CroMo - Morphological Analysis for Standard Croatian and its Synchronic and Diachronic Dialects and Variants

Damir Ćavar[1/2], Ivo-Pavao Jazbec[2], and Tomislav Stojanov[2]

[1] University of Zadar
[2] Institute of Croatian Language and Linguistics

Abstract. We present the development strategies and results of the CroMo morphological analyzer for Croatian. It was designed as a monolithic finite state based morphological segmentation and annotation tool, that emits feature bundles for all recognized morphemes and sub-morphemes, and generates lemmata for the lexical root and complex compounds in one swoop. Its linguistic base uses morphological and morphotactic regularities only, and is easily extensible. The development and potential expansions of the lexical base is and can be done in a short amount of time, generating platform independent and extremely efficient code and binaries based exclusively on open source tools. We approach the (linguistic) interoperability problem utilizing a GOLD-based (Onltology-oriented) annotation schema for uniquely mappable linguistic terminology in the annotation output. CroMo was developed to provide an initial morphological and morpho-syntactic annotation and lemmatization for the *Croatian Language Corpus*, but can be applied to other similar languages.

1 Introduction

CroMo is a morphological analyzer that is being developed for automatic corpus annotation, in first place for the needs of the Croatian Language Corpus (CLC), but with a much broader scope. One of the goals of the CLC project[3] is to provide large quantities of language data with a multi-level annotation for lexicological, lexicographical, and linguistic research.

One particular interest of various related research projects that use CLC is to have access to morphological information, i.e. types of morphemes and allomorphs, their features and qualitative and quantitative distributional properties. The annotation requirements thus include morphological segmentation of each word in the corpus, and detailed annotation of each single morpheme at one layer. In addition to that, the ongoing expansion of the corpus with synchronic, and

* The presented work was supported by the Ministry of Science, Education and Sports of the Republic of Croatia, grant 212-2120920-0930.

[3] The CLC is developed at the Institute of Croatian Language and Linguistics. It is available for online analysis at riznica.ihjj.hr.

in particular diachronic data, motivated by research in the domain of language change and evolution, and thus the necessity to cope with various morphological paradigms, morpheme types, and morphotactic regularities that deviate from the normative standard, requires a technological solution that allows linguists to expand the morpheme set and morphotactic rules easily, and incorporate this in the automatic annotator.

Manual annotation at the morphological and morpho-syntactic layer of large quantities of text is not viable, given common timeframes and limited resources in research projects. Further, the specific goal of CLC to provide linguistically annotated text from a long period of time, and thus synchronic and diachronic information about Croatian and its dialects imposes a variety of problems. The variation and change of morphological regularities and morpho-syntax over longer timeframes, together with the fact that historically the respective orthographic standards were changing and even competing, is a challenge for a human annotator, let alone for automatic annotation tools.

A special interest was also to provide an annotation tool that minimizes the interoperability problem the linguistic community is facing. Besides the fact that corpora and lexical resources are often proprietary and the access to them is limited and restricted in various ways, also standards for annotation schema are manifold, often incompatible and partially inconsistent or linguistically problematic.

Besides general data format strategies to be based on XML and Unicode encodings, we propose and incorporate in CroMo also a linguistic annotation standard that maximizes the compatibility with other annotation schema and processing tools. It is a fact that many linguistic resources are designed with specific research goals in mind. In the same way, linguistic annotation tools are not necessarily designed to be compatible with others, or interoperable in other research and application scenarios, being based on less standardized linguistic and technological grounds.

Our general goal is to minimize the future effort for inclusion of resulting language resources and tools in a larger pool of data and tools. We intend to maximize the interoperability factor of data and tools by using a tagset or feature structure for the annotation of morphological segments that has the potential of being easily mappable or translated into other tagsets. Since we did not find any other existing annotation schema that satisfies these goals, we decided to make use of the General Ontology for Linguistic Description (GOLD) [1] as the exclusive terminological resource of annotation labels that enter the final description of the morphemes and words.

An additional advantages of using the terminology specified in an ontology for the annotation is also that post-processing the output can make use of axioms and concepts specified therein, either for disambiguation, or for further deduction of properties and structures.

A similar automatic morphological analyzer for Croatian, and in particular for the diachronic and synchronic dialects and variants, to our knowledge, does not exist.

2 Previous approaches

As far as Croatian is concerned, there are several documented attempts to develop a lemmatizer and a tagger for the standard language.

