CroMo - Morphological Analysis for Croatian

Damir Ćavar[1], Ivo-Pavao Jazbec[2], Tomislav Stojanov[2]
1: Linguistics Department, University of Zadar
2: Institute of Croatian Language and Linguistics, Zagreb

Summary

CroMo is a FSA-based morphological segmentation, annotation, and lemmatization automaton, that allows for merging of three major functionalities into one highly efficient monolithic FSA, similar to TAGH (Geyken and Hanneforth, 2005). It is designed to be flexible, extensible, and applicable to any language that allows for purely morphotactic modeling at the lexical level. Its code-base is C and C++ for the final machine, and Python and Scheme for data-pre-processing and code-generation. Its language models are character code independent, processing any code-page or Unicode encoding schema.

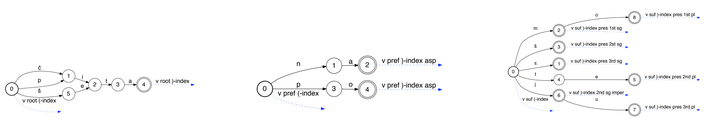
CroMo is designed as a Minimally Invasive system. Lexicologists and Linguists use *classical* and common computational data annotation and collection environments. CroMo generates code and binaries seamlessly, using common data formats (e.g. spreadsheets), minimizing the introduction of complex processes or learning of formal specification languages and digital formats for the developer or user.

CroMo uses an annotation schema (in an initial Croatian language model) that is directly mapped from the General Ontology for Linguistic Description (GOLD) (Farrar and Langendoen, 2003). The feature hierarchy and concept taxonomy in GOLD is mapped on an efficient and compact bit-vector representation, minimizing size and maximizing annotation performance. The ontology-based annotations increase the potential for interoperability, but also opens up advanced possibilities for a Description Logic-based post-processing and analysis.

FSA Architecture

The Finite State Automaton is formally a variant of a Mealy (Mealy, 1955) or a Moore machine (Black, 2004).

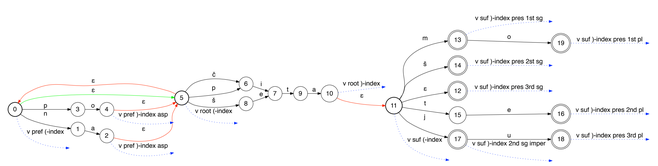
It is composed of non-cyclic Deterministic Finite State Automata (DFSA), as shown for example for verbal roots, prefixes and suffixes in the following graphs:



For each DFSA there is one emission associated with leaving its initial state, and one with reaching its final state. Each emission consists 1 to n feature sets, encoded in a binary bit-vector.

Ambiguity is mapped on multiple emissions, keeping the machine deterministic.

Rule-based composition of these DFSAs, based recursive rules for concatenation and alignment, can result in cyclic-automata:



General Architecture

The lexical basis for CroMo is prepared in tables and spreadsheets. Morphemes and allomorphs are grouped together on the basis of:

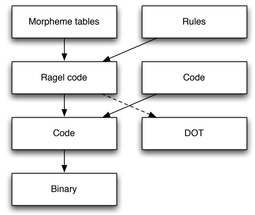
- shared feature bundles
- rules that compose complex FSAs

Each morpheme group represents one DFSA with an associated emission tuple of 1 to n feature bundles, represented as feature vectors in a long integer data type. Each morpheme group is associated with a variable name for rule definition.

Rules are defined in independent files, using the Ragel regular expression syntax with operators for optionality, recursion, concatenation etc., as for example:

verbAspectPrefixes? . verbAtiRoots . verbInflectionalSuffixes

The rules and morpheme sets are used to generate Ragel automata definition files. Ragel compiles the automata definitions into C-code, i.e. one monolithic automaton encoded in C-jump code, as well as a dot-representation for visualization in Graphviz. Predefined C++ code together with the Ragel generated C-code is compiled into the binary format, using current GCC versions. The output is a dynamic library and an executable binary.



The output is a tuple of a stem lemma, a root lemma, and tuples with:

- byte-offsets (begin and end indexes in the input string) for each single morpheme
- long integer based bit-vector encoding the morphological, morphosyntactic, semantic and other relevant features for each single morpheme, which can be mapped on the corresponding string representation such that the output would be a sequence of features represented as character strings

This architecture **does not**:

- transcode the character mappings, i.e. it is completely independent of the character code pages,
- restrict itself to specific feature definitions or tag-sets, i.e. it is open and extensible, allowing for extension of the encoding bit-vector for up to 128 bits,
- include real probabilities for transitions and emissions, due to a lack of training data, but it also doesn't prevent inclusion of such probabilities,
- include disambiguation or relevance metric for the generated output.

This architecture **does**:

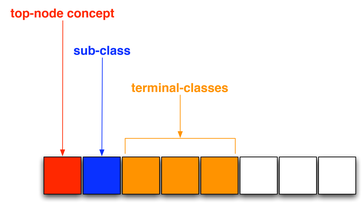
- allow for arbitrary individual feature definitions that can be mapped on bit-vectors,
- allow for the extension of the morphological base and rule set without having to touch the program code, and with a turnaround time of 3 to 5 minutes (recompilation),

General Concept

Research and applications using Natural Language Processing tools and linguistic data is facing a serious problem:

- The lack of interoperability, i.e. annotation compatibility and format, in particular on the linguistic level.

CroMo uses an annotation schema based on GOLD. The taxonomy of concepts and properties defined in GOLD, in particular the ones related to morphology, morpho-syntax, and semantic are mapped on a bit-vector for efficiency, maintaining feature inheritance and sub-relations, i.e. incorporating in some sense implicatures.



GOLD can be used as an interlingua, i.e. it can in principle be translated to various tag-sets and annotation schema. Given the fact that GOLD is embedded in a development process and provides explanations for concepts and features, annotation consistency for linguistic annotators is easier to maintain.

The corresponding C++ code for encoding and decoding of the bit-vector representation is automatically generated from the morpheme annotations. In addition to the GOLD-based features, CroMo accepts user-defined features, that are mapped on bit-vectors in the same way. These features can be added any time during the development process, and they do not imply changes of the programs code base.

This approach in CroMo allows for flexibility for the lexicon developers, maximizing interoperability, and maintaining high performance and efficiency of the resulting analyzer.

Comments and Technical Details

CroMo is highly efficient, segmenting and annotating approx. 50,000 tokens per second. It is highly flexible, allowing for dynamic extension of the morpheme base and annotation schema, while minimizing or avoiding changes and adaptations in the code base. In addition to that, CroMo is highly efficient with respect to persistent and runtime memory. The binary size that includes more than 120,000 morphemes of Croatian is approx. 5 MB for a Mac OS X or Ubuntu AMD64 Linux.

It is also open source, coded in a platform independent way, relying on open source and free components (GCC, Ragel, Python, Scheme). The code and documentation, with detailed evaluation results is available at:

http://personal.unizd.hr/ dcavar/CroMo/

The presented work is part of a research project supported by the Ministry of Science, Education and Sports of the Republic of Croatia, grant 212-2120920-0930.