Making Hidden Markov Models More Transparent

Nashira Richard Lincoln[*] and **Marc Light**[†]
[*†]Linguistics Department
[†]School of Library and Information Science
[†]Computer Science Department
University of Iowa
Iowa, USA 52242
{nashira-lincoln, marc-light}@uiowa.edu

Abstract

Understanding the decoding algorithm for hidden Markov models is a difficult task for many students. A comprehensive understanding is difficult to gain from static state transition diagrams and tables of observation production probabilities. We have built a number of visualizations depicting a hidden Markov model for part-of-speech tagging and the operation of the Viterbi algorithm. The visualizations are designed to help students grasp the operation of the HMM. In addition, we have found that the displays are useful as debugging tools for experienced researchers.

1 Introduction

Hidden Markov Models (HMMs) are an important part of the natural language processing toolkit and are often one of the first stochastic generation models that students[1] encounter. The corresponding Viterbi algorithm is also often the first example of dynamic programming that students encounter. Thus, HMMs provide an opportunity to start students on the correct path of understanding stochastic models, **not** simply treating them as black boxes. Unfortunately, static state transition diagrams, tables of probability values, and lattice diagrams are not enough for many students. They have a general idea of how a HMM works but often have common misconceptions. For example, we have found that students often believe that as the Viterbi algorithm calculates joint state sequence observation sequence probabilities, the best state sequence so far is always a prefix of global best path. This is of course false. Working a long example to show this is very tedious and thus text books seldom provide such examples.

Even for practitioners, HMMs are often opaque in that the cause of a mis-tagging error is often left uncharacterized. A display would be helpful to pinpoint why an HMM chose an incorrect state sequence instead of the correct one.

Below we describe two displays that attempt to remedy the above mentioned problems and we discuss a Java implementation of these displays in the context of a part-of-speech tagging HMM (Kupiec, 1992). The system is freely available and has an XML model specification that allows models calculated by other methods to be viewed. (A standard maximum likelihood estimation was implemented and can be used to create models from tagged data. A model is also provided.)

2 Displays

Figure 1 shows a snapshot of our first display. It contains three kinds of information: most likely path for input, transition probabilities, and history of most likely prefixes for each observation index in the Viterbi lattice. The user can input text at the bottom of the display, e.g., *Pelham pointed out that Georgia voters rejected the bill*. The system then runs Viterbi and animates the search through all possible state sequences and displays the best state sequence prefix as it works its way through the observation

[1]The Introduction to Computational Linguistics course at the University of Iowa has no prerequisites, and over half the students are not CS majors.

Proceedings of the Second ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL, pages 32–36,
Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

