Statistical Models for Term Compression

James Cheney
Cornell University
jcheney@cs.cornell.edu

Computing systems frequently deal with symbolic tree data structures, which are also known as terms in universal algebra and logic. For example, arithmetic expressions such as $2x^2 - 3x + 1$ can be represented as terms. Other common term systems include machine code and compiler intermediate representations, parse trees, mathematical formulas and proofs, and structured documents in markup languages like SGML and HTML. Our goal is to develop universal, effective, and efficient term compression techniques, superior to specialized or universal compression techniques currently available. Our approach is to use knowledge of term structure to build accurate universal statistical models of terms. These models can compress terms faster or more effectively than comparable sequential methods.

We present two statistical term models that are related to Markov random fields over trees. These models gather statistical information about parent-child symbol relationships in terms. Huffman or arithmetic codes generated from these probability estimates are used to encode the terms. In the first model, a symbol's value is predicted by value of its parent symbol alone. Thus, in compressing a subterm of the form $t + u$, the $+$ operator would be used to select a specialized code for the root symbols of subterms t and u. The second model also uses the symbol's argument position as a predictor. For example, in compressing $t + u$, we would make different probability estimates for the first and second arguments, and use one code to encode the first children of $+$ and another code for the second children. It might be the case that $t + 1$ (but not $1 + t$) occurs frequently for many different terms t, in which case we could give 1 a shorter code as the second child of $+$.

We believe that these models are general enough to apply to many different term languages and simple enough that they can be implemented efficiently. In our experiments, we found that these methods compress terms better or faster than similar "one-dimensional" zero or first-order models. The first and second term models obtained 4.0 and 3.4 bits per symbol vs. 6.0 and 4.3 bps for the sequential zero and first order models, respectively. By comparison, `gzip` obtained 2.7 bps, and `gzip` is, of course, much faster than our unoptimized implementation. We have not achieved our goal of improved term compression, but we believe that more sophisticated and more effective techniques remain to be investigated. For example, one improvement would be to make a term version of PPM in which contexts are ancestor symbols in the term rather than predecessors in a sequence. Our implementation could be viewed as a first step in that direction.