Harnessing the CRF complexity with domain-specific constraints. The case of morphosyntactic tagging of a highly inflected language.

Jakub Waszczuk
Institute of Computer Science, Polish Academy of Sciences
`Jakub.Waszczuk@ipipan.waw.pl`

ABSTRACT
We describe a domain-specific method of adapting conditional random fields (CRFs) to morphosyntactic tagging of highly-inflectional languages. The solution involves extending CRFs with additional, position-wise restrictions on the output domain, which are used to impose consistency between the modeled label sequences and morphosyntactic analysis results both at the level of decoding and, more importantly, in parameters estimation process. We decompose the problem of morphosyntactic disambiguation into two consecutive stages of the context-sensitive morphosyntactic guessing and the disambiguation proper. The division helps in designing well-adjusted, CRF-based methods for both tasks, which in combination constitute Concraft, a highly accurate tagging system for the Polish language available under the 2-clause BSD license. Evaluation on the National Corpus of Polish shows that our solution significantly outperforms other state-of-the-art taggers for Polish – Pantera, WMBT and WCRFT – especially in terms of the accuracy measured with respect to unknown words.

KEYWORDS: morphosyntactic tagging, conditional random fields, Polish.

1 Introduction

Morphosyntactic tagging is one of the basic problems in Natural Language Processing (NLP), which can be described as choosing most appropriate morphosyntactic descriptions for each word in the particular sentence. We ignore the problem of lemma-ambiguity in our work, since it doesn't occur often within the context of highly-inflectional languages like Polish or Czech (Smith et al., 2005), and concentrate on the problem of choosing correct values of grammatical attributes. Numerous methods have been proposed as a solution to the tagging problem, ranging from rule-based with hand-crafted rules, through transformation-based with rules automatically extracted from a dataset, to stochastic methods like hidden Markov models (Jurafsky and Martin, 2008) or linear-chain conditional random fields (Lafferty et al., 2001; Sutton and Mccallum, 2006). We focus on stochastic methods in this work.

Best solutions for the English language yield tagging accuracy exceeding 97%, but methods designed for one language do not necessarily solve problems inherent to another one. In case of highly-inflectional languages the first (and rather technical) obstacle is the complexity of the morphosyntactic tagset. Polish tags carry information about a number of grammatical attributes – part of speech (POS), case, number, gender etc. – with 13 grammatical categories in total. The Polish tagset used within the 1-million-word National Corpus of Polish (NCP) (Przepiórkowski et al., 2010) is a positional one: first position of the tag represents the grammatical class and subsequent positions contain category values. In the NCP tagset the first position determines the set of grammatical categories (some obligatory, some optional) used in conjunction with the class. For example, the **subst:sg:nom:m1** tag describes a singular (**sg**) human masculine (**m1**) noun (**subst** – substantive) in the nominative case (**nom**).

There are more than 1000 morphosyntactic tags which can be directly extracted from the NCP corpus. It makes the application of methods, which do not scale with the number of tags very well, hard in practice. One of such methods is the conditional random fields (CRFs) formalism, with tagging complexity which is quadratic in the number of tags[1] and with expensive global training process[2]. After the introduction of CRFs within the field of NLP (Lafferty et al., 2001) many methods have been developed in order to exploit their benefits (conditional, feature-rich modeling) while not being restrained by the expensive, global training process of CRFs at the same time. Collins (2002) has shown that by using discriminative methods hidden Markov models (HMMs) can be trained to achieve comparable performance. Toutanova et al. (2003) have proposed tagging with cyclic dependency network, an alternative to CRFs with similar advantages and, additionally, a much less expensive training procedure. In case of CRFs, Sokolovska et al. (2009) have shown that when the set of bigram features is small, the sparse forward-backward recursions method can be used to significantly speed-up the training process. Other optimization techniques, which make it possible to use CRFs to solve large-scale problems (with large output space, in particular), have been described in (Lavergne et al., 2010). However, all these alternatives either deprive us from the pure context of CRFs, or require some additional assumptions and complicate the implementation. We describe a simple extension of CRFs which makes them better suited to the morphosyntactic tagging and partially solves the problem of high computational cost of the training process. Moreover, it can be used together with more implementation-specific optimizations mentioned above.

[1]In case of the simple first-order model.

[2]Assuming that training is performed with respect to the standard log-likelihood function.

We show that by integrating morphosyntactic analysis results into the internals of CRFs it is easy to obtain a high-quality, efficient solution of the tagging problem, even for a language with rich morphosyntactic structure like Polish. Our method can be used within the context of any language for which a high-quality morphosyntactic analysis tool exists. The analyzer should conform to the principle that, for any given input word, the resulting set of potential morphosyntactic tags is either complete (all theoretically possible interpretations are included) or empty (the tool doesn't recognize the word). Partial results can harm the tagging performance because our method always chooses tags consistent with the analysis output. The current implementation requires that tags are represented in a positional form, but this requirement is of a technical nature and it could be easily relaxed. The only assumption which is actually made by the disambiguation method is that tags can be divided into relatively independent parts which are subsequently modeled in separate CRF layers. In practice, when the size of the tagset is small and only one disambiguation layer is sufficient, there are virtually no requirements about the form or structure of the tagset.

