Intelligent Information Systems 9999
ISBN 666-666-666, pages 1–10

A morphosyntactic rule-based Brill tagger for Polish

Szymon Acedański and Konrad Gołuchowski

Faculty of Mathematics, Informatics and Mechanics,
Warsaw University, Poland

Abstract

In this paper we present and evaluate an implementation of a Brill morphosyntactic rule-based (Part of Speech) tagger adapted for specifics of inflectional languages like Polish. To achieve this, we use an optimized version of Brill's algorithm on morphologically analysed data. Then we perform the tagging in two phases, disambiguating part-of-speech, case and person at first, and other grammatical categories in the second phase.

Performed experiments show the tagging error rate of 10.8%, which is still worse than 9.6% error rate of the stochastic trigram tagger or 6.56% achieved by other Polish rule-based tagger called TaKIPI, which were used for tagging the largest morphosyntactically disambiguated Polish corpus. With such a high error rate we believe that the Brill tagger is not feasible for standalone use, but may be useful in hybrid setups.

Keywords: PoS tagger, Brill tagger, rule-based tagger, morphosyntactic analysis

1 Introduction

One of common problems in language processing is for each segment (usually a single word) in a given text to determine the flexemic class (roughly part of speech) and the values of several grammatical categories (e.g. number, case, etc.). This process is called morphosyntactic tagging. Part of Speech tagging for English is already well explored, however, few experiments are reported for inflectional languages like Hungarian (Megyesi, 1999), Czech (Hajic, 1997) or Polish (Łukasz Dębowski, 2004; Piasecki and Wardyński, 2006; Piasecki and Gaweł, 2005).

A tagger should assign a single *tag* to each segment in the text, at least where it is unambiguous. A tag describes the flexemic class of the segment and the values of matching grammatical categories. Possible tags for a particular language comprise its *tagset*. Sometimes it is impossible even for a human to unambiguously assign a tag to a segment, therefore real-world taggers can sometimes assign several tags to a single segment.

Several tagging techniques are commonly known. The most frequently used approaches are: stochastic, e.g. based on Hidden Markov Models (Jurafsky and Martin, 2008) and rule-based. In a paper (Brill, 1992) the author presents a rule-

Table 1: Example tags in Brown's English Tagset and IPI PAN Polish tagset

English	`VBD`	verb, past tense
	`PPS`	pronoun, personal, nominative, 3rd person singular
Polish	`praet:sg:m1:perf`	l-participle, singular, human masculine, perfective aspect
	`ppron12:sg:nom:f:pri`	1st person pronoun, singular, nominative, feminine

based Part of Speech tagger for English, which automatically chooses good quality rules given a number of general rule templates and a training corpus.

In this paper we describe and evaluate an implementation of the Brill's algorithm, adapted for rich inflectional languages with large tagsets like Polish. Polish is also a free word order language, what justifies the need for slightly expanding the original set of rule templates proposed in the Brill's original paper. This way it is possible to generate rules, which look up larger contexts, a desired feature for free word order languages. Brill uses Brown's Tagset for English, which consists of almost 200 tags (Brill, 1992), whereas IPI PAN Polish tagset (Przepiórkowski and Woliński, 2003; Przepiórkowski, 2005), which we use, contains more than 1000 tags. Examples of tags are shown in Table 1.

The tagger used for morphosyntactic disambiguation of the current version of the IPI PAN corpus (Przepiórkowski, 2004), called TaKIPI (Piasecki and Wardyński, 2006), is also a hybrid (multiclassifier) rule-based tagger. Some of the rules it uses were extracted automatically using sophisticated machine learning algorithms. Brill's approach is significantly simpler, though.

2 The original Brill tagger

Let us describe the original Brill's algorithm in some details. We assume that we are given three corpora — a large high-quality tagged training corpus, smaller tagged corpus called patch corpus and another one — test corpus, which we want to tag. Brill also assumes, that only one correct tag can be assigned to a segment. Tagging is performed in four steps:

1. A simple unigram tagger[1] is trained using the large training corpus.
2. Unigram tagger is used to tag the patch corpus.
3. There are certainly some errors in the tagging of the patch corpus. Therefore we want to generate rules, which will correct as many errors as possible. A rule consists of a *predicate* which specifies places where the rule should be applied and an *action* in the form "change the tag to **A** if the tag is **B**" (as we can see, and action may also contain a condition).

