NAACL HLT 2009

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics

Tutorial Abstracts

Ciprian Chelba, Paul Kantor, Brian Roark
Tutorial Chairs

May 31, 2009
Boulder, Colorado

Table of Contents

Data-Intensive Text Processing with MapReduce

Jimmy Lin and Chris Dyer
University of Maryland, College Park
{jimmylin,redpony}@umd.edu

Overview

This half-day tutorial introduces participants to data-intensive text processing with the MapReduce programming model [1], using the open-source Hadoop implementation. The focus will be on scalability and the tradeoffs associated with distributed processing of large datasets. Content will include general discussions about algorithm design, presentation of illustrative algorithms, case studies in HLT applications, as well as practical advice in writing Hadoop programs and running Hadoop clusters.

Amazon has generously agreed to provide each participant with $100 in Amazon Web Services (AWS) credits that can used toward its Elastic Compute Cloud (EC2) "utility computing" service (sufficient for 1000 instance-hours). EC2 allows anyone to rapidly provision Hadoop clusters "on the fly" without upfront hardware investments, and provides a low-cost vehicle for exploring Hadoop.

Intended Audience

The tutorial is targeted at any NLP researcher interested in data-intensive processing and scalability issues in general. No background in parallel or distributed computing is necessary, but a prior knowledge of HLT is assumed.

Course Objectives

- Acquire understanding of the MapReduce programming model and how it relates to alternative approaches to concurrent programming.
- Acquire understanding of how data-intensive HLT problems (e.g., text retrieval, iterative optimization problems, etc.) can be solved using MapReduce.
- Acquire understanding of the tradeoffs involved in designing MapReduce algorithms and awareness of associated engineering issues.

Tutorial Topics

The following lists topics that will be covered:

- MapReduce algorithm design
- Distributed counting applications (e.g., relative frequency estimation)
- Applications to text retrieval
- Applications to graph algorithms
- Applications to iterative optimization algorithms (e.g., EM)
- Practical Hadoop issues
- Limitations of MapReduce

Instructor Bios

Jimmy Lin is an assistant professor in the iSchool at the University of Maryland, College Park. He joined the faculty in 2004 after completing his Ph.D. in Electrical Engineering and Computer Science at MIT. Dr. Lin's research interests lie at the intersection of natural language processing and information retrieval.

He leads the University of Maryland's effort in the Google/IBM Academic Cloud Computing Initiative. Dr. Lin has taught two semester-long Hadoop courses [2] and has given numerous talks about MapReduce to a wide audience.

Chris Dyer is a Ph.D. student at the University of Maryland, College Park, in the Department of Linguistics. His current research interests include statistical machine translation, machine learning, and the relationship between artificial language processing systems and the human linguistic processing system. He has served on program committees for AMTA, ACL, COLING, EACL, EMNLP, NAACL, ISWLT, and the ACL Workshops on Machine translation, and is one of the developers of the Moses open source machine translation toolkit. He has practical experience solving NLP problems with both the Hadoop MapReduce framework and Google's MapReduce implementation, which was made possible by an internship with Google Research in 2008.

Acknowledgments

This work is supported by NSF under awards IIS-0705832 and IIS-0836560; the Intramural Research Program of the NIH, National Library of Medicine; DARPA/IPTO Contract No. HR0011-06-2-0001 under the GALE program. Any opinions, findings, conclusions, or recommendations expressed here are the instructors' and do not necessarily reflect those of the sponsors. We are grateful to Amazon for its support of tutorial participants.

References

[1] Dean, Jeffrey and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI 2004), p. 137-150, 2004, San Francisco, California.

[2] Jimmy Lin. Exploring Large-Data Issues in the Curriculum: A Case Study with MapReduce. Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics (TeachCL-08) at ACL 2008, p. 54-61, 2008, Columbus, Ohio.

Distributed Language Models

Thorsten Brants and Peng Xu, Google Inc.

Language models are used in a wide variety of natural language applications, including machine translation, speech recognition, spelling correction, optical character recognition, etc. Recent studies have shown that more data is better data, and bigger language models are better language models: the authors found nearly constant machine translation improvements with each doubling of the training data size even at 2 trillion tokens (resulting in 400 billion n-grams). Training and using such large models is a challenge. This tutorial shows efficient methods for distributed training of large language models based on the MapReduce computing model. We also show efficient ways of using distributed models in which requesting individual n-grams is expensive because they require communication between different machines.

