Reusable Tagset Conversion Using Tagset Drivers

Daniel Zeman
Univerzita Karlova
Ústav formální a aplikované lingvistiky
Malostranské náměstí 25
CZ-11800 Praha
E-mail: zeman@ufal.mff.cuni.cz

Abstract

Part-of-speech or morphological tags are important means of annotation in a vast number of corpora. However, different sets of tags are used in different corpora, even for the same language. Tagset conversion is difficult, and solutions tend to be tailored to a particular pair of tagsets. We propose a universal approach that makes the conversion tools reusable. We also provide an indirect evaluation in the context of a parsing task.

1. Introduction

Most annotated corpora use various types of tags to encode additional information on words. In some cases this information is merely the part of speech ("noun", "verb" etc.–hence the term *part-of-speech* or *POS tags*). In many cases, however, the string of characters comprising the tag is a compressed representation of a feature-value structure. Most of the features encoded this way are morphosyntactic (e.g. "gender = masculine", "number = singular"), hence the term *morphological tags*.

Unfortunately, it is very rare to see two corpora sharing a common set of tags. Language differences are only partially responsible—it is the corpus designers, their diverse views, theories and intended uses of the corpora, what matters most. Even two corpora of the same language may define two completely incompatible tagsets.

Such diversity proves disadvantageous for both human users and NLP software. A human user (linguist) typically wants to submit queries such as "show me all occurrences of a noun in plural, preceded by a preposition". Tags however rarely contain statements like "number = plural" literally. That would be prohibitively space-consuming. Instead we have to know that e.g. the fourth character of the tag being "P" means "plural". For instance, the tag `NNIS7-----A----`[1] may read as "part of speech = noun, detailed part of speech = common noun, gender = masculine inanimate, number = singular, case = 7th (instrumental), negativeness = affirmative". To work with the corpus efficiently, a linguist either needs to interpret the tags using specialized software, or to memorize the particular tag scheme. Obviously, if the same linguist has to switch to a different corpus, he/she must memorize more schemes or replace the tag interpretation software.

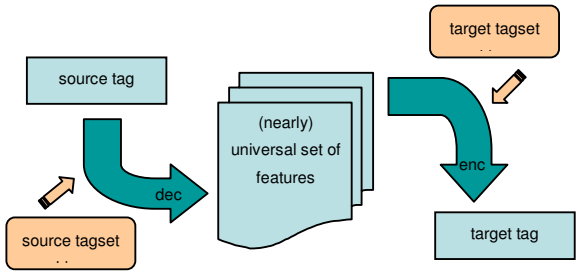
Similarly, various NLP tools may depend on particular tagsets. While some tools indeed treat tags as atomic strings, others could exploit the tag structure to dig more information about the word—no matter whether they use the features in machine learning, or in human-designed rules. If the tagset changes, manual rules become useless and statistical models have to be retrained at least; even that may not be possible in case the training procedure works with selected subsets of the feature pool. Applicability of NLP software to multiple corpora is exactly the reason why one would want to convert tags from one tagset to another.

For many tagset pairs, designing the conversion procedure is not easy. On one hand, there are rare tagsets (e.g. MULTEXT-EAST, Erjavec 2004) fitting at the same time languages as distant as Czech and Estonian; on the other hand, tagsets of two closely related languages (e.g. Danish and Swedish) or even two tagsets of the same language may differ substantially (for instance, the Mamba tagset of Swedish (Nivre et al. 2006) contains detailed classification of auxiliary verbs and punctuation but lacks features like number, mood, tense etc.; this is in sharp contrast to another Swedish tagset, Parole (Cinková and Pomikálek 2006), which in turn is not compatible with the Danish Parole (Kromann et al. 2004) tagset (the former classifies participles as verb forms, the latter as adjective forms; the former has separate tags for numerals, the latter classifies both cardinal and ordinal numbers as adjectives; etc.)