[1]Unigram tagger always assigns to a particular segment the most frequently seen tag for this segment (in the training corpus)

(a) We are given a small list of so called *rule templates*. Brill uses the following templates in his paper:
   i. change tag **A** to **B** if preceding (following) word has tag **C**,
   ii. change tag **A** to **B** if word two before (after) has tag **C**,
   iii. change tag **A** to **B** if one of the two preceding (following) words has tag **C**,
   iv. change tag **A** to **B** if one of the three preceding (following) words has tag **C**,
   v. change tag **A** to **B** if preceding word has tag **C** and following word has tag **D**,
   vi. change tag **A** to **B** if preceding word has tag **C** and word two before has tag **D**,
   vii. change tag **A** to **B** if following word has tag **C** and word two after has tag **D**,
   viii. change tag **A** to **B** if current (previous) word is capitalized.

(b) For each rule r, which can be generated using these templates, we compute two statistics:
   i. **good**(r) — number of places in the patch corpus, where the rule matches and changes an incorrect tag into a correct one,
   ii. **bad**(r) — number of places in the patch corpus, where the rule matches and changes the tagging from correct to incorrect.

(c) Now we find rule r_b, which maximizes **good**(r) − **bad**(r), i.e. reduces the largest possible number of errors when applied. We save the rule and apply it to the patch corpus. If the training corpus still contains many errors, return to 3a.

4. The test corpus is first tagged using the unigram tagger, and then the saved rules are applied in order.

If the test corpus was previously manually tagged, we can evaluate the performance of the tagger.

3 Adaptation for inflectional languages

Starting with this basic approach described above, we gradually applied a number of changes, as described in the following subsections.

3.1 Two-phase tagging

As mentioned in the introduction, Polish tagset is much bigger than English one. Therefore, the space of possible rules is also much bigger, so computations are far more time-consuming. Moreover, much more rules should be created to obtain good results. We did not expect good results using our 880 000-words corpus in which over 50% tags occur less than 100 times.

To solve this problem we considerably redesigned the process of tagging. At first we split each tag into two parts, called *P1-tag* and *P2-tag*. P1-tag contains

the flexemic class (roughly part of speech), case and person (where appropriate), whereas P2-tag contains the flexemic class once again, and all the appropriate grammatical categories except case and person i.e. number, gender, degree, aspect, negation, accentability, post-prepositionality, accommodability, aggultination, vocalicity. This way we obtained 97 P1-tags and 260 P2-tags. Piasecki and Wardyński (2006) also uses a similar approach, but splits the tags into three parts.

Table 2 shows examples of split tags.

Table 2: Examples of tags split into two parts

Original tag	P1-tag	P2-tag
`adj:sg:loc:f:pos`	`adj:loc`	`adj:sg:f:pos`
`fin:sg:ter:perf`	`fin:ter`	`fin:sg:perf`
`pact:sg:inst:f:imperf:aff`	`pact:inst`	`pact:sg:f:imperf:aff`
`ppas:pl:acc:f:imperf:aff`	`ppas:acc`	`ppas:pl:f:imperf:aff`
`subst:pl:nom:f`	`subst:nom`	`subst:pl:f`

Now we use the following tagging algorithm:

1. Use the original Brill's approach for P1-tags.
2. From now on we treat P1-tags as unchangeable, both in the patch corpus and in the test corpus. We will only try to choose a best matching P2-tag for each word.
3. Once again start by training an unigram tagger, so for each pair $(word, P1\text{-}tag)$ it assigns the most common P2-tag in the training corpus. If such a pair does not occur in the training corpus, assign any P2-tag, with a flexemic class, which matches the chosen P1-tag.
4. Tag the patch corpus using unigram tagger.
5. Using the patch corpus generate a set of good rules from given rule templates. Here we use a different set of templates than for assigning P1-tags. The following templates are used:
   (a) change P2-tag **A** to P2-tag **B** if the preceding (second before, following, second after) word has P1-tag **D**
   (b) change P2-tag **A** to P2-tag **B** if the preceding (second before, following, second after) word has P1-tag **D** and P2-tag **E**
   (c) change P2-tag **A** to P2-tag **B** if one of the two (three) preceding (following) words has P1-tag **D**
   (d) change P2-tag **A** to P2-tag **B** if one of the two (three) preceding (following) words has P2-tag **D**
   (e) change P2-tag **A** to P2-tag **B** if one of the two (three) preceding (following) words has P1-tag **D** and P2-tag **E**
   (f) similar rules as above, but only if the P1-tag of the current word is **C**

Also we say, that a rule matches only if the P2-tag, which it tries to assign to a segment, matches the already chosen P1-tag (namely the flexemic class is the same).