To our knowledge, the currently online accessible lemmatizer for Croatian described in [2] is based on a full-form word list with corresponding lemmata. From a technological perspective such an approach has various disatvantages. One is that morphotactic and phonotactic regularities are not exploited in rules, and unknown words or unseen derivations are rejected, and thus manual work is necessary for expansion and maintanance. Technically, a list-based approach requires significantly more memory than a compressed FSA, since usually full-forms and lemmata have to be stored as complete strings. The processing speed in simple applications (even when using hashing) with tables is significantly lower than FSA processing. Although hashing algorithms can help improving the access speed, they tend to be even more demanding with respect to persistent and runtime memory.

Other approaches related to lexical resources for lemmatization and morphological annotation of Croatian are documented in [3] and [4]. These approaches describe the current effort, and potential uses of building morphological dictionaries, and their potential for text processing applications.

To our knowledge, there are no publicly available tools or resources otherwise.

As far as FSAs for morphological analysis are concerned, we are aware of the fact that there are numerous tools, applications, FSA libraries and documentations available, both general and specific ones, and for all kinds of typologically different languages. Mentioning and discussing them all is beyond the scope of this paper. We mention the one that influenced our approach and design most, i.e. the TAGH architecture, as discussed e.g. in [5]. Although we did not yet make direct use of the Potsdam FSA library, we are grateful to the TAGH developers for extremely valuable advice and suggestions, while planing our implementation.

As far as the GOLD ontology-based terminological source for annotations in morphological analysis, but also in other linguistic annotation domains is concerned, we are not aware of any similar approach. However, we are grateful to authors of the following papers [6], [7], [1], [8], for for helpful hints and comments related to this idea.[4]

3 Architecture

The architecture of CroMo is specified as follows. The algorithm expects an input string and generates a list of possible morphemes with their byte offsets within the input string, and corresponding linguistic feature sets, as well as two types of lemma, a lemma generated from the morphological root, and a lemma

[4] We are grateful to Helen Aristar Dry and Anthony Aristar for comments and suggestions with respect to GOLD and its possible adaptation for improvement of common interoperability issues with natural language resources and processing tools.

generated from the morphological base of the word. The two lemmata are equal, if the word is neither morphologically derived, nor a compound of some kind.

In order to generate and compile a FSA as a recognizer with emission capabilities, we create lists of all possible morphemes, sets of morphotactic rules, and corresponding feature sets. For each morpheme recognized in the input string, a corresponding feature set is emitted. In cases of ambiguous morphemes a set of feature sets can be emitted. A morpheme is only recognized, and features are emitted, if it is not only a morpheme or allomorph as specified in some morpheme list, but if the necessary context is met, as specified in the morphotactic rules.

Formally we specify the algorithm as follows. A morpheme m is an ordered list of UTF8 encoded characters. Each linguistic feature f is defined as a binary value encoded in one bit. A feature vector F is an ordered list of bits (or a bit-vector). We extend the feature vector by using a portion of it to contain the byte offset for position at which the lemma suffix is appended, as well as the numerical ID of the lemma suffix as such, if applicable to the specific morpheme type.

An emission a is a tuple of an integer for the byte offset position p in the input string, and a feature vector F: $a = (p, F)$. The offset p is determined during runtime. F is specified prior to compilation in the morpheme specification and via code generation tools. A set of emissions A is empty or contains a finite number of different emissions a.

A morpheme set M is a set of tuples: (m, A). Thus each morpheme can be specified by 0 or more emission tuples. Each morpheme set is merged into a labeled morpheme set, a tuple (L, M, A), with L a unique label or name of the morpheme set, i.e. for each morpheme set M there is a corresponding unique label, and a set of emissions. Each morpheme set M is compiled into a non-recursive deterministic FSA. A trivial (and arbitrary) example of a morpheme definition is:

metl ((NOUN|ROOT|FEM, 0, 34))
boji ((VERB|ROOT|TRANS, 0, 21), (VERB|ROOT|TRANS, 1, 24))

The literal feature specification, together with the lemma offset and lemma ID is transformed into a bit-vector representation during compilation, and translated back to the literal representation during program output generation. This way, the emission of feature sets and other transformation instructions within the runtime system is encoded in a single 64-bit sequence (in our case), and thus contributes to the minimization of speed and size of the binary automaton.