The rest of this document is organized as follows. First we describe the constrained version of CRFs (section 2), which provides a way to restrict sets of possible output labels on individual sentence positions. Restrictions are used to impose consistency between morphological analysis results and CRF output labels. Next we show how the introduced mathematical formalism can be used to perform a context-sensitive morphosyntactic guessing (section 3) and disambiguation (section 4) which together constitute Concraft[3], a state-of-the-art tagging system for the Polish language. Final evaluation on the National Corpus of Polish, carried out in accordance with the guidelines described in (Radziszewski and Acedański, 2012), is described in section 6.

2 Constrained Conditional Random Fields

We now recall the definition of conditional random fields and define their extension which we call constrained conditional random fields (CCRFs) throughout this paper. Similar modification of the CRF formalism can be found in (Smith et al., 2005), while comprehensive introduction to CRFs can be found in (Lafferty et al., 2001; Sutton and Mccallum, 2006). CCRFs is a simple but useful extension which employs constraints over individual labels in the output sequence. We represent each constraint as a set of labels, to which a label on the particular position in the sentence must belong. Those constraints are satisfied both during the inference and parameters estimation algorithms, thus improving their efficiency and allowing the model to be better adapted to a particular problem domain. Within the context of morphosyntactic tagging, constraints can be used to enforce consistency between results of morphosyntactic analysis and the tagging process itself.

For simplicity we consider only a first-order (with features defined over two subsequent labels at maximum) HMM-like (no bigram features) conditional random fields. Without the loss of generality we assume that (i) each word in the input sequence is represented by a descriptive set of observations, with each observation capturing some aspect of the word itself or its context, (ii) features can refer only to observations related to the current word. This design decision makes it easier to separate the notion of conditional knowledge, supplied in form of the input sequence, from the internal workings of the CRF implementation. Definitions and equations given below can be extended to describe models of higher order and models with more complex feature types.

[3]Available at `http://hackage.haskell.org/package/concraft` under the 2-clause BSD license. The acronym stands for *Constrained Conditional Random Fields Tagger*.

The formalism described in this section, as well as its generic extension to the second-order case, have been implemented as separate Haskell programming language modules[4] which can be used with other applications in mind (chunking or named entity recognition, for instance). The Haskell language combines advantages of high-level programming and type safety with excellent performance of generated programs[5], which makes it an ideal candidate for conceptually complex mathematical tasks. Both libraries have been used to perform tests described in this paper, as a statistical core of the Concraft tool.

2.1 Definition

Let O be a set of *observations*, Y a set of *labels* and $X = 2^O$. Let $\mathbf{x} = (x_1 \in X, \ldots, x_n \in X)$ be an input sequence of words, where each word is represented by a descriptive set of observations, and $\mathbf{y} = (y_1 \in Y, \ldots, y_n \in Y)$ an output sequence of labels. We also assume that $x_i = \emptyset$ and $y_i = \delta$ for $i < 1 \vee i > n$, where δ is a dummy label not belonging to the Y set. First-order linear-chain CRF defines a conditional probability of the sequence of labels $\mathbf{y}$ given the sentence $\mathbf{x}$ as a normalized product of position-wise model potentials:

$$p_\theta(\mathbf{y}|\mathbf{x}) = Z_\theta(\mathbf{x})^{-1} \prod_{i=1}^{n} \phi_\theta(x_i, y_i, y_{i-1}) \tag{1}$$

where $\theta = \{\theta_k\}_{k=1}^{K}$ is a set of *parameters*, with the k-th parameter representing the contribution of the k-th *feature* to the overall probability, $\phi_\theta(x_i, y_i, y_{i-1})$ denotes a *potential* on the i-th position of the sentence and $Z_\theta(\mathbf{x})$ is a normalization factor. Potential ϕ is defined as an exponential of the sum of parameters related to features present within the context of (x_i, y_i, y_{i-1}):

$$\phi_\theta(x_i, y_i, y_{i-1}) = \exp\left(\sum_{k=1}^{K} \theta_k f_k(x_i, y_i, y_{i-1})\right) \tag{2}$$

The implemented CCRF handles two kinds of features. A *unigram feature* $(u \in Y, o \in O)$ takes responsibility for modeling dependency between a word described by the observation o and the label u assigned to that word. A *transition feature* $(v \in Y \cup \{\delta\}, u \in Y)$, on the other hand, serves to model dependency between adjacent labels v (which is equal to δ when considering the first position of the sentence) and u. Therefore, the form of the feature function f_k depends on what kind of feature, a unigram one or a transition one, it represents:

$$f_k(x_i, y_i, y_{i-1}) = \begin{cases} \mathbf{1}(y_i = u, o \in x_i) & \text{if } k \text{ identifies unigram feature } (u, o) \\ \mathbf{1}(y_i = u, y_{i-1} = v) & \text{if } k \text{ identifies transition feature } (v, u) \end{cases} \tag{3}$$

where $\mathbf{1}(cond.)$ is equal to 1 when the given condition holds and to 0 otherwise.

Finally, the normalization factor Z is defined as

$$Z_\theta(\mathbf{x}) = \sum_{\mathbf{y} \in Y^n} \prod_{i=1}^{n} \phi_\theta(x_i, y_i, y_{i-1}). \tag{4}$$

[4]Available at `http://hackage.haskell.org/package/crf-chain1-constrained` and `http://hackage.haskell.org/package/crf-chain2-generic` under the 2-clause BSD license.

[5]We relate to the efficiency of optimized programs compiled with a Glasgow Haskell Compiler.

