. The sequence of tags for a sentence is called a sentence type. This makes possible for us to introduce knowledge about the structure of a sentence and functions of particular words in an utterance.

In order to get the sentence type, we have proposed the approximative part-of-speech tagger for the Czech language, the NNTagger, based on neural networks. Both procedures – the morphological analysis and disambiguation – were implemented by Backpropagation nets. Such an approach was motivated by a great amount of information contained in the regular structures of words. The NNTagger was trained on the corpus of tagged texts and tested in various configurations. The slightly lower accuracy of this method is compensated by the absence of any dictionary in both phases and justified by the observation that a compression ratio is not sensitive to small errors.

We have constructed model structures of texts on different layers (a symbol, word, sentence type, sentence part) with respect to the additional knowledge (part of speech, kind of symbol). The coding and decoding procedures have been

 ISBN: 978-1-61208-127-4

implemented based on these models and for that purpose a novel notation has been introduced.

We proposed the methods of creating databases of frequent words and sentence types from a set of Czech documents in order to set initially each of the models. Such initialization can improve the resulting compression ratio especially in domain of small documents. However, each of the models contributes to this improvement with another proportion due to its different function in the language. Therefore, several possible mixtures of the initial dictionaries have been compiled with respect to the improvement of a compression ratio and the size of the databases.

The method has been tested in various configurations of the initialization databases and results compared with both commonly used compression programs and similar purely word-based compression methods. These results proved an advantage of coding of the sentence type as information about syntactic structures separately from the words of a sentence. The creation of the initialization mixtures shows that the sets containing words of a part of speech have different properties, e.g., the variability both in a document and among different documents, the distribution of word lengths etc., and their models must be initialized differently. With such an initialization, the algorithm overcame commonly used compression algorithms gzip and bzip2 in compression of small files up to 100 kB.

Features of our solution can be summarized as follows:

- Approximative tagger NNTagger had slightly lower accuracy than usually used Czech tagging algorithms. On the other hand, it does not need any additional dictionary of word forms or lemmas.
- The method of exploitation of part-of-speech tags sequences as a linguistic information in a sentence has been proposed which allows us better to grasp model of text generation.
- Our text compression algorithm gained 17–27 % better compression ratio in comparison with algorithms based only on words with comparable settings and initialization.

The experiments with the linguistic compression in the Czech language have shown some possible challenges for a future work. The tagging method NNTagger has recorded good results regarding the low costs on additional data; however it has lower accuracy than other methods, particularly on non-inflective words. Thus some of the more sophisticated methods should be used and the effect of a tagger's accuracy on the compression should be tested. Also the frequency of unknown sentence types has been relatively high during the compression, so we plan to implement other string compression method, e.g., PPM [20]. Finally, an open question is an application of analogous compression method on languages with different characteristics. For instance, due to less stress on the word morphology in the English language, appropriate tagger has to be used, the morphological phase of NNTagger should be replaced by dictionary-based method, and another corresponding tag set has to be chosen. On the other side, fixed word order can cause an increase in the significance of sentence types and possibly their better compression ratio. The symbol-based languages, like Chinese, raise other challenges. In such languages, the word segmentation is a hard problem which has to be solved by non-trivial algorithm. Also the model of words would be modified with respect to different relation between words and characters.

Ackowledgement

This work has been supported by the Internal Grant Agency of VŠFS under project IGA VŠFS 7721.

REFERENCES

[1] Abel J. and Teahan W. Universal Text Preprocessing for Data Compression. In: *IEEE Transactions on Computers*, **54** (5), pp. 497–507, (2005)

[2] Adiego J. and Fuente P. On the Use of Words as Source Alphabet Symbols in PPM. In: Storer JA, Cohn M (Eds.). *Proceedings of 2006 IEEE Data Compression Conference*, IEEE Computer Society Press, Los Alamitos, California, USA (2006) 435.

[3] Brill E. *A Corpus-Based Approach to Language Learning*, dissertation, Department of Computer and Information Science, University of Pennsylvania; Philadelphia, 1993.

[4] Chapin B. and Tate SR. Higher Compression from the Burrows-Wheeler Transform by Modified Sorting. In: Storer JA, Cohn M (Eds.). *Proceedings of 1998 IEEE Data Compression Conference*, IEEE Computer Society Press, Los Alamitos, California, USA (1998) 532.

[5] Collins M. Discriminative Training Methods for Hidden Markov Models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Morristown, NJ, 2002; 1–8.

[6] Cutting D., Kupiec J., Pedersen J., and Sibun P. A Practical Part-of-Speech Tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*. Trento, 1992.

[7] Dvorský J., Pokorný J., and Snášel V. Word-based Compression Methods for Large Text Documents. In: Storer JA, Cohn M (Eds.). *Proceedings of 1999 IEEE Data Compression Conference*, IEEE Computer Society Press, Los Alamitos, California, USA (1999) 523.

[8] Gailly J. *The gzip home page.* http://www.gzip.org/ [1 March 2010].

[9] Hajič J. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*, Karolinum; Prague, 2004.

[10] Hajič J. and Hladká B. Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset. In *Proceedings of COLING-ACL Conference*. Montreal, 1998; 483–490.

 ISBN: 978-1-61208-127-4

[11] Hana J. and Zeman D. *Manual for Morphological Annotation*. Revision for the Prague Dependency Treebank 2.0. ÚFAL Technical Report No. 2005–27. Charles University; Prague, 2005.

[12] Honetschläger V et al. *The Prague Dependency Treebank,* ÚFAL MFF UK; Prague, 2006. http://www.ufal.mff.cuni.cz/pdt2.0/ [1 March 2010].