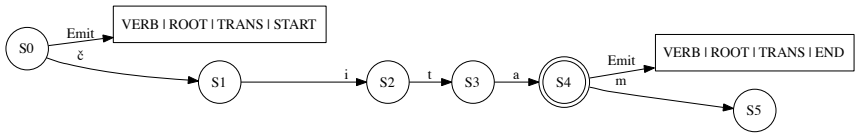
In the first experimental version, lemmata are generated by moving the negative offset of the identified root or base from the back to the front, and appending the lemma-sufix with the corresponding ID. For Croatian we chose the rightmost root morpheme to be the root of the root-lemma, while all morphemes without the word-final inflectional suffixes are used for generation of the base-lemma.

Morphotactic rules are combinations of concatenations, unions etc. of morpheme set labels using some variant of regular expression syntax, as specified for

example in Ragel. They can be recursive, if using e.g. the Kleene star operator. A trivial (and arbitrary) example of a morphotactic rule definition is in Ragel syntax:

verb1 = verb_pref_1 . verb_roots_2 . verb_suffixes_3

All morphotactic rules are united into one monolithic automaton, a FSA that emits emission tuples at the entry edge of each path that starts a morpheme or sub-morpheme, and also at the final state of such a path, as simplified in the following graph:



Entry and exit of a sub-path are marked with a special START/END bit in the feature vector F of each emission. Some resulting output for a random sample word in the current experimental version is:

metla
(metl) (0, 4, noun;feminine;root, metla, metla)
(a) (4, 6, noun;singular;nominative;suffix;inflectional)
(a) (4, 6, noun;plural;genitive;suffix;inflectional)

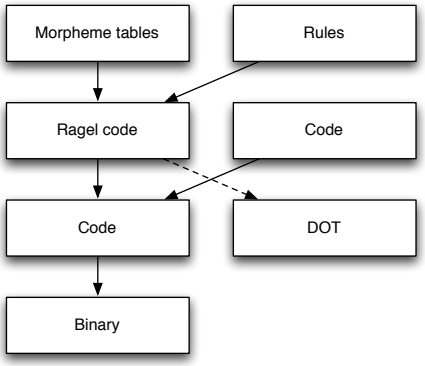
All morphemes that are compiled into the FSA, are grouped on the basis of being subject to the same type of rules, and having mostly equal emission features. We use exclusively morphotactic modeling, and exclude any modeling of two- or multi-level derivations. Phonological phenomena across morpheme boundaries for example are modeled as allomorphic variations, rather than phonological derivations that are triggered by morphological ones. This way we can reduce the complexity of rules, the resulting FSA, and the modeling effort for the linguist.

From the defined morphemes, feature sets and rules we generate grammar definitions as Ragel code. The choice to use Ragel [9] as the automaton compiler was based on our particular requirements, after evaluation of various common compiler compilers, lexers, and FSA tools, and after implementing our own testing algorithms. We require the possibility to associate potentially many emission with one transition in the automaton, to cope with ambiguity, but be able to keep the automaton deterministic. Ragel provides this capability since version 6.1.[5] Further requirements were met by Ragel as well, i.e. Unicode compatibility, availability on multiple platforms as open source, and the ability to generate extremely efficient code for various programming languages.

[5] We are grateful for Adrian Thurston's quick response and decision to significantly improve Ragel's usability for NLP by introducing this compiler flag.

The morphemes and their feature sets are edited in tables (e.g. OpenOffice Calc), and together with the word formation rule sets these are automatically compiled into Ragel code. The Ragel code is compiled into code of the target language, which is then, together with the wrapper classes in C++ and/or Java compiled into the target binary.

The following graph shows the general architecture and compilation process:



In this process, the lexicographers and linguists are only concerned with the definition of morphemes and their feature sets, as well as morphological rule definitions. Any change in the lexical basis can immediately enter CroMo via one compilation step.

The code frame is kept general, and can be applied to other languages with similar morphological patterns and properties (i.e. Slavic, and Indo-European in general). The components are not only based on open and free tools, they are open as well and freely available, thus encouraging application of the same development environment to other languages.

3.1 The lexical base

Our morpheme sets were derived from initial lexical resources that existed in various databases and Excel tables an the Institute of Croatian Language and Linguistics (IHJJ), and were made available to us. The databases were sorted on the basis of word classes, which were further sub-classified into morpheme classes and paradigm classes etc.