2.2 Constraints

Let $\mathbf{r} = (r_1 \subseteq Y, \ldots, r_n \subseteq Y)$ be a sequence of non-empty constraints over individual labels within the given sentence $\mathbf{x}$. For convenience we assume that $r_i = \{\delta\}$ for $i < 1 \vee i > n$. Definition 1 can be easily modified to take those constraints into account. In particular, the model assigns probability 0 to all label sequences which are inconsistent with the constraints imposed by $\mathbf{r}$:

$$p_\theta(\mathbf{y}|\mathbf{x},\mathbf{r}) = \begin{cases} Z_\theta(\mathbf{x},\mathbf{r})^{-1} \prod_{i=1}^{n} \phi_\theta(x_i, y_i, y_{i-1}) & \text{if } \mathbf{y} \in \prod_{i=1}^{n} r_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The normalization factor within the constrained context is defined as

$$Z_\theta(\mathbf{x},\mathbf{r}) = \sum_{\mathbf{y} \in \prod_{i=1}^{n} r_i} \prod_{i=1}^{n} \phi_\theta(x_i, y_i, y_{i-1}). \tag{6}$$

2.3 Inference

We describe here two methods of inference using CCRFs: finding the most probable label sequence given the sentence and assigning marginal probabilities to individual labels from restricted sets. The first one is used within the context of morphosyntactic disambiguation (section 4), second one within the context of guessing (section 3). Throughout this section we assume that input sentence $\mathbf{x}$, constraints $\mathbf{r}$ and parameters θ are fixed.

2.3.1 Decoding

Given a sentence $\mathbf{x}$ and constraints $\mathbf{r}$, the CCRF model with parameters θ can be used to find the most probable label sequence $\mathbf{y}^*$. Since the normalization factor $Z_\theta(\mathbf{x},\mathbf{r})$ doesn't depend on the label sequence $\mathbf{y}$, the task is equivalent to finding label sequence which maximizes the potential.

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \prod_{i=1}^{n} r_i} \left(\prod_{i=1}^{n} \phi_\theta(x_i, y_i, y_{i-1}) \right) \tag{7}$$

The task of finding $\mathbf{y}^*$ is often called *decoding*. It can be solved efficiently using the *max-product* algorithm (Wainwright and Jordan, 2008), which can be also employed within the context of the constrained CRF model. Let $r[i \rightarrow u]$ denote a sequence of constraints resulting from substituting the i-th element in the sequence r with the singleton $\{u\}$. For $i \in [0..n+1]$ and $u \in r_i$ we denote by $\omega_i(u)$ the partial, maximum cumulative potential acquired between the beginning of the sentence and the i-th position assuming, that label at position i takes the value of $u \in r_i$.

$$\omega_i(u) = \max_{\mathbf{y} \in \prod_{j=1}^{i} r[i \rightarrow u]_j} \left(\prod_{j=1}^{i} \phi_\theta(x_j, y_j, y_{j-1}) \right) \tag{8}$$

It can be also defined in a recursive way:

$$\omega_i(u) = \begin{cases} \max_{v \in r_{i-1}} \ \omega_{i-1}(v) \cdot \phi_\theta(x_i, u, v) & \text{if } i > 0 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

The recursive definition, together with dynamic programming or memoization techniques, can be used to find the most probable label sequence $\mathbf{y}^*$ (the sequence corresponding to the value $\omega_{n+1}(\delta)$) in an efficient way.

2.3.2 Marginal Probabilities

The CRF models conditional probability of the entire label sequence, but conditional *marginal* probabilities also can be computed in an efficient way. Marginal probabilities can be used in the output of the CRF instead of (or together with) the decoded sequence $\mathbf{y}^*$. In this work we use marginal probabilities to perform context-sensitive morphosyntactic guessing, which is described in section 3.

Let $i \in [0..n+1]$ and $u \in r_i$. We define a forward potential sum as

$$\alpha_i(u) = \sum_{\mathbf{y} \in \prod_{j=1}^{i} r[i \to u]_j} \left(\prod_{j=1}^{i} \phi_\theta(x_j, y_j, y_{j-1}). \right) \tag{10}$$

The notion of a forward potential sum is similar to that defined by the formula (8), but instead of maximizing the cumulative potential between the first and the i-th position we are summing up all cumulative potentials between positions 1 and i of the input sentence. Backward potential sum $\beta_i(u)$ is a reverse concept corresponding to the sum of cumulative potentials between positions $i+1$ and n given that the value of the i-th label is equal to u.

It should be noted that both forward and backward potential sums can be used to compute the value of the normalization factor, since $Z_\theta(\mathbf{x}, \mathbf{r}) = \alpha_{n+1}(\delta) = \beta_0(\delta)$ for the input sentence $\mathbf{x}$ and constraints $\mathbf{r}$. Furthermore, it can be shown that all three concepts provide the following formula to determine the marginal probability of label u at position i:

$$p_\theta(y_i = u | \mathbf{x}, \mathbf{r}) = \alpha_i(u) \cdot \beta_i(u) \;/\; Z_\theta(\mathbf{x}, \mathbf{r}) \tag{11}$$

Finally, the efficient algorithm for forward potential sum computation is known as a *sum-product* algorithm (Wainwright and Jordan, 2008) and it is based on the following recursive definition of the forward potential sum:

$$\alpha_i(u) = \begin{cases} \sum_{v \in r_{i-1}} \alpha_{i-1}(v) \cdot \phi_\theta(x_i, u, v) & \text{if } i > 0 \\ 1 & \text{otherwise} \end{cases} \tag{12}$$

which has an analogous structure to the definition presented in equation (9). Backward potential sum β can be recursively defined in a similar way.