6. We tag the test corpus using the unigram tagger for P2-tags and then apply the generated rules.

3.2 Speeding up algorithm

The straightforward implementation of Brill's algorithm is slow, especially when dealing with large tagsets. Even after splitting each tag into two parts the rule space is so large that using the original Brill's algorithm is computationally unfeasible.

Ramshaw and Marcus (1994), and later (Ngai and Florian, 2001), propose computationally more efficient implementations of the original algorithm. We implemented the presented FastTBL algorithm for our experiments. The main differences between the original Brill's approach and FastTBL are:

1. FastTBL considers only rules which matches in at least one place of the patch corpus. This reduces the number of considered rules significantly.
2. FastTBL keeps values of $\mathbf{good}(r)$ and $\mathbf{bad}(r)$ for each rule r, and updates them through subsequent iterations, instead of recomputing them.

According to Ngai and Florian (2001) using FastTBL makes tagging process last over 300 times shorter than original Brill's algorithm. What is more, both algorithms give the same accuracy.

3.3 Morphological analyser for preprocessing

A common technique used for inflectional languages is to run a morphological analyser on the patch and test corpora. This initially reduces the space of considered tags for particular segments. For each segment in both the patch corpus and the test corpus, the morphological analyser assigns a number of candidate tags (based on its syntactic features only, not on the context).

For example, for word **podejrzaną** (a form of *suspected*) the morphological analyser decides, that only four tags need to be considered: `adj:sg:acc:f:pos`, `adj:sg:inst:f:pos`, `ppas:sg:acc:f:perf:aff` and `ppas:sg:inst:f:perf:aff`.

We use the information generated by the morphological analyser in the tagging algorithm itself. Namely, we consider that a rule matches only if it changes the tag of a segment to one of the possible tags assigned by the analyser. This way we also reduce the number of considered rules in the FastTBL algorithm by over an order of magnitude.

4 Experiments

For experiments we used the manually disambiguated fragment of the IPI PAN corpus of Polish language (Przepiórkowski, 2004). Its size is over 880 000 segments. For each segment, a set of possible tags is given with the corpus. Possible tags

were chosen using the morphological analyser Morfeusz (Woliński, 2006) and then correct tags were selected by hand by linguists.

After that we split the entire corpus into ten similar-sized parts. An experiment was performed ten times — each time a different part was used as the test corpus, and the other nine parts as both training and patch corpora. Reported values are averaged across these ten runs.

We repeated this process and varied the *threshold* (Thr) parameter, which describes the lowest accepted quality of chosen rules (the minimum accepted $\mathbf{good}(r) - \mathbf{bad}(r)$ value), after reaching which we stop producing rules. Lowering *threshold* gives more rules, but also increases the time needed for computations. The results of experiments are shown in Table 3.

Table 3: Testing corpus tagging results

Thr	Time (s)	# rules		P (%)	R (%)	F (%)	E (%)	E_{P1} (%)
		P1	P2					
6	1584.70	727	473	89.46	83.55	86.40	10.79	7.97
10	1018.32	472	303	89.08	83.20	86.04	11.19	8.26
20	626.99	262	175	88.45	82.61	85.43	11.90	8.77
50	362.02	124	94	87.61	81.82	84.62	12.87	9.44
100	238.74	71	53	86.72	80.99	83.76	13.95	10.15

It is worth noticing that lowering threshold to a value less than 6 does not result in significantly better tagging accuracies. We experimented with letting the rule generator stop the generator at threshold 5, but no of the extra rules matched the test corpus more than once, and most of the extra rules did not match at all.

Below, we describe the measures used for evaluation.

Time — the total run time, includes parsing of training corpus, training the unigram tagger and the rule-based tagger, tagging the test corpus and producing evaluation results. The experiments were run on a 2.66GHz Intel Xeon machine.

Precision (P) — percent of tags (given by the morphological analyser) marked by the tagger as correct which were also chosen by linguists as correct. Please note that in our IPI PAN corpus for some segments several tags are marked correct. Our tagger always assigns exactly one tag to a segment.

Recall (R) — percent of tags marked by linguists as correct which were marked by the tagger as correct.