Tutorial Outline

1) Training Distributed Models

* N-gram collection
Use of the MapReduce model; compressing intermediate data; minimizing communication overhead with good sharding functions.

* Smoothing
Challenges of Katz Backoff and Kneser-Ney Smoothing in a distributed system; Smoothing techniques that are easy to compute in a distributed system: Stupid Backoff, Linear Interpolation; minimizing communication by sharding and aggregation.

2) Model Size Reduction

* Pruning
Reducing the size of the model by removing n-grams that don't have much impact. Entropy pruning is simple to compute for Stupid Backoff, requires some effort for Katz and Kneser-Ney in a distributed system. Effects of extreme pruning.

* Quantization
Reducing the memory size of the model by storing approximations of the values. We discuss several quantizers; typically 4 to 8 bits are sufficient to store a floating point value.

* Randomized Data Structures
Reducing the memory size of the model by changing the set of n-grams that is stored. This typically lets us store models in 3 bytes per n-gram, independent of the n-gram

order without significant impact on quality. At the same time it provides very fast access to the n-grams.

3) Using Distributed Models

* Serving
Requesting a single n-gram in a distributed setup is expensive because it requires communication between machines. We show how to use a distributed language model in the first-pass of a decoder by batching up n-gram request.

Target Audience

Target audience are researchers in all areas that focus on or use large n-gram language models.

Presenters

Thorsten Brants received his Ph.D. in 1999 at the Saarland University, Germany, on part-of-speech tagging and parsing. From 2000 to 2003, he worked at the Palo Alto Research Center (PARC) on statistical methods for topic and event detection. Thorsten is now a Research Scientist at Google working on large, distributed language models with focus on applications in machine translation. Other research interests include information retrieval, named entity detection, and speech recognition.

Peng Xu joined Google as a Research Scientist shortly after getting a Ph.D. in April 2005 from the Johns Hopkins University. While his research is focused on statistical machine translation at Google, he is also interested in statistical machine learning, information retrieval, and speech recognition.

Dynamic Programming-based Search Algorithms in NLP

Liang Huang, Google Research

Dynamic Programming (DP) is an important class of algorithms widely used in many areas of speech and language processing. It provides efficient solutions to seemingly intractable inference over exponentially-large spaces by sharing overlapping subproblems. Well-known examples of DP in our field include Viterbi and Forward-Backward Algorithms for finite-state models, CKY and Earley Algorithms for context-free parsing, and A* Algorithm for both. These algorithms are widely used to solve problems ranging from sequence labeling to word alignment to machine translation decoding.

With this overwhelming popularity, this tutorial aims to provide a better understanding of DP from both theoretical and practical perspectives. In the theory part, we try to unify various DP algorithms under a generic algebraic framework, where the above mentioned examples are merely special cases, and we can easily analyze their correctness and complexities. However, exact DP algorithms are often infeasible in practice due to time and space constraints. So in the practice part, we will survey several widely used tricks to reduce the size of the search space, including beam search, histogram pruning, coarse-to-fine search, and cube pruning. We will discuss these methods within the context of state-of-the-art large-scale NLP systems.

1 Outline

- Part A: Dynamic Programming on Lattices/Graphs under Semiring Framework
  - theory:
    - Motivations and Examples
    - Semirings
    - Viterbi Algorithm
    - Dijkstra and A* Algorithms
    - Comparison between Viterbi and Dijkstra/A* Algorithms
  - practice:
    - Beam Search and Histogram Pruning; E.g.: *Pharaoh* (phrase-based MT)

- Part B: Dynamic in Programming on Packed Forests under Hypergraph Framework
  - theory:

    * Hypergraphs; Examples in Parsing and Machine Translation
    * Generalized Viterbi Algorithm; CKY Parsing
    * Knuth and A* Algorithms
  - practice:
    * A* in Practice: A* Parsing; beam A*; inadmissible heuristics
    * Coarse-to-Fine Search; Example: *Charniak and Berkeley parsers*
    * Cube Pruning; Example: *Hiero* (syntax-based decoding)

2 Target Audience

This tutorial is intended for researchers with *any* level of familiarity with dynamic programming. A basic understanding of the CKY Algorithm is recommended, but not required.

3 Brief Bio of the Presenter

Liang Huang is a Research Scientist at Google Research (Mountain View). He recently obtained his PhD in 2008 from the University of Pennsylvania under Aravind Joshi and Kevin Knight (USC/ISI). His research interests include algorithms in parsing and translation, generic dynamic programming, and syntax-based machine translation. His work on "forest-based algorithms" received an Outstanding Paper Award at ACL 2008, as well as Best Paper Nominations at ACL 2007 and EMNLP 2008. He also loves teaching and was a recipient of the University Teaching Prize at Penn.