2.4 Parameters estimation

We note only that we apply a Stochastic Gradient Descent method in order to find parameter values which maximize log-likelihood of a training dataset with a Gaussian prior over the parameter values.

3 Morphosyntactic Guessing

In this section we describe an adaptation and an application of the formalism described in section 2 to the morphosyntactic guessing. A dictionary of known forms can be used in order to find all possible interpretations – tags and base forms – of the given word. This language processing phase is called *morphosyntactic analysis*. Unfortunately, due to incompleteness of data resources or spelling errors some forms may not be recognized. Morphosyntactic tagging of such words, called *unknown* throughout this paper, is severely hampered. The average number of morphosyntactic tags assigned by the morphosyntactic analysis tool to known words, computed on the basis of the NCP million subcorpus, is approximately equal to 4. In case of unknown words we have to consider every possible morphosyntactic tag present in the tagset. This difficulty is reflected in the evaluation results of taggers designed for Polish (see section 6). Results of the morphosyntactic analysis are especially helpful in selecting correct values of lexeme-specific attributes like gender, since it is often hard to determine their values even on the basis of the occurrence context.

By morphosyntactic guessing we mean a method of determining sets of possible morphosyntactic tags for unknown words in the sentence. Thus, it can be thought of as a complement problem to the morphosyntactic analysis. Conceptually, the division of the tagging process into three separate steps of analysis, guessing and disambiguation, is a superficial one. It can be also pointed out that context-sensitive guessing is – from a theoretical, not practical, point of view – a generalization of the disambiguation process. Disambiguator is just a guesser which always proposes one possible tag in the resulting sequence. Yet, it is easier to reason about those three processing stages and to develop efficient tools designed to solve them separately. For example, if we ignore the problem of unknown words completely, we will be constrained to use methods which manage to handle adjacent words with more than 1000 possible tags at the stage of disambiguation. In particular, the method described in section 4 would not be able to handle such cases in an efficient way.

3.1 Model Adaptation

To use the constrained model described in section 2, every input word has to be linked with a list of possible morphosyntactic tags. Each tag is treated as an atomic entity, that is two tags are considered to be different if they differ on at least one grammatical attribute value. Since the method doesn't adopt any assumptions about the structure of morphosyntactic tags, it is very flexible and can be applied to a broad class of natural languages. In the case of known words the choice of possible labels is simple – morphosyntactic analysis is a source of label restrictions. For each unknown word we should consider all morphosyntactic tags from the tagset as potential interpretations. We use a simple heuristic instead, described in (Radziszewski, 2013), where a special set U – a set of all labels assigned to unknown words in the training corpus – is prepared. Since the U set can be in practice much bigger ($\approx$ 300 tags in NCP experiments) than the average set proposed by the analyzer and it is often the case that two adjacent words are unknown, we use the first-order model for the task of morphosyntactic guessing.

While there are many possible ways of using the CCRF to perform actual guessing, we have tested only the simplest one so far. For each unknown word in the input sequence, with the set of restrictions $\mathbf{r}$ fixed to U, we take k most probable labels according to marginal probabilities determined by the CCRF (see 2.3.2) for some arbitrarily chosen value of k.

To speed-up computations, especially within the context of known words, we implement sparse forward-backward recursions (Sokolovska et al., 2009; Lavergne et al., 2010). To make this optimization effective, a set of model features is constructed with respect to words and chosen labels from the training data T, but not with respect to restrictions in T. It is important to note that we can safely apply this optimization within the context of the sum-product algorithm (used during the training phase), but not with the max-product algorithm (used in decoding). Fortunately, to perform guessing we do not have to use max-product algorithm, since only the sum-product is needed to find marginal probabilities of individual labels.

3.2 Observations

A common solution to the morphosyntactic guessing problem is to propose the set of possible tags on the basis of orthographic features acquired for the particular word (e.g. prefixes and suffixes). By using the CRF sequential model we extend this method to take tags assigned on adjacent positions into account. The advantage of such a solution is that the guesser is able to reject morphosyntactic interpretations on the basis of two factors simultaneously: the context and orthographic features of the word.

Our observation schema contains the following set of observations, which are subsequently used to construct the set of unigram features:

- Lowercase prefixes and suffixes of lengths 1 and 2,
- A Boolean value indicating if the word is known,
- Packed shape of the word, and information whether the word is positioned at the beginning of the sentence, combined into one observation.

where each observation is related to the current word, the shape of a word is a string with all lowercased characters replaced by 'l', uppercased characters replaced by 'u', digits replaced by 'd' and any other character replaced by 'x' („Coling-2012" → „ulllllxdddd"), while a packed shape is a shape with all duplicate code characters removed („ullllxdddd" → „ulxd").

word	observations	choice	interpretations
Szef	{ 1.S, 2.Sz, 3.f, 4.ef , 5.True, 6.ul-True }	subst:sg:nom:m1	{ subst:sg:nom:m1 }
administracji	{ 1.a, 2.ad, 3.i, 4.ji , 5.True, 6.l-False }	subst:sg:gen:f	{ subst:sg:gen:f , subst:sg:dat:f , subst:sg:loc:f , subst:pl:gen:f }
Wołodymyr	{ 1.W, 2.Wo, 3.r, 4.yr , 5.False, 6.ul-False }	subst:sg:nom:m1	*U*
Łatwyn	{ 1.Ł, 2.Ła, 3.n, 4.yn , 5.False, 6.ul-False }	subst:sg:nom:m1	*U*

Table 1: Morphosyntactically analyzed sentence *Szef administracji Wołodymyr Łatwyn – Chief administrator Wołodymyr Łatwyn* prepared for the CCRF guessing. The observations column includes observations extracted for individual words according to the guessing observation schema. The choice indicates the morphosyntactic tag correct within the context. In the last column results of morphosyntactic analysis are presented. In case of unknown words the *U* set designates the set of potential interpretations.