[13] Horspool RN. and Cormack GV. A general purpose data compression technique with practical applications. *Proceedings of the CIPS Session 84*, (1984) 138-141.

[14] Horspool RN. and Cormack GV. Constructing WordBased Text Compression Algorithms. In: Storer JA, Cohn M (Eds.). *Proceedings of 1992 IEEE Data Compression Conference, IEEE Computer Society Press*, Los Alamitos, California, USA (1992) 62-71.

[15] Isal RYK. and Moffat A. Word-based Block-sorting Text Compression. In: *Proceedings of the 24th Australasian conference on Computer science*, Gold Coast, Queensland, Australia, (2001) 9299.

[16] Kochánek J., Lánský J., Uzel P., and Žemlička M. Multistream Compression. In: Storer JA, Marcellin MW (Eds.). *Proceedings of 2008 Data Compression Conference*, IEEE Computer Society Press, Los Alamitos, California, USA, (2008) pg. 527.

[17] Korodi G., Rissanen J., and Tabus I. Lossless Data Compression Using Optimal Tree Machines. In: Storer JA, Cohn M (Eds.). *Proceedings of 2005 IEEE Data Compression Conference*, IEEE Computer Society Press, Los Alamitos, California, USA (2005) 348357.

[18] Kuthan T., and Lánský J. Genetic algorithms in syllable based text compression. In: Pokorný J, Snášel V, and Richta K (Eds.). *Proceedings of the Dateso 2007 Annual International Workshop on DAtabases, TExts, Specifications and Objects*. CEUR-WS, Vol. 235, (2007) 21-34.

[19] Lánský J., Chernik K., and Vlčková Z. Comparison of Text Models for BWT. In: Storer JA, Marcellin MW (Eds.). *Proceedings of 2007 IEEE Data Compression Conference*, IEEE Computer Society Press, Los Alamitos, California, USA (2007) 389.

[20] Lánský J., Šesták R., Uzel P., Kovalčin S., Kumičák P., Urban T., and Szabó M. XBW - *Word-based compression of nonvalid XML documents*. http://xbw.sourceforge.net/ [1 March 2010].

[21] Lánský J. and Žemlička M. Compression of Small Text Files Using Syllables. In: Storer JA, Cohn M (Eds.). *Proceedings of 2006 IEEE Data Compression Conference*, IEEE Computer Society Press, Los Alamitos, California, USA (2006) 458.

[22] Lánský J. and Žemlička M. Text Compression: Syllables. In: Richta K, Snášel V, Pokorný J (Eds.). *Proceedings of the Dateso 2005 Annual International Workshop on DAtabases, TExts, Specifications and Objects*. CEUR-WS, Vol. 129, (2005) 32-45.

[23] Manning C. and Schütze H. *Foundations of Statistical Natural Language Processing,* MIT Press; Cambridge, 1999.

[24] Moffat A. Word based text compression. In: *Software-Practice and Experience* **19** (2), (1989) 185198.

[25] Moura ES., Navarro G., Ziviani N., and Baeza-Yates R. Fast searching on compressed text allowing errors. In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM Press, New York, (1998) 298-306.

[26] Ratnaparkhi A. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proceedings of the First Empirical Methods in Natural Language Processing Conference*. Philadelphia, 1996; 133–141.

[27] Rein S., Gühmann C., and Fitzek F. Compression of Short Text on Embedded Systems. *Journal of Computers*, Vol. 1, No. 6, (2006).

[28] Ryabko BY. Data compression by means of a book stack. *Prob. Inf. Transm.*, **16**(4), (1980). In Russian.

[29] Schmid H. Part-of-speech tagging with neural networks. In *Proceedings of the International Conference on New Methods in Language Processing*. Manchester, 1994; 44–49.

[30] Schmid H. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the 15th COLING Conference*. Kyoto, 1994; 172–176.

[31] Sedláček R. *Morfologický analyzátor češtiny*, diploma thesis, FI MU; Brno, 1999. In Czech.

[32] Šesták R., and Lánský J. Compression of Concatenated Web Pages Using XBW. In: Geffert V et al. (Eds.). *SOFSEM 2008*, LNCS 4910, Springer-Verlag Berlin, Heidelberg, Germany, (2008) pg. 743–754.

[33] Šesták R., Lánský J., and Žemlička M. Suffix Array for Large Alphabet. In: Storer JA, Marcellin, MW (Eds.). *Proceedings of 2008 Data Compression Conference*, IEEE Computer Society Press, Los Alamitos, California, USA, (2008) pg. 543.

[34] Seward H. *bzip2: Documentation.* http://www.bzip.org/1.0.5/bzip2-manual-1.0.5.pdf [1 March 2010].

[35] Shkarin D. *Durilca Light and Durilca 0.4b.* http://compression.graphicon.ru/ds/ [3 February 2007].

[36] Skibinski P. *Reversible data transforms that improve effectiveness of universal lossless data compression.* Doctor of Philosophy Dissertation, University of Wroclaw, (2006).

[37] Šíma J., and Neruda R. *Teoretické otázky neuronových sítí,* Matfyzpress; Prague, 1997. In Czech.

[38] Teahan W. *Modelling English text.* Ph.D. dissertation, University of Waikato, New Zealand, (1998).

[39] Üçolük G., and Toroslu H. A Genetic Algorithm Approach for Verification of the Syllable Based Text Compression Technique. *Journal of Information Science*, Vol. 23, No. 5, (1997) 365-372.

[40] Votrubec J. *Volba vhodné sady rysů pro morfologické značkování češtiny*, diploma thesis, MFF UK; Prague, 2005. In Czech.

 ISBN: 978-1-61208-127-4