From the above said it follows that the typical tag conversion is an information-losing process. Though it is often desirable to perform it anyway and preserve as much information as possible. We have not been able to identify any previously published universal approach to do tagset conversion, which is not so surprising given the fact that for most part the conversion code must simply mimic the interpretation charts of the particular tagsets. We believe that most researchers solve the problem using specialized programs tailored to the two tagsets at hand. For subtly differing tagsets this may be the best thing to do; however, in all other cases, there is considerable effort put into analyzing the tag schemes, that cannot be reused for converting, say, the same source tagset into a new target tagset.

[1] This example is taken from the Prague Dependency Treebank (Böhmová et al. 2003)

In the present paper we propose an approach that makes the conversion code reusable. We define a (nearly) universal set of features and their values, and describe a way how *tagset drivers* can be used to convert various tagsets in and out of the universal feature set. In Section 2 we describe our universal set of features, in Section 3 we describe the encoding algorithm and the architecture of tagset drivers, in Section 4 we mention difficult phenomena and in Section 5 we present experiments.



2. Universal Set of Features

The key idea of our approach is to have a feature structure capable of storing all or most information from any tagset. The structure contains all features whose values are usually encoded in tags. The role of this universal set *("Interset")* is similar to the role of Interlingua in Interlingua-based machine translation (Richens 1958) or the role of Unicode among character sets. The Interset serves as an intermediate step on the way from tagset A to tagset B. The interaction between the Interset and tagsets A and B, respectively, is described in what we call *tagset drivers*. Once we write the drivers, we can do the two-way conversion A to B and B to A, plus the conversion between one of these tagsets and any other tagset that has been defined so far.

We are not likely to spare much time during the initial phase, if compared to just writing a targeted A-to-B conversion procedure. Actually, covering two completely new tagsets requires more work and care: we should describe both encoding and decoding of each tagset, we may have to think about features that are present in neither of them, and we will probably want to be more careful about aspects that may not matter to our current application. However, the reusability of the resulting code should compensate for the effort more than adequately. Plus we provide some algorithms to make adding new tagsets easier, and it is also possible that the required tagset has been covered by someone else who is sharing the code on the web.

Having analyzed about dozen tagsets, [2] we have identified the following features:

- part of speech
- various features for further details on part-of-speech: subpos, pronoun type, punctuation class and side (left vs. right bracket), syntactic part of speech, subcat
- yes/no features related to part of speech: possessive, reflexive, foreign, abbreviation, first part of hyphenated compound
- various inflectional features: gender, animateness, number, case, degree, definiteness, negativeness, person, politeness, possessor's gender and number, verbal form, mood, tense, subtense, aspect, voice
- the rest: style, variant, other, tagset

Although covering new tagsets may lead to adding new features to the central pool, it is desirable to find most of them in the very beginning. It is good to know what can be there when writing drivers. On the other hand, we do not intend to cram the Interset with hundreds of features, each of them specific to just one corpus. Some information in tags is really difficult to use out of the context of the original tagset. It is delicate to judge what belongs here; however, if there were a tag defined as "the word 'apple' occurring in a nested clause," we could probably live without that information saved. The only reason of saving really everything is that converting a tagset to itself should not lose information. For that purpose we use the "other" feature. It contains arbitrary information that does not fit in other features and distinguishes tags. Since the information is not understood by any other tagset, we need to know which tagset the value comes from. Thus the identifier of the tagset should be stored in the "tagset" feature.

Except for "tagset" and "other", there is a predefined list of possible values for each feature. Every feature also allows the empty value. While several feature-based tagsets distinguish between unknown values and irrelevant features, we do not find it wise in Interset. For instance, the fifth character in the PDT Czech tagset identifies grammatical case. Its normal values are `1` to `7`. For parts of speech that do not have case (e.g. interjections) the fifth character is `-` (dash). Adjectives generally do have case, yet there are borrowed words without Czech case suffixes whose case value is unknown (`X`). An example is the tag `AAIPX----1A----` for "Buenos" in *Buenos Aires*. The benefit of making this distinction explicit in a tagset is unclear. What is clear, however, is that we must not reflect it in the universal feature set. Who can say that a feature will be irrelevant—given the context of the values of the other features—in any tagset whatsoever? It is quite easy to find features that are relevant in one tagset and not the other: e.g. Czech past participles distinguish gender, English don't.

