An Overview of Probabilistic Tree Transducers for Natural Language Processing

Kevin Knight and Jonathan Graehl

Information Sciences Institute (ISI) and Computer Science Department
University of Southern California
knight@isi.edu, graehl@isi.edu

Abstract. Probabilistic finite-state string transducers (FSTs) are extremely popular in natural language processing, due to powerful generic methods for applying, composing, and learning them. Unfortunately, FSTs are not a good fit for much of the current work on probabilistic modeling for machine translation, summarization, paraphrasing, and language modeling. These methods operate directly on trees, rather than strings. We show that tree acceptors and tree transducers subsume most of this work, and we discuss algorithms for realizing the same benefits found in probabilistic string transduction.

1 Strings

Many natural language problems have been successfully attacked with finite-state machines. It has been possible to break down very complex problems, both conceptually and literally, into cascades of simpler probabilistic **finite-state transducers** (FSTs). These transducers are bidirectional, and they can be trained on sample input/output string data. By adding a probabilistic **finite-state acceptor** (FSAs) language model to one end of the cascade, we can implement probabilistic **noisy-channel** models. [1] Figure 1 shows a cascade of FSAs and FSTs for the problem of transliterating names and technical terms across languages with different sounds and writing systems [1].

The finite-state framework is popular because it offers powerful, generic operations for statistical reasoning and learning. There are standard algorithms for:

- **intersection** of FSAs
- **forward application** of strings and FSAs through FSTs
- **backward application** of strings and FSAs through FSTs
- **composition** of FSTs
- **k-best** path extraction
- supervised and unsupervised **training** of FST transition probabilities from data

[1] In the noisy-channel framework, we look for the output string that maximizes P(output | input), which is equivalent (by Bayes Rule) to maximizing P(output) · P(input | output). The first term of the product is often captured by a probabilistic FSA, the second term by a probabilistic FST (or a cascade of them).