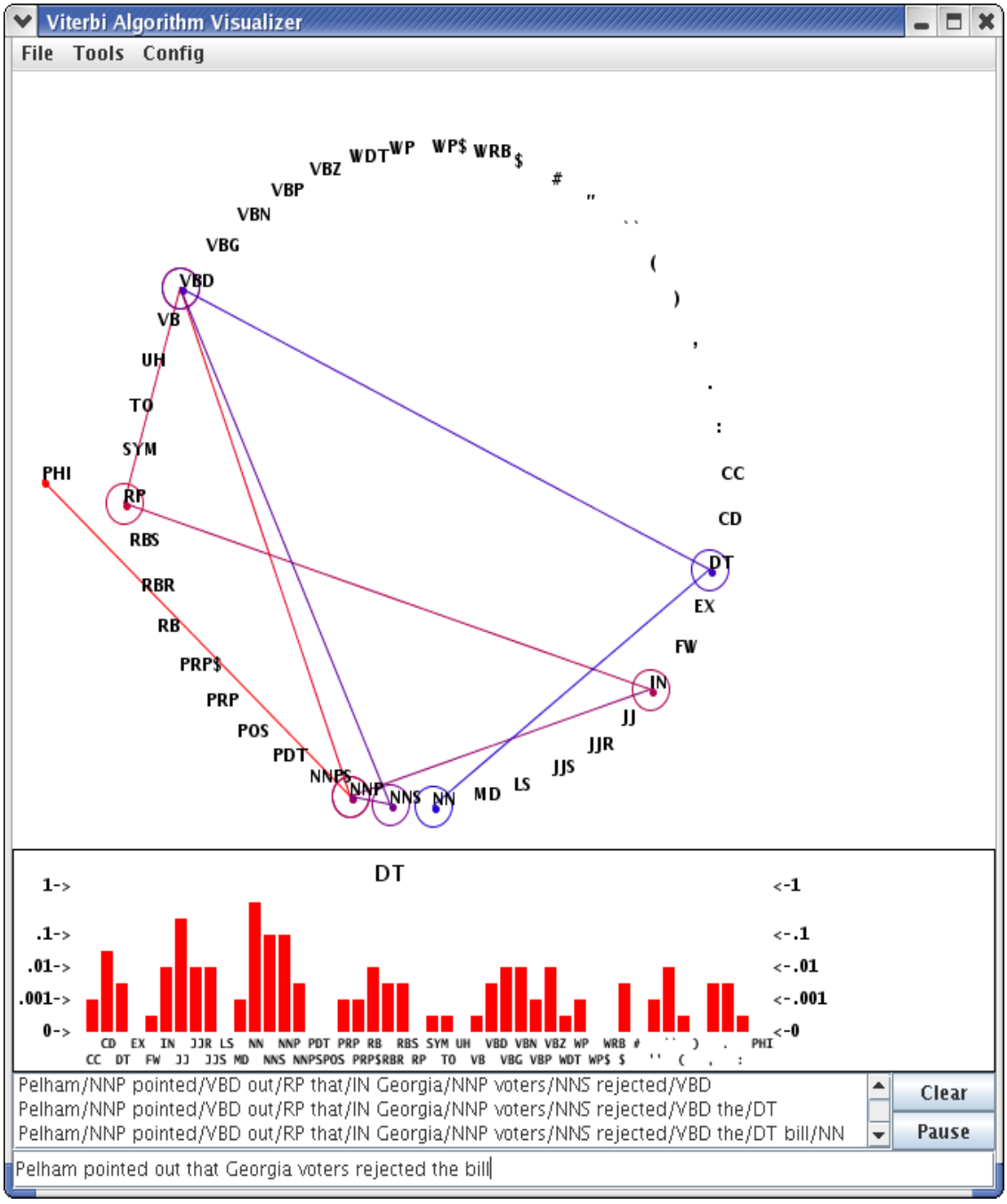

Figure 1: The system's main display. **Top pane**: shows the state space and animates the derivation of the most likely path for "Pelman pointed out that Georgia voters ..."; **Middle pane**: a mouse-over-triggered bar graph of out transition probabilities for a state; **Bottom pane**: a history of most likely prefixes for each observation index in the Viterbi lattice. Below the panes is the input text field.