Extracting World and Linguistic Knowledge from Wikipedia

Simone Paolo Ponzetto
Dept. of Computational Linguistics
University of Heidelberg
Heidelberg, Germany
`http://www.cl.uni-heidelberg.de/~ponzetto`

Michael Strube
EML Research gGmbH
Schloss-Wolfsbrunnenweg 33
Heidelberg, Germany
`http://www.eml-research.de/~strube`

Overview

Many research efforts have been devoted to develop robust statistical modeling techniques for many NLP tasks. Our field is now moving towards more complex tasks (e.g. RTE, QA), which require to complement these methods with a semantically rich representation based on world and linguistic knowledge (i.e. annotated linguistic data). In this tutorial we show several approaches to extract this knowledge from Wikipedia. This resource has attracted the attention of much work in the AI community, mainly because it provides semi-structured information and a large amount of manual annotations. The purpose of this tutorial is to introduce Wikipedia as a resource to the NLP community and to provide an introduction for NLP researchers both from a scientific and a practical (i.e. data acquisition and processing issues) perspective.

Outline

The tutorial is divided into three main parts:

1. **Extracting world knowledge from Wikipedia**. We review methods aiming at extracting fully structured world knowledge from the content of the online encyclopedia. We show how to take categories, hyperlinks and infoboxes as building blocks for a semantic network with unlabeled relations between the concepts. The task of taxonomy induction then boils down to labeling the relations between these concepts, e.g. with isa, part-of, instance-of, located-in, etc. relations.

2. **Leveraging linguistic knowledge from Wikipedia**. Wikipedia provides shallow markup annotations which can be interpreted as manual annotations of linguistic phenomena. These ‘annotations’ include word boundaries, word senses, named entities, translations of concepts in many languages. Furthermore, Wikipedia can be used as a multilingual comparable corpus.

3. **Future directions**. Knowledge derived from Wikipedia has the potential to become a resource as important for NLP as WordNet. Also the Wikipedia edit history provides a repository of linguistic knowledge which is to be exploited. Potential applications of the knowledge implicitly encoded in the edit history include spelling corrections, natural language generation, text summarization, etc.

Target audience

This tutorial is designed for students and researchers in Computer Science and Computational Linguistics. No prior knowledge of information extraction topics is assumed.

Speakers' bios

Simone Paolo Ponzetto is an assistant professor at the Computational Linguistics Department of the University of Heidelberg, Germany. His main research interests lie in the area of information extraction, knowledge acquisition and engineering, lexical semantics, and their application to discourse-based phenomena.

Michael Strube is group leader of the NLP group at EML Research, a privately funded research institute in Heidelberg, Germany. The NLP group focuses on the areas of semantics, pragmatics and discourse and applications like summarization and information extraction.

OpenFst: An Open-Source, Weighted Finite-State Transducer Library and its Applications to Speech and Language

Michael Riley, Cyril Allauzen, and Martin Jansche, Google Inc.

Finite-state methods are well established in language and speech processing. OpenFst (available from `www.openfst.org`) is a free and open-source software library for building and using finite automata, in particular, weighted finite-state transducers (FSTs). This tutorial is an introduction to weighted finite-state transducers and their uses in speech and language processing. While there are other weighted finite-state transducer libraries, OpenFst (a) offers, we believe, the most comprehensive, general and efficient set of operations; (b) makes available full source code; (c) exposes high- and low-level C++ APIs that make it easy to embed and extend; and (d) is a platform for active research and use among many colleagues.

1 Tutorial Outline

1. Introduction to OpenFst

   The first part of the tutorial introduces operations on weighted automata such as determinization and intersection/composition as well as the corresponding OpenFst binaries and library-level APIs that implement those operations. We describe how to read FSTs from simple textual descriptions and combine them into larger and more complex machines, and optimize them using simple command-line and library calls.