Prefix and suffix observations are the basic type of information needed to determine the set of possible morphosyntactic interpretations of the given word. By using only short affixes we can be sure to get highly accurate estimations of corresponding parameters. The model learns dependencies between prefixes/suffixes and morphosyntactic tags mostly on words from the corpus which are known, since they are far more frequent in the training material than the unknown words. After that it is able to use this knowledge to guess interpretations of unknown words. Packed shape might be helpful to distinguish some frequent classes of unknown words –

named entities and numbers, in particular. Information about the fact that the word is known or not is also provided. It allows the guesser to exhibit a slightly different behavior on known and unknown words. We do not employ the standard observation – full orthographic form – on purpose. It would probably make the model more accurate with respect to known words, but our main goal is to obtain accurate estimations of prefix- and suffix-specific parameters.

Table 1 presents a sample sentence prepared for the subsequent processing with the CCRF guessing module. For each word the set of observations is determined on the basis of the observation schema described above. Observations are extended with additional prefixes identifying the kind of the particular observation. Every observation is a string, but from a technical point of view observations can be represented by more complex structures as long as their data type is totally ordered, which is the only assumption about the form of observations made on the level of the CCRFs implementation. The presented structure can be used to perform subsequent CCRF processing in the following way:

- The 'observations' column is translated to the input sequence **x**,
- The 'interpretations' column is converted to the sequence of CCRF constraints **r**,
- The contents of the 'choice' column (if any) can be used as the output sequence **y**.

All three columns can be used as the input to the CCRF training process. To perform the actual guessing only the observations and interpretations columns need to be known, of course.

word	guessing results
Szef	{ subst:sg:nom:m1 }
administracji	{ subst:sg:gen:f , subst:sg:dat:f , subst:sg:loc:f , subst:pl:gen:f }
Wołodymyr	{ subst:sg:nom:m2, subst:sg:nom:n, subst:sg:gen:m3 , subst:pl:nom:m1, subst:sg:nom:m3, subst:sg:nom:m1 , adj:sg:gen:m3:pos, subst:sg:loc:f, qub, brev:npun }
Łatwyn	{ subst:sg:gen:m3, subst:sg:nom:m2, subst:sg:nom:n , subst:sg:gen:m1, subst:sg:gen:f, subst:sg:nom:m3, subst:sg:acc:n , subst:sg:nom:m1, subst:pl:gen:m1, subst:sg:nom:f }

Table 2: Results of guessing interpretations of unknown words.

Table 2 shows the results of the guessing process performed on the phrase presented in table 1 with $k = 10$. We do not show the 'choice' column, since it is not used within the process, and the 'observations' column, which has been relevant only for input. Potential interpretations assigned to known words are not influenced by the guesser. The guessing has been performed using the model trained on the part which has not contained the phrase used as an example.

Since the guesser always proposes multiple morphosyntactic tags[6] for individual unknown words in the input, there must be also a lot of incorrect (w.r.t. the context) or even theoretically impossible (w.r.t. the word in question) interpretations in the output. The 'subst:sg:loc:f' tag is clearly not a possible interpretation of the masculine 'Wołodymyr' lexeme, but this fact is hard to determine without any reference to the word in the external dictionary. Therefore, to estimate the accuracy of the guesser we should only take the presence (or absence) of the correct tag

[6]Assuming that $k > 1$ and $|U| > 1$.

in the output set into account. As long as the correct tag is included in the output, it is of no particular concern to us what other output interpretations are. Looking at the 'guessing results' column in table 2 we can see that the guesser successfully determined that noun (substantive) is the most probable class for both unknown words, 'Wołodymyr' and 'Łatwyn', in the example. It has been also able to determine nominative case with a high confidence. The gender attribute is the biggest source of ambiguity in the output results – since the first-order model is used to perform the guessing, the method is not able to agree gender values of unknown words with the gender of the first word in the sentence, 'Szef', which is known to represent a human masculine (m1). Finally and most importantly, the correct full morphosyntactic tag (subst:sg:nom:m1) is included in the output for both unknown words.

4 Morphosyntactic Disambiguation

The last and the most important step of morphosyntactic tagging is a disambiguation phase. It involves choosing for each word, from the set of possible tags, the most appropriate one in the context of the particular sentence. It can be resolved without the prior morphosyntactic analysis or guessing, but these steps – as long as they demonstrate low false negative error rate – should only accelerate disambiguation (thanks to the reduced search space) and improve its accuracy.

