CMPT 825: Natural Language Processing **Spring 2008**

Lecture 5 — Feb 13, 2006

Lecturer: Anoop Sarkar *Scribe: Mohsen Jamali*

5.1 Estimating Hidden Markov Model Topologies [2]

Hidden Markov Models need a large set of parameters which are induced from a text-corpus. The parameters should be optimal in the sense that resulting models assign high probabilities to seen training data. There are several methods to estimate model parameters. The first one is to use each word as a state and estimate the probabilities using the relative frequencies. The second method is a variation of the first method. In this model, words are automatically grouped by similarity of distribution in the corpus. Each group is represented by a state in the model. The second method has the advantage of drastically reducing the number of model parameters and thereby reducing the sparse data problem.

The third method uses manually defined categories. An important difference to the second method with automatically derived categories is that with manual definition a word can belong to more than one category. The fourth method is a variation of the third method and is also used for part of speech tagging. This method does not need a pre-annotated corpus for parameter estimation. The parameters are estimated using the Baum-Welch algorithm.

This paper proposes a fifth method for estimating natural language models combining the advantages of the methods mentioned above.

5.1.1 Part of Speech Tagging

The task of PoS tagging is the unique annotation of a word with a syntactic category, called *part-of-speech* or *tag*. Given $W = w_1...w_k \in \Sigma^*$ the sequence of observed words, we are looking for a sequence of tags $T = t_1...t_k \in \tau^*$ that maximized the conditional probability $P(T|W)$, hence we are looking for :

$$argmax_T p(T|W) = argmax_T \frac{p(T)p(W|T)}{p(W)}$$

5.1.2 Model Merging

This technique not only induces transitions and output probabilities from the corpus, but also the model topology. In *n-gram* approaches, the states are fixed and mostly linguistically motivated. The model merging approach groups words together by their statistical distributions and estimates transition and output probabilities for a HMM at the same time. This approach adapts to the amount of data available. There is a limiting factor for Model Merging approach that the process of merging is very time consuming.

Merging starts with an initial model. To do this we can choose a trivial HMM that exactly matches the corpus. there is exactly one path for each expression in the corpus. Each path gets the probablity of $1/u$, with u the number of expressions in the corpus. Figure 5.1 shows the trivial HMM for a corpus with words a, b, c and expressions $ab, ac, abac$

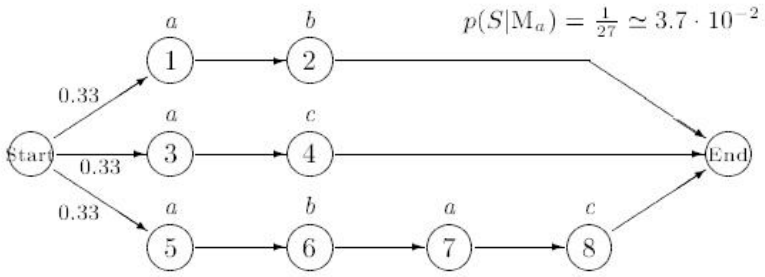

Figure 5.1. The trivial HMM for a single corpus cosisting of expressions ab,ac,abac

States are merged successively, except for the start and end state. The transitions from and to the old states are redirected to the new merged state, the probabilities are adjusted to maximize the likelihood of the corpus; also the outputs are joined. Of all possible merges we take the merge that results in the minimal change of the probability. The probability never increases because the trivial model is the maximum likelihood model which maximizes the probability of the corpus given the HMM. Model merging stops when a predefined threshold for corpus probability is reached.

5.1.3 Model Merging and Natural Language Models

Model merging has three important features: 1- It can classify words into categories, 2- One word may belong to several categories, 3- Model merging can

	categorization of words	multiple categories for words	sequence recognition
n-grams, manual	✓	✓	×
n-grams, automatic	✓	×	×
HMMs, model merging	✓	✓	✓

Table 5.1. Features of Categorizations for Language Models

recognize static sequences of words or parts-of-speech. Table 5.1 compares the features included in different estimation approaches.

Introducing Constraints

The Model Merging algorithm is really time consuming (generally $O(l^4)$, where l is the length of the corpus -this complexity seems to be incorrect, it should be either l^3 or l^5 as discussed in the class-).

In the merging steps of the algorithm, mainly, states that output words of the same syntactic category are merged. This behavior can be exploited by introducing constraints on the merging process. At the beginning we consider only states with the same output. After a while, this constraint is relaxed and states that output words of the same syntactic category. Again, after a while, this constraint is relaxed and all states can merge.

Model Merging and Part-of-Speech Tagging

This paper extend the Markov model in such a way that the output no longer consists of a single word but of a word with its annotated part-of-speech. The task of part-of-speech tagging in this case would be to determine the sequence of tags $T = t_1...t_k$ with the highest probability from all sequences of states $Q = q_1...q_k$ that can output a given sequence of words $W = w_1...w_k$. For a given sequence of words W we have to find

$$argmax_T \sum_Q P(Q) \cdot P(W, T|Q)$$

Since most of the time, the will be one main path, it suffices to find the Viterbi approximation

$$argmax_T \max_Q P(Q) \cdot P(W, T|Q)$$

The last formula shows that it is useless to merge two states that output the same words but with two different categories. Because if we do that, the resulting model will always favor the part-of-speech with higher probability.