   - Introduction
     - Motivating examples
     - Finite-state methods in NLP and Speech
   - A quick tour of OpenFst
     - Finite-state machines and operations
     - OpenFst binaries
     - High-level C++ API
     - Human-readable file formats
   - Comparison of OpenFst and competing libraries
     - Comparison with the AT&T FSM Library^{TM}
     - Brief comparison with SFST and related libraries
   - Advanced usage of OpenFst
     - Low-level C++ API
     - Constructing and modifying `Fst` objects programmatically
     - Implementing new concrete `Fst` classes
     - Adding new weight semirings
     - Customizing matchers and filters for composition

2. Applications

   The second part of the tutorial focuses on several application areas of interest to the NAACL HLT audience, including speech recognition, speech synthesis, and general text processing. In each application area we discuss a straightforward example in detail, then delve into an advanced example that highlights important features of OpenFst, common pitfalls, efficiency considerations, new research directions, etc. These examples are drawn from our extensive experience in applying OpenFst to problems in speech and language processing.

   - Automatic speech recognition
     - Context dependency transducers
     - Language models and grammars
   - Natural Language Processing
     - Unicode processing
     - Text analysis, text normalization
     - Pronunciation models
   - Other areas
     - Computational biology
     - Pattern matching

2 Target Audience

This tutorial is intended for students, researchers, and practitioners interested in applying finite-state methods. It is suitable for participants of a variety of backgrounds, including those without any background in finite-state techniques as well as advanced users of existing software packages. For those unfamiliar with finite-state transducers, the first portion of the tutorial provides an introduction to the theory and application areas. Users of other finite-state libraries will particularly benefit from the contrastive description of the unique features of OpenFst, its C++ API, and the examples drawn from real-world applications.

3 Presenters

Michael Riley began his career at Bell Labs and AT&T Labs where he, together with Mehryar Mohri and Fernando Pereira, introduced and developed the theory and use of weighted finite-state transducers (WFSTs) in speech and language. This work was recognized in best paper awards from the journals *Speech Communication* and *Computer Speech and Language*. He has been a research scientist at Google, Inc. since 2003. He is a principal author of the OpenFst library and the AT&T FSM Library$^{\text{TM}}$. He has given several tutorials on WFSTs before: at ACL 1994, Coling 1997 and Interspeech 2002.

Cyril Allauzen is another key author of the OpenFst library. His main research interests are in finite-state methods and their applications to text, speech and natural language processing and machine learning. Before joining Google, he worked as a researcher at AT&T Labs – Research and at NYU's Courant Institute of Mathematical Sciences.

Martin Jansche has applied the OpenFst library to several speech and language problems at Google. His FST-related interests are in text processing for speech tasks and in learning and applying pronunciation and transliteration models.

OntoNotes: The 90% Solution

Sameer S. Pradhan and Nianwen Xue

OntoNotes is a five year multi-site collaboration between BBN Technologies, Information Sciences Institute of University of Southern California, University of Colorado, University of Pennsylvania and Brandeis University. The goal of the OntoNotes project is to provide linguistic data annotated with a skeletal representation of the literal meaning of sentences including syntactic parse, predicate-argument structure, coreference, and word senses linked to an ontology, allowing a new generation of language understanding technologies to be developed with new functional capabilities.

In its third year of existence, the OntoNotes project has generated a large amount of high quality data covering various layers of linguistic annotation. This is probably the first time that data of such quality has been available in large quantities covering multiple genres (newswire, broadcast news, broadcast conversation and weblogs) and languages (English, Chinese and Arabic). The guiding principle has been to find a "sweet spot" in the space of *inter-tagger agreement*, *productivity*, and *depth of representation*. The most effective use of this resource for research requires simultaneous access to multiple layers of annotation. This has been made possible by representing the corpus with a relational database to accommodate the dense connectedness of the data and ensure consistency across layers. In order to facilitate ease of understanding and manipulability, the database has also been supplemented with a object-oriented Python API.

The tutorial consists of two parts. In the first part we will familiarize the user with this new resource, describe the various layers of annotations in some detail and discuss the linguistic principles and sometimes practical considerations behind the important design decisions that shapes the corpus. We will also describe the salient differences between the three languages at each layer of annotation and how linguistic peculiarities of different languages were handled in the data.

In the second part, we will describe the data formats of each of the layers and talk about various design decisions that went into the creation of the architecture of the database and the individual tables comprising it, along with issues that came up during the representation process and compromises that were made without sacrificing some primary objectives one of which being the independent existence of each layer that is necessary to allow multi-site collaboration. We will explain how the database schema attempts to interconnect all the layers. Then we will go into the details of the Python API that allows easy access to each of the layers and show that by making the objects closely resemble database tables, the API allows for their flexible integration. This will be followed by a hands-on working session.