Second-order stochastic methods are often chosen as a solution to the tagging problem, see (Collins, 2002; Smith et al., 2005), and we use the second-order flavor of constrained CRFs for disambiguation as well. The complexity of the task is no longer problematic, because sets of possible interpretations are strongly limited for individual words. If a word is known, the set of potential labels is provided by the morphosyntactic analyzer. Otherwise, potential labels are restricted to the set of the k most probable labels determined at the stage of morphosyntactic guessing described in the previous section, where k is the parameter of the guesser.

4.1 Model Adaptation

In case of the disambiguation problem we are more concerned about the performance of the solution in terms of quality than in terms of speed. We rely on the analysis and guessing tools in the second aspect and, in particular, on their ability to reduce the search space for the disambiguator. Therefore, we have chosen to implement a second-order variant of CCRF which incorporates a dense feature set (constraints from the training set are taken into account during the construction of the feature set) with no sparse forward-backward recursions optimization.

The second-order characteristic means that relations between three subsequent labels are modeled directly within the CRF. The sparse forward-backward recursions optimization has not been implemented in this case because it doesn't bring any improvements when the set of model features is dense. An important difficulty stems from the fact that we use the second-order model within the context of a language with more than 1000 potential morphosyntactic tags. It is easy to deduce that there are more than 10^9 possible transition features and, even with additional constraints taken into consideration, there are more than $1.5 * 10^7$ transition features which can be extracted from the NCP corpus directly. It is not only hard to store the model with so many features, but it can also cause problems related to data sparseness and make the model prone to overfitting.

To alleviate this problem we change our treatment of morphosyntactic tags as atomic labels and regard them as complex structures. We also extend our second-order CCRF to model multiple morphosyntactic layers. Tags are divided into separate label layers according to a configuration file. The file specifies the number of layers and the destination layers (zero, one or more) for each grammatical attribute (POS or grammatical category). Layers are modeled separately (i.e., features cover label values from exactly one layer), but not independently. Finally, labels in individual layers are treated as atomic entities.

This method is also rather flexible and could be used with respect to a language with other tagset structure than the positional one. The only important assumption which is really adopted here is that it is possible to divide a tag into parts which are relatively independent. For instance, in case of the partitioning which assigns case values to the first layer and number values to the second layer, implicit assumption is being made that – given observations – there is no direct dependency between those values apart from the dependency enforced by constraints. If we ignore constraint-related dependencies we can write it in a more formalized way: the case value of the i-th word is conditionally independent of its number value (and, in fact, all other number values in the sentence) given case values of words in positions $i-2$, $i-1$, $i+1$ and $i+2$ and observation values related to the i-th word.

4.2 Observations

In the case of disambiguation we use a richer set of observation types than within the context of guessing, yet it is still rather a minimalistic one. Let w_i denote the word at the i-th position in the input sentence. The following set of observations is used with respect to the i-th position:

- Lowercase orthographic forms of words w_{i-1}, w_i and w_{i+1},
- If the word w_i is unknown:
  - Lowercase prefixes and suffixes of length 1, 2 and 3 of w_i,
  - Packed shape of w_i and information, whether w_i is positioned at the beginning of the sentence, combined into one observation.

The shape observation type has been mentioned earlier within the context of the guesser's observation schema. We haven't recorded any improvements in the tagging quality when using prefix, suffix or shape features as observations for known words. Therefore, we do not use them in our final model.

4.3 Features

Let L be a number of layers and $y(l)$ denote a part of the morphosyntactic tag y assigned to the l-th layer for $l \in \{1, \ldots, L\}$. The layered model handles unigram (l, u, o) and transition (l, w, v, u) features in a similar way to that specified in equation 3. However, in the layered case features are enriched with value l identifying the layer to which the particular feature should be related. Moreover, the form of the feature function f_k changes due to the second-order character of the model.

$$f_k(x_i, y_i, y_{i-1}, y_{i-2}) = \begin{cases} \mathbf{1}(y_i(l) = u, o \in x_i) & k \text{ identifies } (l, u, o) \\ \mathbf{1}(y_i(l) = u, y_{i-1}(l) = v, y_{i-2}(l) = w) & k \text{ identifies } (l, w, v, u) \end{cases} \tag{13}$$

While the implementation is general and follows the specification described above, all tests within this work (most importantly, the evaluation) were performed using the configuration inspired by (Acedański, 2010). Tags are divided into two layers, with POS, case and person in the first layer and all the other grammatical categories in the second layer.

5 Related Work

We compare our method with solutions for morphosyntactic tagging of highly inflectional languages, the Polish language in particular. However, the method described in (Smith et al., 2005) – which bears many similarities with our solution – has been successfully used in tagging not only inflectional but also concatenative and templatic languages and we believe that our system could be also adapted to handle them.

Radziszewski (2013) describes WCRFT, an implementation of a tiered tagging method in which a cascade of independently trained linear-chain first-order CRF models is used to disambiguate input in a multi-pass manner: POS values are resolved at first, then case values, gender values and so on. Our system relies on a more general method of tag partitioning and performs disambiguation on all layers simultaneously, so that the choice made in higher layers can propagate down into lower layers, which is not possible in case of a cascade. Besides, our system uses second-order model for disambiguation which allows to capture dependencies between three subsequent tags in an idiomatic way. WCRFT uses the simple heuristic mentioned in section 3.1 to ascertain potential interpretations of unknown words and, optionally, a morphosyntactic guesser developed within a rule-based TaKIPI tagger (Piasecki, 2007; Piasecki and Radziszewski, 2007). It is an *a tergo* guesser, which uses pseudo-suffixes to infer potential morphosyntactic interpretations of unknown words. While it also proposes base forms in its output, it doesn't make use of context-specific information and often commits false negative errors (does not include correct morphosyntactic tags in its output). Since the disambiguation module in not designed so as to correct errors performed on the level of guessing, those false negative errors propagate to subsequent processing phases.