For the initial automaton we used a set of root morphemes for all necessary lexical classes, as described in table 1. Together with all possible allomorphs, inflectional and derivational suffixes, and (aspectual) prefixes, the morpheme-base currently contains more than 250,000 morphemes. However, it is continuously being extended. Due to the fact that we include word-formation rules that are recursive (e.g. aspectual prefixes in Croatian can be cascaded), our coverage is indefinite. This might be interpreted as over-generation. On the other hand, we cover this way potentially words that are extremely rare, or cannot be found in corpora, but are in principle well-formed.

Table 1. Root morpheme counts

Nouns	**76,649**	**Verbs**	**24,441**
feminine	36,767	perfective	12,284
masculine	35,374	imperfective	9,897
neuter	4,461	ambiguous	2,258
neuter/masculine	38		
Others			
adjectives	27,908	conjunctions	67
adverbs	7,672	particles	60
pronouns	129	interjections	390
prepositions	147	numbers	180

4 Evaluation

An evaluation of the analysis of words in terms of lexical coverage is pointless in our case. All morphemes and word-formation rules that are defined, are also recognized and annotated. If a word is not analyzed, it is being added to the lexical base. The current version still lacks many morphemes found in the CLC or other textual sources that must be dealt with. The final release will cover the complete lexical items found in our lexical resources.

Interesting questions are rather related to performance and memory requirements.

As for the pure compilation and regeneration process of the C++ code, we observe a memory requirement of up to 2,5 GB RAM using the GCC 4.2 on a Linux system. The resulting binary file size of the monolithic FSA is less than 4 MB. We do not expect it to grow dramatically with the continuous expansion of the morpheme base, as long as we stay in the current language environment.

The runtime behavior of the optimized binary, generating morphological segmentation and lemmatization, is optimal. On an Intel Quad-Core Xeon CPU at 1,86 GHz the automaton generates segmentations and lemmata for approx. 50,000 tokens in 1 second. The runtime behavior will not be affected dramatically neither, with the expansion of the morpheme base.

Since the analyzer is not disambiguating, a qualitative analysis of the best choice for the segmentation is pointless.

References

[1] Farrar, S.O.: An Ontology for Linguistics on the Semantic Web. PhD thesis, The University of Arizona, Tucson, Arizona (2003)

[2] Tadić, M.: Croatian lemmatization server. In Koeva, S., Dimitrova-Voulchanova, M., eds.: Proceedings of the 5th Formal approaches to South Slavic and Balkan languages Conference (FASSBL2006), Sofia (2006) 140–146

[3] Tadić, M., Fulgosi, S.: Building the croatian morphological lexicon. In: Proceedings of the EACL2003 Workshop on Morphological Processing of Slavic Languages. (2003) 41–46

[4] Šnajder, J., Bašić, B.D., Tadić, M.: Automatic acquisition of inflectional lexica for morphological normalisation. Information Processing and Management (0306-4573) (2008)
[5] Geyken, A., Hanneforth, T.: TAGH: A complete morphology for german based on weighted finite state automata. In Yli-Jyrä, A., Karttunen, L., Karhumäki, J., eds.: FSMNLP. Volume 4002 of Lecture Notes in Computer Science., Springer (September 2005) 55–66
[6] Farrar, S., Lewis, W.D.: The GOLD community of practice: An infrastructure for linguistic data and the web. In: Language Resources and Evaluation. (2006) to appear.
[7] Farrar, S.O., Langendoen, D.T.: A linguistic ontology for the semantic web. Glot International **7**(3) (March 2003) 1–4
[8] Farrar, S.O., Lewis, W.D., Langendoen, D.T.: A common ontology for linguistic concepts. In Ide, N., Welty, C., eds.: Semantic Web Meets Language Resources: Papers from the AAAI Workshop. AAAI Press, Menlo Park, CA (2002) 11–16
[9] Thurston, A.D.: Parsing computer languages with an automaton compiled from a single regular expression. In: 11th International Conference on Implementation and Application of Automata (CIAA 2006). Volume 4094 of Lecture Notes in Computer Science., Taipei, Taiwan (August 2006) 285–286
[10] Yergeau, F.: RFC 2279: UTF-8, a transformation format of ISO 10646 (January 1998) Obsoletes RFC2044 [11]. Status: PROPOSED STANDARD.
[11] Yergeau, F.: RFC 2044: UTF-8, a transformation format of Unicode and ISO 10646 (October 1996) Obsoleted by RFC2279 [10]. Status: INFORMATIONAL.