1 Tutorial Outline

1. Annotation Layers
    - Overview of OntoNotes
    - Design principles
        - Depth of annotation
        - Consistency (ITA)
        - Linguistics principles
    - Potential applications
        - Question Answering
        - Machine Translation
    - Layers of Annotation in English, Chinese and Arabic
        - Treebank
        - PropBank
        - Word Sense

    - Name
    - Coreference
    - Ontology
  - Comparison with existing multi-layer annotation corpora

2. Data Access API
  - Data
    - File format
    - Metadata specification
  - Database schema representing each layer of annotation
    - ER diagram
    - Inter-connection between the annotation layers (database tables)
  - Python Access API
    - Introduction to the Python modules
    - Correspondence between MySQL tables and Python classes
    - Introduction to some frequently used module functionalities
    - Extending the API to add a new layer of annotation
  - Hands on Session
    - Creating a sample OntoNotes database from MySQL dump file
    - Loading it into memory
    - Creating Python objects representing various annotation layers
    - Performing cross-layer queries using a combination of API and database
      - We will provide some sample queries
      - Users can use their own experience to generate novel queries
    - Manipulating the data as in Python world and MySQL world
    - Writing the modified versions back to the database

2 Target Audience

This tutorial is designed for people interested in using one or more layers of OntoNotes in their research to further language understanding through improved shallow semantic analysis. Detailed knowledge of any of the layers is not necessary. Some familiarity with Python would be preferable.

Sameer Pradhan is a Research Scientist at BBN Technologies. His research interests include computational semantics, question answering, application of machine learning to language understanding and annotation science. He have been leading the data integration and coreference annotation effort in the DARPA funded GALE OntoNotes project at BBN. In the past he was the technical lead on the AQUAINT project at BBN. He serves on the ACL SIGANN committee and has been one of the organizers of the Linguistics Annotation Workshops (LAW II and III) He has been on the programme committees of Workshop on UIMA for NLP, and Conference on Global Interoperability of Language Resources (ICGL) He has also served on the guest Editorial Board of Computational Linguistics: Special issue on Semantic Role Labeling. He got his PhD in Computer Science at the University of Colorado at Boulder.

Nianwen Xue is an Assistant Professor of Language & Linguistics and Computer Science at Brandeis University. His research interests include formal representation of linguistic structures and its impact on natural language processing, aspects of syntax, computational linguistics, corpus linguistics and Chinese language processing. He is currently leading the effort to expand the Chinese Treebank, Proposition Bank and word sense annotation, funded by DARPA as part of the GALE OntoNotes project. He serves on the ACL SIGANN committee. He is one of the organizers of the Linguistics Annotation Workshops (LAW II and III) and is also on the organizing committee of the CoNLL Shared Task on Syntactic and Semantic Dependencies in Multiple Languages. He has also served on the guest Editorial Board of Computational Linguistics: Special issue on Semantic Role Labeling. He got his PhD in linguistics from University of Delaware.

VerbNet overview, extensions, mappings and applications

(Karin Kipper Schuler, Anna Korhonen, Susan Brown)

Abstract:

The goal of this tutorial is to introduce and discuss VerbNet, a broad coverage verb lexicon freely available on-line. VerbNet contains explicit syntactic and semantic information for classes of verbs and has mappings to several other widely-used lexical resources, including WordNet, PropBank, and FrameNet. Since its first release in 2005 VerbNet is being used by a large number of researchers as a means of characterizing verbs and verb classes.

The first part of the tutorial will include an overview of the original Levin verb classification; introduce the main VerbNet components, such as thematic roles and syntactic and semantic representations, and present a comparison with other available lexical resources.

During the second part of the tutorial, we will explore VerbNet extensions (how new classes were derived and created through manual and semi-automatic processes), and we will present on-going work on automatic acquisition of Levin-style classes in corpora. The latter is useful for domain-adaptation and tuning of VerbNet for real-world applications which require this.

The last part of the tutorial will be devoted to discussing the current status of VerbNet; including recent work mapping to other lexical resources, such as PropBank, FrameNet, WordNet, OntoNotes sense groupings, and the Omega ontology. We will also present changes designed to regularize the syntactic frames and to make the naming conventions more transparent and user friendly. Finally, we will describe some applications in which VerbNet has been used.