Pantera (Acedański, 2010) is an implementation of the Brill method (Brill, 1992) adapted for the specifics of highly inflectional languages. Pantera also adopts multi-pass method for morphosyntactic tagging without the possibility of downward propagation of disambiguation choices. Tag partitioning in Pantera is a configurable process and the evaluation of Pantera involved the same tag partitioning as the one used for the evaluation of our system: POS, case and person attributes are placed in the first layer, while the rest of grammatical categories is stored in the second layer (Acedański, 2010).

A variant of CRF, where each label must belong to a restricted set of interpretations recognized by a morphosyntactic analysis tool, has been previously used in tagging Korean, Arabic and Czech language (Smith et al., 2005). While the Czech model exhibits similarities to the model described in this paper, there are also important differences between them. In (Smith et al., 2005) there is no special treatment of unknown words. We, in comparison, introduce the context-sensitive morphosyntactic guessing method designed specifically for them. All grammatical attributes are taken into account in our model for Polish, while the model for Czech includes only the four main grammatical attributes – POS, case, number and gender – thus rendering it incapable of disambiguating between values of the remaining attributes. In order to make the training process feasible, Smith et al. (2005) employ factored training, dividing features into five separate classes and estimating parameters within each class independently. We do not adopt such parameter-level independence assumptions and perform global training using the Stochastic Gradient Descent method, yet we achieve an empirical performance which is several times better in terms of the training time. According to Smith et al. (2005), training the POS-specific part of the model for Czech on 768K words from the Czech PDT corpus takes up to two weeks on the 2GHz Pentium machine. Our system needs less than 8 hours to learn parameters for guessing (3 hours) and disambiguation (5 hours) on 1M words from the NCP corpus using one core of the 2.40GHz Xeon E5620 machine.

6 Evaluation

Evaluation of Concraft has been performed on the one-million-word, balanced NCP subcorpus (Przepiórkowski et al., 2010). In order to be able to compare the tool with other taggers available for Polish, we have followed guidelines provided by Radziszewski and Acedański (2012). It describes a fair method of taggers evaluation, which involves obligatory resegmentation (sentence splitting and tokenization) and reanalysis of the evaluation corpus part. The advantage of this evaluation method is that it measures tagging quality in an environment as close as possible to the real usage case, given the available resources. No artificial assumptions of perfect segmentation or perfect morphosyntactic analysis are made, which was often the case with earlier evaluations of tagging systems (Acedański, 2010; Radziszewski and Śniatowski, 2011) which reported much higher results than could be expected in the real-world usage.

Our tool does not include any sentence splitting, tokenization or morphosyntactic analysis modules yet. Results presented below have been acquired using the preprocessing functionality provided by the MACA tool (Radziszewski and Śniatowski, 2011). For each cross-validation fold, the guesser is trained on the reanalyzed training part. The resulting model is used to guess potential interpretations of unknown words both in the training and the evaluation part, with k (number of retained labels for each unknown word) set to 10. Next, the disambiguation tool is trained on the training part with tags already guessed. Finally, the model obtained is used to perform disambiguation on the evaluation part (processed beforehand by the guesser).

Tagger	Acc_{lower}	Acc_{upper}	Acc^{K}_{lower}	Acc^{U}_{lower}
Pantera	88.99%	89.28%	91.27%	14.74%
WMBT	89.71%	90.04%	91.20%	41.45%
WCRFT	90.34%	90.67%	91.89%	40.13%
Concraft	91.12%	91.44%	92.10%	59.19%

Table 1: Average accuracy measures obtained by individual taggers on the NCP corpus.

Table 1 presents cross-validation results of our system in comparison to the state-of-the-art taggers for Polish: Pantera (Acedański, 2010), WMBT (Radziszewski and Śniatowski, 2011) and WCRFT (Radziszewski, 2013). Pantera and WCRFT have been described in the 5 section. WMBT (Radziszewski and Śniatowski, 2011) and WCRFT (Radziszewski, 2013) both represent a class of tiered tagging methods (Tufiş, 1999), using a cascade of models to perform disambiguation. WMBT uses a memory-based learning method to model individual tiers, while WCRFT uses CRFs for this task. All tools have been evaluated on the same extract of the NCP corpus, and with respect to exactly the same corpus partitioning. Numbers included in the first three rows have been reported in (Radziszewski and Acedański, 2012) and (Radziszewski, 2013).

We report separate evaluation statistics corresponding to different assumptions about the system's behavior in the situation of tokenization-level errors. Accuracy lower bound Acc_{lower} corresponds to the situation when each tokenization error is penalized and treated as a tagging error. Conversely, accuracy upper bound Acc_{upper} doesn't take tokenization errors into account – the measure is computed as if for each token from the reference corpus associated with the incorrectly tokenized part the tagger made the correct decision. Therefore, assuming perfect tokenization, accuracy of the tagger would be somewhere between Acc_{lower} and Acc_{upper}. Next two statistics, Acc^K_{lower} and Acc^U_{lower}, show lower bound accuracy with respect to known and unknown words, respectively. Sentence splitting errors are not taken into account. Detailed description of individual statistics can be found in (Radziszewski and Acedański, 2012).