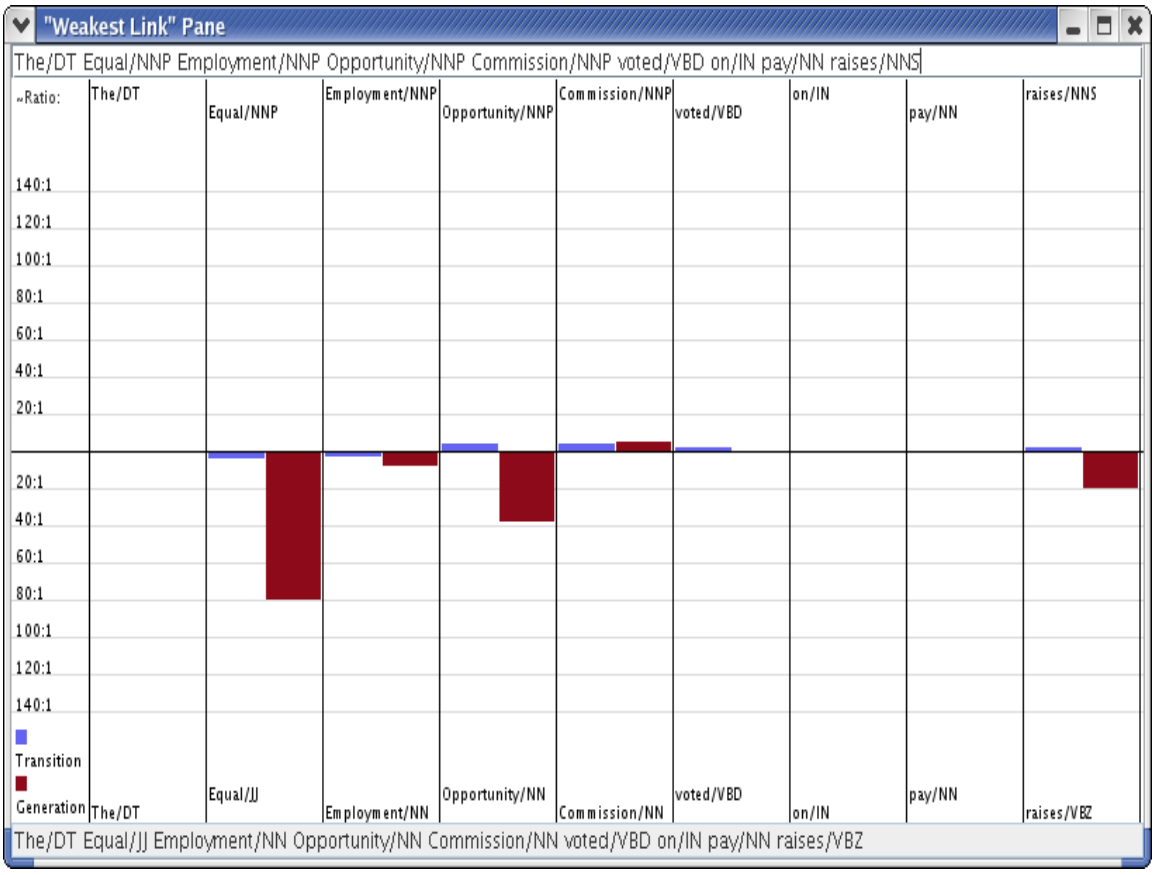

Figure 2: Contrast display: The user enters a sequence on the top text field and presses enter, the sequence is tagged and displayed in both the top and bottom text fields. Finally, the user changes any incorrect tags in the top text field and presses enter and the probability ratio bars are then displayed.

sequence from left to right (these are lines connecting the states in Figure 1). At any point, the student can mouse-over a state to see probabilities for transitions out of that state (this is the bar graph in Figure 1). Finally, the history of most likely prefixes is displayed (this history appears below the bar graph in Figure 1). We mentioned that students often falsely believe that the most likely prefix is extended monotonically. By seeing the path through the states reconfigure itself in the middle of the observation sequence and by looking at the prefix history, a student has a good chance of dispelling the false belief of monotonicity.

The second display allows the user to contrast two state sequences for the same observation sequence. See Figure 2. For each contrasting state pairs, it shows the ratio of the corresponding transition to each state and it shows the ratio of the generation of the observation conditioned on each state. For example, in Figure 2 the transition DT→JJ is less likely than DT→NNP. The real culprit is generation probability P(*Equal*|JJ) which is almost 7 times larger than P(*Equal*|NNP). Later in the sequence we see a similar problem with generating *opportunity* from a NNP state. These generation probabilities seem to drown out any gains made by the likelihood of NNP runs.

To use this display, the user types in a sentence in the box above the graph and presses enter. The HMM is used to tag the input. The user then modifies (e.g., corrects) the tag sequence and presses enter and the ratio bars then appear.

Let us consider another example: in Figure 2, the mis-tagging of *raises* as a verb instead of a noun at the end of the sentence. The display shows us that although NN→NNS is more likely than NN→VBZ, the generation probability for *raises* as a verb is over twice as high as a noun. (If this pattern of mis-taggings caused by high generation probability ratios was found repeatedly, we might consider smoothing these distributions more aggressively.)

3 Implementation

The HMM part-of-speech tagging model and corresponding Viterbi algorithm were implemented based on their description in the updated version, `http://www.cs.colorado.edu/~martin/SLP/updated.html` , of chapter 8 of (Jurafsky and Martin, 2000). A model was trained using Maximum Likelihood from the UPenn Treebank (Marcus et al., 1993). The input model file is encoded using XML and thus models built by other systems can be read in and displayed.

The system is implemented in Java and requires 1.4 or higher to run. It has been tested on Linux and Apple operating systems. We will release it under a standard open source license.

4 Summary and future work

Students (and researchers) need to understand HMMs. We have built a display that allow users to probe different aspects of an HMM and watch Viterbi in action. In addition, our system provides a display that allows users to contrast state sequence probabilities. To drive these displays, we have built a standard HMM system including parameter estimating and decoding and provide a part-of-speech model trained on UPenn Treebank data. The system can also read in models constructed by other systems.

This system was built during this year's offering of *Introduction to Computational Linguistics* at the University of Iowa. In the Spring of 2006 it will be deployed in the classroom for the first time. We plan on giving a demonstration of the system during a lecture on HMMs and part-of-speech tagging. A related problem set using the system will be assigned. The students will be given several mis-tagged sentences and asked to analyze the errors and report on precisely why they occurred. A survey will be administered at the end and improvements will be made to the system based on the feedback provided.

In the future we plan to implement Good-Turing smoothing and a method for dealing with unknown words. We also plan to provide an additional display that shows the traditional Viterbi lattice figure, i.e., observations listed left-to-right, possible states listed from top-to-bottom, and lines from left-to-right connecting states at observation index *i* with the previous states, *i-1*, that are part of the most likely state sequence to *i*. Finally, we would like to incorporate an additional display that will provide a visualization of EM HMM training. We will use (Eisner, 2002) as a starting point.

References

Jason Eisner. 2002. An interactive spreadsheet for teaching the forward-backward algorithm. In *Proc. of the ACL 2002 Workshop on effective tools and methodologies for teaching natural language processing and computational linguistics*.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: an introduction to natural language processing, and computational linguistics, and speech recognition*. Prentice-Hall.

J. Kupiec. 1992. Robust part-of-speech tagging using a hidden markov model. *Computer Speech and Language*, 6:225–242.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, June.