[2] Penn tagset of English, PDT tagset of Czech, STTS tagset of German, Mamba and Parole tagsets of Swedish, CoNLL tagsets of Arabic, Bulgarian, Chinese, Danish (and of Czech, English, German and Swedish; these four are however based on the other tagsets mentioned earlier).

3. Tagset Drivers

While the Interset is merely an abstract definition, the real implementation lies in the tagset drivers. A driver is a code library responsible for decoding and encoding tags. Decoding is reading a string (tag) into an internal data structure, in accordance with the list of possible features and their values. Encoding works the other way around.

The encoder obviously is the more difficult part. The decoder just reads and sorts the information, ideally not losing a single piece of it. If anything has to be discarded because it does not fit the target tagset, the discarding is encoder's task. There are two main reasons why encoding is not easy:

1. The encoder should be prepared to all values of all features, regardless that some of them are unknown in the particular tagset. For instance, if number = dual and the tagset does not know dual, it is probably better to encode plural than just leave number unknown.
2. Even if the target tagset knows features A and B, concrete value of A can restrict permitted values of B. Some *combinations* of feature values are not allowed. For instance, the Swedish Parole tagset allows "pos = noun & gender = common | neuter", and also "pos = pronoun & gender = masculine | feminine | common | neuter". If we are to encode "pos = noun & gender = masculine", we can either honor the part of speech, or the gender, but not both.

Fortunately enough, unknown feature values / combinations can be dealt with automatically if the driver has the list of all possible tags. By decoding all tags on the list, we get feature values for every tag. We thus know all feature values permitted in the given tagset and we know all value combinations. We have defined an ordered list of back-off values for every Interset feature value. The back-off lists contain all other values of the feature, including the empty value, so it is guaranteed that we always find a value that is permitted.[3] Of course, the encoder can override the default back-off list if necessary.

As for unknown feature combinations, there is a predefined total ordering of the features that defines their priority (this can be overridden, too). Since features are ordered, all value combinations can be stored in a trie structure. On selecting value of a higher-priority feature, the structure immediately reveals restricted value space for all lower-priority features.

This back-off technique is implemented in a helper module. Any driver can call it and have the features adjusted to something the driver itself might produce during decoding. The encoder can then concentrate on the driver's native feature combinations. Besides that, the helper module can also check a driver's integrity by looking whether the decoder only sets known features and values, whether encode(decode(x)) = x etc.

[3] The necessary condition is that the decoder only sets known feature values, which is desirable anyway.

The whole thing is implemented in Perl[4]. The drivers are Perl modules whose `encode` and `decode` functions can be called from other Perl programs, either to access the feature values, or to convert tagsets. The conversion script is very simple and looks like this:[5]

```
use tagset::cs::pdt;
use tagset::en::penn;
while(<>)
{
  print tagset::en::penn::encode
  tagset::cs::pdt::decode $_, "\n";
}
```

So far we have implemented and tested drivers for several tagsets of the CoNLL 2006 (Buchholz and Marsi 2006) and 2007 (Nivre et al. 2007) shared task treebanks, for the Penn Treebank (Marcus et al. 1993), the Prague Dependency Treebank (Böhmová et al. 2003) and others, totaling 14 drivers. Those drivers are freely available on the web.[6] We believe that the reusability will only be truly exploited if the drivers are shared in the community and we encourage everyone to contribute with drivers they need to write for themselves.

4. Difficult Phenomena

Working with various tagsets, we identified several fields that were difficult to capture and unify.

Endemic word classes[7] were one example. Whenever seen fit, we tried to roof them with some more common parts of speech, instead of introducing a new high-level class. We wanted to reduce the necessity of encoders' taking care of parts of speech unknown in their home tagsets. Roofed word classes are usually distinguishable by one of the detailed-part-of-speech features.