The results show that our system significantly outperforms other taggers, especially within the context of unknown words. They suggest that feature-rich stochastic methods perform better than transformation-based methods (represented by Pantera in the comparison). We suspect that the main reason of the gap between the WCRFT and Concraft performance is the difference in the choice of guessing methods. Our context-sensitive guessing method has been specifically designed as a preliminary filter, which reduces the number of possible interpretations of unknown words. The TaKIPI guesser used within the WCRFT tool during the evaluation yields too many false negatives, which cannot be corrected on the level of disambiguation.

Conclusion and perspectives

We have shown that a domain-specific modification of the CRFs formalism solves most of the complexity-related issues inherent to CRFs with respect to the morphosyntactic tagging of highly-inflectional languages. Not only is the modification easy to comprehend and implement, but it can be also used in combination with other optimizations designed for CRFs in general. We regard this as a confirmation that adapting the model to a particular problem domain should be the foremost optimization considered when designing a solution of a high quality.

The comparison of Concraft and WCRFT doesn't clearly show which tagging system would perform better if we used both of them in combination with the same guesser and this question should be further investigated. In particular, we plan to evaluate the guessing and the disambiguation components of the Concraft system separately. The current state of the disambiguation module still leaves some room for improvement on the level of tag partitioning and observation schema configuration. The CRF implementation can be further generalized by expanding the set of features with bigrams, trigrams or features capturing dependencies between individual tag layers (between parts of the same morphosyntactic tag for instance).

The guessing model can be used to perform guessing in a few reasonable ways. The first one, used within this work, is to output the k most probable labels from the U set for each unknown word in the sentence. Another option is to choose all labels with marginal probability higher than some arbitrarily chosen threshold. This solution has the advantage of making the size of the label set depend on probabilities of individual labels, so that very improbable labels do not get into the result even if they are in the set of the k most probable ones. Advantages of both solutions could be aggregated in a solution which uses both the marginal probability threshold and limits the size of the output set. The same techniques can be used to trim down not only label sets for unknown words, but also those which are assigned to known words. Advantage given by this solution would be a further speed up of computations on the level of the disambiguation phase, but it could potentially harm the tagging accuracy.

Finally, it would be worthwhile to confirm the usability of the tagger by testing its performance on another Slavic language and, afterwards, on a typologically different language (e.g. English, German or French) for which high-quality morphosyntactic analysis tools exist.

Acknowledgments

The release of the tagger described here is supported by the CESAR project (European project CIP ICT-PSP 271022, part of META-NET).

References

Acedański, S. (2010). A Morphosyntactic Brill Tagger for Inflectional Languages. In Loftsson, H., Rögnvaldsson, E., and Helgadóttir, S., editors, *Advances in Natural Language Processing*, volume 6233 of *Lecture Notes in Computer Science*, pages 3–14. Springer.

Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, ANLC '92, pages 152–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lavergne, T., Cappé, O., and Yvon, F. (2010). Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 504–513, Stroudsburg, PA, USA. Association for Computational Linguistics.

Piasecki, M. (2007). Polish Tagger TaKIPI: Rule Based Construction and Optimisation. *Task Quarterly*, 11(1–2):151–167.

Piasecki, M. and Radziszewski, A. (2007). Polish Morphological Guesser Based on a Statistical A Tergo Index. In *Proceedings of the International Multiconference on Computer Science and Information Technology — 2nd International Symposium Advances in Artificial Intelligence and Applications (AAIA'07)*, pages 247–256.

Przepiórkowski, A., Górski, R. L., Łaziński, M., and Pęzik, P. (2010). Recent Developments in the National Corpus of Polish. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., and Tapias, D., editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Radziszewski, A. (2013). A tiered CRF tagger for Polish. In R. Bembenik, Ł. Skonieczny, H. R. M. K. M. N., editor, *Intelligent Tools for Building a Scientific Information Platform: Advanced Architectures and Solutions*. Springer Verlag.

Radziszewski, A. and Acedański, S. (2012). Taggers gonna tag: an argument against evaluating disambiguation capacities of morphosyntactic taggers. In *Proceedings of TSD 2012*, LNCS. Springer-Verlag.

Radziszewski, A. and Śniatowski, T. (2011). Maca — a configurable tool to integrate Polish morphological data. In *Proceedings of the Second International Workshop on Free/Open-Source Rule-Based Machine Translation*.

Radziszewski, A. and Śniatowski, T. (2011). A Memory-Based Tagger for Polish. In *Proceedings of the LTC 2011*. Tagger available at http://nlp.pwr.wroc.pl/redmine/projects/wmbt/wiki/.

Smith, N. A., Smith, D. A., and Tromble, R. W. (2005). Context-based morphological disambiguation with random fields. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 475–482, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sokolovska, N., Lavergne, T., Cappé, O., and Yvon, F. (2009). Efficient Learning of Sparse Conditional Random Fields for Supervised Sequence Labelling. *CoRR*, abs/0909.1308.

Sutton, C. and Mccallum, A. (2006). *Introduction to Conditional Random Fields for Relational Learning*. MIT Press.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tufiş, D. (1999). Tiered Tagging and Combined Language Models Classifiers. pages 28–33.

Wainwright, M. J. and Jordan, M. I. (2008). *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., Hanover, MA, USA.