Tutorial Outline:

VerbNet overview:

- Original Levin classes
- VerbNet components (roles, syntactic and semantic descriptions)
- Related work in lexical resources

VerbNet extensions:

- Manual and semi-automatic extension of VerbNet with new classes
- On-going work on automatic acquisition of Levin-style classes in corpora

VerbNet mappings and applications:

- Mappings to other resources (PropBank, FrameNet, WordNet, OntoNotes sense groupings, Omega ontology)
- Current status of VerbNet
- Ongoing improvements
- VerbNet applications

Short biographical description of the presenters:

Karin Kipper Schuler
University of Colorado
kipper@verbs.colorado.edu

Karin Kipper Schuler received her PhD from the Computer Science Department of the University of Pennsylvania. Her primary research is aimed at the development and evaluation of large-scale computational lexical resources. During the past couple of years she worked for the Mayo Clinic where she was involved in applications related to bio/medical informatics including automatic extraction of named entities and relations from clinical data and development of knowledge models to guide this data extraction.

Anna Korhonen
University of Cambridge
alk23@cam.ac.uk

Anna Korhonen received her PhD from the Computer Laboratory of the University of Cambridge in the UK. Her research focuses mainly on automatic acquisition of lexical information from texts. She has developed techniques and tools for automatic lexical acquisition for English and other languages, and has used them to acquire large lexical resources , extend manually built resources, and help NLP application tasks (e.g. parsing, word sense disambiguation, information extraction) . She has applied the techniques to different domains (e.g. biomedical) and has also used them to advance on research in related fields (e.g. cognitive sciences).

Susan Brown
University of Colorado
susan.brown@colorado.edu

Susan Brown is currently a PhD student in Linguistics and Cognitive Science at the University of Colorado under Dr. Martha Palmer's supervision. Susan's research focuses on lexical ambiguity, especially as it pertains to natural language processing tasks. Her research methodologies include psycholinguistic experimentation, corpus study, and annotation analysis. She has also been involved in the development of large-scale lexical resources and the design and construction of a lexically based ontology.

Writing Systems, Transliteration and Decipherment

Kevin Knight (USC/ISI)
Richard Sproat (CSLU/OHSU)

Description

Nearly all of the core data that computational linguists deal with is in the form of text, which is to say that it consists of language data written (usually) in the standard writing system for the language in question. Yet surprisingly little is generally understood about how writing systems work. This tutorial will be divided into three parts. In the first part we discuss the history of writing and introduce a wide variety of writing systems, explaining their structure and how they encode language. We end this section with a brief review of how some of the properties of writing systems are handled in modern encoding systems, such as Unicode, and some of the continued pitfalls that can occur despite the best intentions of standardization. The second section of the tutorial will focus on the problem of transcription between scripts (often termed "transliteration"), and how this problem—which is important both for machine translation and named entity recognition—has been addressed. The third section is more theoretical and, at the same time we hope, more fun. We will discuss the problem of decipherment and how computational methods might be brought to bear on the problem of unlocking the mysteries of as yet undeciphered ancient scripts. We start with a brief review of three famous cases of decipherment. We then discuss how techniques that have been used in speech recognition and machine translation might be applied to the problem of decipherment. We end with a survey of the as-yet undeciphered ancient scripts and give some sense of the prospects of deciphering them given currently available data.

Outline

First hour:

- History of writing
- Survey of writing systems and how they work
- Modern encodings

Second hour:

- Problems of transcription (transliteration)
- Generative models of transcription

Break

- More on generative models of transcription
- Discriminative models

Third Hour

- Famous cases of decipherment
- Prospects for "autodecipherment"
- What's left to decipher?

Target Audience

This tutorial will be of interest to anyone who wishes to have a better understanding of how writing (the form of language that most computational linguists deal with) works, and how such problems as transcription (transliteration) and decipherment are approached computationally.

Bios

Kevin Knight is a Research Associate Professor in Computer Science at the University of Southern California, a Senior Research Scientist and Fellow at the USC/Information Sciences Institute, and Chief Scientist at Language Weaver. Dr. Knight received a Ph.D. from Carnegie Mellon University in 1992, and a bachelor's degree from Harvard University. His current interests include better statistical machine translation through linguistics, and he is also working on exploiting cryptographic techniques to solve hard translation problems.

Richard Sproat received his Ph.D. in Linguistics from the Massachusetts Institute of Technology in 1985. Since then he has worked at AT&T Bell Labs, at Lucent's Bell Labs and at AT&T Labs – Research, before joining the faculty of the University of Illinois, and subsequently the Oregon Health & Science University. Sproat has worked in numerous areas relating to language and computational linguistics, including syntax, morphology, computational morphology, articulatory and acoustic phonetics, text processing, text-to-speech synthesis, writing systems, and text-to-scene conversion.