As mentioned earlier, some tagsets consider participles forms of verbs, others classify them as adjectives; some tagsets make numerals special cases of adjectives, others have separate POS tags for cardinals, ordinals and various other numeral classes, yet others separate cardinal numbers and put the rest under other POSes. Differences in approaches taken by different tagsets might result in different feature values; for instance, we could decode verbform = "participle" without regard to whether pos = "verb" or pos = "adj". Naturally it is desirable to decode the same thing into the same set of features each time. Although we could ban particular feature-value combinations in Interset, effectively forcing the driver authors to seek the permitted decoding, we prefer to leave it as a recommendation, since we do not want to predict, which feature combinations will never ever be needed to distinguish two different words. The recommending

[4] http://www.perl.org/

[5] Real conversion script would also have to deal with the format in which the tags are mixed with text in the corpus. This example merely assumes a list of tags, without the actual words and other annotation.

[6] https://wiki.ufal.ms.mff.cuni.cz/user:zeman:interset

[7] Examples include infinitival markers (English *to*), English existential *there*, Chinese classifiers before counted nouns etc.

guidelines (part of Interset documentation) are another output of our study.

Probably the broadest source of problems is pronouns, determiners and various WH-words. Somewhere pronouns are only personal or possessive; somewhere there is a diversity of interrogative, relative, demonstrative, indefinite and negative pronouns. In the BulTreeBank (Simov et al. 2004), anything interrogative is a pronoun, although it could be considered numeral *(how much?)* or adverb *(where? when? how?)* elsewhere. Some tagsets address the variable syntactic behavior of pronouns (*I* substitutes a noun, *my* substitutes an adjective). Some tagsets and languages do not have determiners but they have pronouns (demonstrative, indefinite) instead. All that lead us to remove pronouns and determiners as independent parts of speech. Instead, nouns, adjectives and adverbs have the feature "prontype" to distinguish the various types (personal, demonstrative, interrogative…) Empty value of this feature signals a normal noun (adjective, adverb).

Note however, that any guidelines are only to ensure unified approach to different presentations of the same information. It does not apply to information that simply is not there. If cardinals were tagged as *normal* adjectives (without sub-classing adjectives to numeral and others) they would remain so in Interset and also in the target tagset. We cannot add information, we only can lose it.

5. Experiments

At the time of writing, 14 drivers have been completed, with quite differing numbers of tags.[8] Some of the CoNLL tagsets are derived from other tagsets and share their properties (except for Czech, there is a one-to-one mapping between the original and the derived tagset; for Czech, the original PDT tagset is a subset of the CoNLL tagset). Table 1 shows an overview:

Tagset / Driver	**Number of tags**	**Approximate implementation time**
ar::conll	241	13 h
bg::conll	528	35 h
cs::conll	4854	6 h
cs::pdt	4288	18 h
da::conll	143	7 h
de::conll	54	10 min
de::stts	54	4 h
en::conll	45	45 min
en::penn	45	3 h
sv::conll	41	20 min
sv::hajic	156	*estimated 8 h*
sv::mamba	41	3 h
sv::svdahybrid	76	*estimated 2 h*
zh::conll	294	21 h

Table 1: Overview of tagset drivers.

[8] For some of the tagsets, the number of tags in the respective corpus has been counted; the true total of possible tags is probably higher.

The working times needed to design particular drivers differ greatly due to various reasons. The Czech tagsets are the most complex but they did not take the most time because the PDT tagset is the native environment for the author. On the other hand, Bulgarian was both complex and differing enough from Czech in approach to pronouns, necessity of introducing new verb tenses, definiteness values etc. Also, the CoNLL conversion of this and other tagsets is quite inconsistent and represents the same feature-value pair in different tags differently. The most exotic tagset w.r.t. this work is the Chinese (Chen and Hsieh 2004) one. Its nearly 300 tags encode mostly things that cannot be represented in Interset (e.g., there are more than 60 classes of prepositions, containing one to three words each). The intersection of the information encoded by the Chinese tagset with the other tagsets contains only about 10 basic parts of speech. Processing time of Chinese has been further extended because of poor documentation bundled with the CoNLL data.