F-measure (F) — $F = \frac{2PR}{P+R}$ — the measure commonly used in information retrieval.

Error rate (E) — percent of incorrectly tagged segments. For segments with multiple tags assigned by linguists, we arbitrarily choose one of them as correct, and compare it with tagger's choice.

Error rate of P1-tags (E_{P1}) — percent of incorrectly determined P1-tags.

The original Brill tagger for English reaches 4.5% error rate. The trigram stochastic tagger (Łukasz Dębowski, 2004) used for tagging the first version of the IPI PAN corpus of Polish reaches 9.6% error rate, so it is still significantly better than our tagger, which reaches 10.8%. Even better results (6.56% error rate) are reported for TaKIPI (Broda *et al.*, 2008) — a hybrid rule-based tagger used for tagging the current version of the mentioned corpus. For P1-tags only we get 8.0% error rate. Megyesi (1999) reports 14% error rate for basic version of Brill tagger for Hungarian, and improvements yield 8.1% (reaching even 5.5% for PoS only tags, which are very similar to our P1-tags). It is worth mentioning that Megyesi uses lexical rule templates, but no morphological analyser, for example

> Change the most likely tag (from tag **X**) to **Y** if the current segment has prefix/suffix x, $|x| < 7$.

Maybe adding rules similar to these would result in lower error rates for our tagger too, however it may not be the case, as the morphological analyser already takes into account the lexical structure of the current segment. Hajic (1997), who applies the original Brill's algorithm to Czech language, obtains 20.2% error rate, but he does not use morphological analyser. A wide range of possible improvements includes constructing hybrid taggers. Spoustová *et al.* (2007) compares a number of hybrid approaches for Czech, obtaining up to 4.3% error rate.

In order to find the potential sources of errors, we analysed per-tag results of P1 tagging. Any segments mistagged in P1 phase potentially strongly contributes to the overall error score. Per-tag results are presented in Table 4.

Table 4: Error rates for most common P1-tags

Tag	Count	E (%)	Tag	Count	E (%)
`interp`	14697	0.00	`adj:nom`	3042	20.23
`subst:gen`	8005	8.21	`subst:loc`	2894	4.69
`qub`	6448	3.61	`prep:gen`	2299	1.13
`subst:nom`	5795	21.74	`adj:acc`	1745	26.20
`conj`	5487	3.79	`prep:acc`	1727	3.89
`subst:acc`	4046	20.21	`subst:inst`	1721	3.66
`praet`	3763	0.33	`ign`	1456	0.97
`prep:loc`	3182	2.29	`adv:pos`	1437	1.84
`adj:gen`	3177	11.92	`inf`	1389	0.57
`fin:ter`	3120	1.43	`adj:loc`	1259	9.96

As we can see from the table, the most problematic for our tagger was to differentiate between `subst:nom` and `subst:acc` tags, as well as `adj:nom` and `adj:acc`. In Polish these two cases frequently have the same orthographical form, so they are not disambiguated by the morphological analyser, and also it is hard to determine them correctly by looking only on the small context, which is used by our rule templates. Hajic (1997) suggest joining these pairs of P1-tags into single P1-tags. We could then use more fine-grained P2 rules for disambiguating them. Another approach would be to design hand-made rules (or rule templates) covering most common cases. See also Piasecki (2006).

To get an insight into the type of rules actually generated during training, Table 5 shows chosen rules together with their **good**(r) and **bad**(r) values.

Table 5: Sample P1 rules generated by the tagger

No.	r	**good**(r)	**bad**(r)
1	Change `prep:gen` to `prep:inst` if any of the three following segments has tag `subst:inst`.	4055	105
3	Change `adj:gen` to `adj:loc` if any of the two preceding segments has tag `prep:loc`.	3710	770
5	Change `subst:gen` to `subst:loc` if it is preceded by `prep:loc`.	1814	13
60	Change `adj:gen` to `subst:gen` if it is preceded by `prep:gen` and followed by `interp`.	143	19
70	Change `num:nom` to `num:acc` if any of the two preceding segments has tag `qub`.	246	114

After carefully looking at these rules, we can see that first three rules are simple case matching rules. Sometimes a bit more sophisticated rules are found, e.g. the fourth one, which really has some linguistic sense, but it is not a very obvious rule. Unfortunately, as seen in the fifth example, sometimes the rules have no justification in the Polish grammar.