The processing times are to be compared to time needed to accomplish a targeted conversion for a given tagset pair. Earlier experiments showed us that they are roughly comparable to writing a driver. (We were able to implement conversion from the Russian Dependency Treebank (Boguslavsky et al. 2000) to the Czech PDT tags in about 12 hours; Arabic tags by the Tim Buckwalter's morphological analyzer (Buckwalter 2002) took about 8 hours. However, drivers presented in Table 1 allow for 14 × 13 = 182 conversions, yielding less than 1 hour per conversion on average.

Table 2 illustrates the proportion of information that is shared by tagsets and can be preserved by the conversion. Note that even tagsets for the same language or closely related languages (Danish, Swedish) can be quite divergent due to different corpus designs.

	ar	bg	csc	csp	da	de	en	svh	svm	zh
ar	241	42	68	54	29	17	15	33	12	11
bg	65	528	104	94	64	32	25	50	15	11
csc	68	46	4854	4288	44	21	26	56	14	11
csp	66	42	4288	4288	42	20	24	54	13	11
da	25	46	55	54	143	24	24	71	14	11
de	14	16	17	16	17	54	20	18	15	10
en	16	17	28	26	22	20	45	28	17	11
svh	33	34	63	62	62	22	28	156	17	11
svm	14	15	15	14	15	17	17	16	41	10
zh	10	9	10	10	10	11	9	10	9	294

Table 2: Number of tags resulting from conversion from drivers named in row headers to drivers named in column headers.

As a practical application, driver-based tag conversion has been used in experiments with cross-language parser adaptation from Danish to Swedish (Zeman and Resnik 2008). We have used the reranking parser by (Charniak and Johnson 2005), originally written for the English Penn Treebank. Although the parser can be given a table of symbols from a new corpus, with Interset we could take

a much faster approach: we simply converted the Danish and Swedish data to the Penn Treebank format (including the POS tags), and made the parser think it was working with Penn data. Also, converting the divergent Danish and Swedish data to a common tagset was a crucial point in the adaptation technique itself.

Finally, we experimented with a dependency parser that is statistical in nature (Zeman 2004) and can learn dependencies of tags from any tagset; however it contains also many ad-hoc rules that bound it to the format of the Prague Dependency Treebank. The results of the experiments, shown in Table 3, reveal that tagset conversion help the parser better adapt to new corpora. Experiments have been conducted with the CoNLL data.

Lang	Year	P(orig)	P(conv)	McNemar
ar	2006	64.3	67.6	yes
ar	2007	59.8	66.9	yes
bg	2006	68.0	71.3	yes
cs	2006	56.1	71.4	yes
cs	2007	58.7	74.0	yes
da	2006	68.3	69.8	yes
en	2007	63.8	67.3	yes
sv	2006	71.0	73.5	yes
zh	2006	69.0	68.0	no
zh	2007	66.1	63.5	yes

Table 3: Accuracy of the parser on various CoNLL data sets, using original and converted tags. The last column indicates whether the change was statistically significant, using the McNemar's test with $p \leq 0.05$.

The decrease of accuracy for Chinese can be easily explained due to the large divergence of the Chinese tagset from the others: too much information gets lost during the conversion.

We are currently experimenting with other parsers (Malt parser, MST parser) as well; however, we do not expect significant improvements here, since these parsers are not so heavily dependent on one "home" treebank.

Conclusion

We have proposed a method for tagset conversion that is reusable and, to a reasonable extent, universal. Our interlingua-inspired approach enables to interpret part-of-speech and morphological tags in a uniform way, and to convert information that is shared by two tagsets. Besides the obvious advantage of being able to use tools that expect a particular tagset, we also observed improvements in performance of a statistical parser.

6. Acknowledgements

I would like to thank Philip Resnik for helpful comments and encouragement.

The research reported on in this paper has been supported by Grant No. N00014-01-1-0685 ONR. Ongoing research is supported by the Ministry of Education of the Czech Republic, project MSM0021620838, and Czech Academy of Sciences, project No. 1ET101470416.

7. References

Igor Boguslavsky, Svetlana Grigorieva, Nikolai Grigoriev, Leonid Kreidlin, Nadezhda Frid (2000): *Dependency Treebank for Russian: Concept, Tools, Types of Information.* In: Proceedings of COLING 2000. Saarbrücken, Germany.

Alena Böhmová, Jan Hajič, Eva Hajičová, Barbora Hladká (2003). *The Prague Dependency Treebank: A Three-Level Annotation Scenario.* In: Anne Abeillé (ed.): Treebanks: Building and Using Syntactically Annotated Corpora. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Sabine Buchholz, Erwin Marsi (2006). *CoNLL-X Shared Task on Multilingual Dependency Parsing.* In: Proceedings of the Tenth on Computational Natural Language Learning (CoNLL). New York, USA.

Tim Buckwalter (2002). *Buckwalter Arabic Morphological Analyzer Version 1.0.* Linguistic Data Consortium, LDC Catalog No. LDC2002L49, University of Pennsylvania, Philadelphia, USA.

Eugene Charniak, Mark Johnson (2005). *Coarse-to-Fine N-Best Parsing and MaxEnt Discriminative Reranking.* In: Proceedings of ACL, pp. 173–180. Ann Arbor, Michigan, USA.

Keh-Jiann Chen, Yu-Ming Hsieh (2004). *Chinese Treebanks and Grammar Extraction.* In: Proceedings of IJCNLP 2004, pp. 560–565. Hainan, China.

Silvie Cinková, Jan Pomikálek (2006). *LEMPAS: A Make-Do Lemmatizer for the Swedish PAROLE-Corpus.* In: Prague Bulletin of Mathematical Linguistics, vol. 86. Univerzita Karlova, Praha, Czechia.

Tomaž Erjavec (2004). *MULTEXT-East Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora.* In: Fourth International Conference on Language Resources and Evaluation (LREC 2004). Lisboa, Portugal.

Jan Hajič, Otakar Smrž, Petr Zemánek, Jan Šnaidauf, Emanuel Beška (2004): *Prague Arabic Dependency Treebank: Development in Data and Tools.* In: Proceedings of NEMLAR-2004, pp. 110–117.

Matthias T. Kromann, Line Mikkelsen, Stine Kern Lynge (2004). *Danish Dependency Treebank.* At http://www.id.cbs.dk/~mtk/treebank/. København, Denmark.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz (1993). *Building a Large Annotated Corpus of English: the Penn Treebank.* In: Computational Linguistics, vol. 19, pp. 313–330. USA.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, Deniz Yuret (2007). *The CoNLL 2007 Shared Task on Dependency Parsing.* In: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 915–932. Praha, Czechia.

Joakim Nivre, Jens Nilsson, Johan Hall (2006). *Talbanken05: A Swedish Treebank with Phrase Structure and Dependency Annotation.* In: Proceedings of the 5th International Conference on Language

Resources and Evaluation (LREC 2006). May 24–26. Genova, Italy.

Richard Hook Richens (1958). *Interlingual Machine Translation.* In: The Computer Journal 1958 1(3):144–147. British Computer Society, United Kingdom.

Kiril Simov, Petya Osenova, Milena Slavcheva (2004). *BTB-TR03: BulTreeBank Morphosyntactic Tagset.* BulTreeBank Project Technical Report No. 03. Sofija, Bulgaria.

Daniel Zeman (2004): *Parsing with a Statistical Dependency Model (Ph.D. thesis).* Univerzita Karlova, Praha, Czechia.

Daniel Zeman, Philip Resnik (2008): *Cross-Language Parser Adaptation between Related Languages.* In: Proceedings of IJCNLP. Hyderabad, India.