5 Conclusions and further work

In this paper we presented results of one of the first attempts to implement a Brill tagger for Polish language. Even though the obtained error rate of 10.8% is still high, we believe that addressing weaknesses like

- problems with differentiating between nominative and accusative forms,
- not using lexical information in rule templates,
- using local-context rule templates for a language with free word order,

may lower the error rate by a few percentage points. We implemented the presented algorithms in such a way, that adding and modifying rules is straightforward. Furthermore, high computational efficiency allows us to experiment with rule templates without need of dedicated computational resources. The tagger could be successfully run on any modern computer.

It would also be possible to adapt the rule system such that we allow multiple tags to be attached to a single segment, and the rules are used to attach or detach tags.

Another idea is to incorporate a Brill tagger as one of the classifiers in a hybrid tagger, like TaKIPI. It may further improve its accuracy, especially that the Brill tagger may generate correct tags in places, where other taggers frequently mistags segments. Of course, the inverse is true as well.

It is also the case, that taggers based on Brill's approach, when used in hybrid setups — in one stage of a linear topology hybrid tagger — may automatically identify and correct weaknesses specific for preceding taggers.

References

Eric Brill (1992), A simple rule-based part of speech tagger, in *Proceedings of the third conference on Applied natural language processing*, pp. 152–155, Association for Computational Linguistics, Morristown, NJ, USA.

Bartosz Broda, Maciej Piasecki, and Adam Radziszewski (2008), Towards a Set of General Purpose Morphosyntactic Tools for Polish, in *Intelligent Information Systems XVI*, pp. 441–450, Zakopane, Poland, ISBN 978-83-60434-44-4.

Jan Hajic (1997), Probabilistic and Rule-Based Tagger of an Inflective Language - a Comparison, in *In Proceedings of the 5th Conference on Applied Natural Language Processing*, pp. 111–118.

Daniel Jurafsky and James H. Martin (2008), *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*, Prentice Hall, second edition, ISBN 013122798X.

Beta Megyesi (1999), Improving Brill's POS tagger for an agglutinative language, in *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 275–284.

Grace Ngai and Radu Florian (2001), Transformation-based learning in the fast lane, in *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pp. 1–8, Association for Computational Linguistics, Morristown, NJ, USA.

Maciej Piasecki (2006), Hand-Written and Automatically Extracted Rules for Polish Tagger, in *Lecture Notes in Computer Science*, volume 4188, pp. 205–212, ISBN 3-540-39090-1.

Maciej Piasecki and Bartłomiej Gaweł (2005), A Rule-Based Tagger for Polish Based on Genetic Algorithm, in Mieczysław A. Kłopotek, Sławomir T. Wierzchoń, and Krzysztof Trojanowski, editors, *Intelligent Information Processing and Web Mining*, Advances in Soft Computing, pp. 247–258, Springer-Verlag, Berlin.

Maciej Piasecki and Adam Wardyński (2006), Multiclassifier Approach to Tagging of Polish, in *Proceedings of 1st International Symposium Advances in Artificial Intelligence and Applications*, unknown.

Adam Przepiórkowski (2004), *The IPI PAN Corpus: Preliminary version*, Institute of Computer Science, Polish Academy of Sciences, Warsaw.

Adam Przepiórkowski (2005), The IPI PAN Corpus in Numbers, in *Proceedings of the* 2nd Language & Technology Conference, Poznań, Poland.

Adam Przepiórkowski and Marcin Woliński (2003), A Flexemic Tagset for Polish, in *Proceedings of* Morphological Processing of Slavic Languages*, EACL 2003*.

Lance A. Ramshaw and Mitchell P. Marcus (1994), Exploring the Statistical Derivation of Transformational Rule Sequences for Part-of-Speech Tagging, in *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, pp. 00–00, Association for Computational Linguistics, New Mexico State University, Las Cruces, New Mexico, USA.

Drahomíra Spoustová, Jan Hajič, Jan Votrubec, Pavel Krbec, and Pavel Květoň (2007), The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for

Czech, in *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing*, pp. 67–74, Association for Computational Linguistics, Prague, Czech Republic.

Marcin Woliński (2006), Morfeusz — a Practical Tool for the Morphological Analysis of Polish, in *Intelligent Information Processing and Web Mining*, pp. 511–520.

Łukasz Dębowski (2004), Trigram morphosyntactic tagger for Polish, in *Intelligent Information Systems*, Advances in Soft Computing, pp. 409–413, Springer, ISBN 3-540-21331-7.