Using the derived models for part-of-speech tagging, we may encounter two problems. A lot of sequences can not be recognized, because there are unknown words in them, or even all words are known, there is no matching path in the HMM. The problem of unknown words can be solved by a mapping from unknown words to known words and using the states emitting known words for unknown words. The problem of non-existing paths can be solved by "smoothing" the transition probabilities between states.

Experiments

They perform a supervised learning using labeled data to train their model. They show some improvements in accuracy, but the problem is that, they don't show the significance of their improvement in accuracy; whether just two errors have been fixed or 1000 errors have been fixed.

5.2 Structure Learning in conditional probability models via an entropic prior and parameter extinction [1]

This paper introduces an entropic prior for multinomial parameter estimation problems and solve for its maximum a posteriori (MAP) estimator. The prior is bias for maximally structured and minimally ambiguous models. Iterative maximum a posteriori (MAP) estimation using this prior tends to drive weakly supported parameters toward extinction, sculpting a lower-dimensional model whose structure comes to reflect that of the data. To accelerate this process, they establish when weakly supported parameters can be trimmed from the model.

5.2.1 A maximum structure entropic prior

In entropic estimation, we assert that parameters that do not reduce uncertainty are improbable. For example, in a multinomial distribution over K mutually exclusive kinds of events, a parameter at chance $\Theta_i = \frac{1}{k}$ adds no

information to the model, and is thus a wasted degree of freedom. In this view, learning is a process which increase the specificity of the model, or equivalently, minimizing the entropy. This intuition is well captured by the expression:

$$P_e(\mathbf{\Theta}) \propto e^{-h(\mathbf{\Theta})} = exp \sum_i \Theta_i \log \Theta_i = \Pi_i \Theta_i^{\Theta_i} = \mathbf{\Theta}^{\mathbf{\Theta}}$$

$P_e(.)$ is non-informative to the degree that it does not favor one parameter set over another provided they specify equal uncertain models. Combining the prior with the multinomial yields the entropic posterior:

$$P_e(\mathbf{\Theta}|\omega) \propto P(\omega|\mathbf{\Theta}) P_e(\mathbf{\Theta}) \propto \Pi_i^N \Theta_i^{\Theta_i + \omega_i}$$

where non-negative ω_i is evidence(observation) for event type i.

As an example, consider the case in which we are tossing a coin. Our model has two parameters Θ_h and Θ_t which correspond to the probabilities for head and tail. Considering the following observations of evidence, we will have:

$$D =< h, h, h, t, t, h, h >$$

$$\mathbf{\Theta} = \{\Theta_h, \Theta_t\} \quad \Theta_h + \Theta_t = 1$$

$$\omega = \{\omega_h, \omega_t\} \quad N = \omega_h + \omega_t$$

$$P(D|\mathbf{\Theta}) = \Theta_h^5 \cdot \Theta_t^2 \quad P(\mathbf{\Theta}) = \Theta_h^{\Theta_h} \cdot \Theta_t^{\Theta_t}$$

$$P(\mathbf{\Theta}|D) \propto \Theta_h^{\Theta_h + 5} \cdot \Theta_t^{\Theta_t + 2}$$

As figure 5.2 shows, with ample evidence this distribution becomes sharply peaked around the maximum likelihood estimate, but with scant evidence it flattens and skews to stronger odds. Note that this is the opposite behavior that one obtains from a Dirichlet prior, often used in learning Bayes net parameters from data.

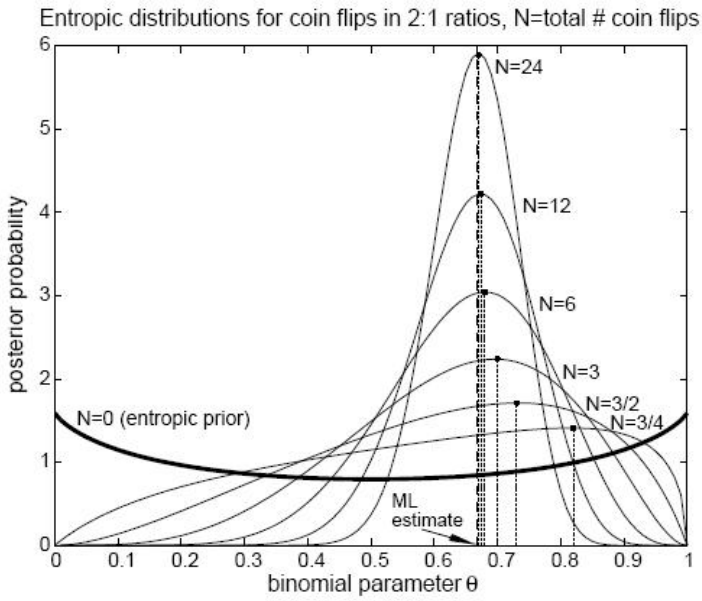

Figure 5.2. entropic p.d.f.s over a binomial parameter Θ_h

5.2.2 Interpretation

The entropic MAP estimator strikes a balance which favors fair (ML) parameter values when data is extensive, and biases toward low-entropy parameters when data is scarce. The less the entropy the more sure you are about the probabilities. It leads to more sharp models rather than defused models. The entropic MAP estimator may be understood to select the strongest hypothesis compatible with the data, rather than fairest, or best unbiased model. Formally, some manipulation of the posterior allows us to understand the MAP estimate in terms of entropies:

$$-max_{\Theta}\log P_e(\mathbf{\Theta}|\omega) = min_{\Theta} -\log \Pi_1^N \Theta_i^{\Theta_i+\omega_i} = ... = min_{\Theta} H(\mathbf{\Theta}) + D(\omega||\mathbf{\Theta}) + H(\omega)$$

The first term $min_{\Theta}H(\mathbf{\Theta})$ measures the ambiguity in the model and shows the uncertainty you have about the model. As $H(\mathbf{\Theta})$ declines, the model becomes increasingly structured and near-deterministic. The second term $D(\omega||\mathbf{\Theta})$ id the relative entropy and measures divergence between the parametrer $\mathbf{\Theta}$ and the data's descriptive statistics ω. $H(\omega)$ is a lower bound on the expected number of bits needed to specify which of the variations al-

lowed by the model is instantiated by the data. As $H(\omega)$ declines, the model comes to agree with the underlying structure of the data.

5.2.3 Training

The entropic posterior defines a distribution over all possible model structures and parameterizations within a class; small, accurate models having minimal ambiguity in their joint distribution are the most probable. To find these models, we simply replace the M-step of EM with the entropic MAP estimator, with the following effect: First, the E-step distributes probability mass unevenly through the model, because the model is not in perfect accordance with the intrinsic structure of the training data. In the MAPstep, the estimator exaggerates the dynamic range of multinomials in improbable parts of the model. This drives weakly supported parameters toward zero and concentrates evidence on surviving parameters, causing their estimates to approach the ML estimate. Structurally irrelevant parts of the model gradually expire, leaving a skeletal model whose surviving parameters become increasingly well-supported and accurate.

5.2.4 Trimming

In this section, we explore some conditions under which we can drive irrelevant parameters to zero. One may trim a parameter Θ_i whenever the loss in the likelihood is balanced by a gain in the prior

$$P_e(\mathbf{\Theta}\backslash\Theta_i|\mathbf{X}) \;\geqslant\; P(\mathbf{\Theta}|\mathbf{X})$$

$$P(\mathbf{X}|\mathbf{\Theta}\backslash\Theta_i)P_e(\mathbf{\Theta}\backslash\Theta_i) \;\geqslant\; P(\mathbf{X}|\mathbf{\Theta})P_e(\mathbf{\Theta})$$

Comparing the trimming approach with simple model merging approach we see that in the model merging approach, we have sequenced comparisons, but in trimming, we check the state for the first inequality. If the inequality holds, then we throw out the state and continue with the remaining states.

Trimming accelerates training by removing parameters that would otherwise decay asymptotically to zero. Although the mathematics makes no recommendation when to trim, as a matter of practice we wait until the model is at or near convergence. Note that if a model is to be used for lifelong learningperiodic or gradual retraining on samples from a slowly evolving

non-stationary processthen trimming is not advised, since nearly extinct parameters may be revived to model new structures that arise as the process evolves.

In sum, a parameter can be trimmed when varying it increases the entropy faster than the log-likelihood.

$$\Theta_i \geqslant exp[-\frac{\partial \log P(\mathbf{X}|\mathbf{\Theta})}{\partial \Theta_i}]$$

Conveniently, the gradient of the log-likelihood $\frac{\partial \log P(\mathbf{X}|\mathbf{\Theta})}{\partial \Theta_i}$ will have already been calculated for re-estimation in most learning algorithms.

5.2.5 Continuous-output HMMs

A hidden Markov model is a dynamically evolving mixture model, where mixing probabilities in each time-step are conditioned on those of the previous time-step via a matrix of transition probabilities.

Authors of the paper compared entropically and conventionally estimated continuous-output HMMs on sign-language gesture data provided by a computer vision lab. Entropic estimation consistently yielded HMMs with simpler transition matrices having many parameters at or near zero (e.g., figure 5.3)lower-entropy dynamical models. When tested on held-out sequences from the same source, entropically trained HMMs were found to over-fit less in that they yielded higher log-likelihoods on held-out test data than conventionally trained HMMs. This translated into improved classification: The entropically estimated HMMs also yielded superior generalization in a binary gesture classification task

Most interestingly, the dynamic range of surviving transition parameters was far greater than that obtained from conventional training. This remedies a common complaint about continuous-output HMMs that model selectivity is determined mainly by model structure, secondly by output distributions, and only lastly by transition probabilities, because they have the smallest dynamic range.

Transition Trimming

Following is a trimming criterion for HMM transition parameters:

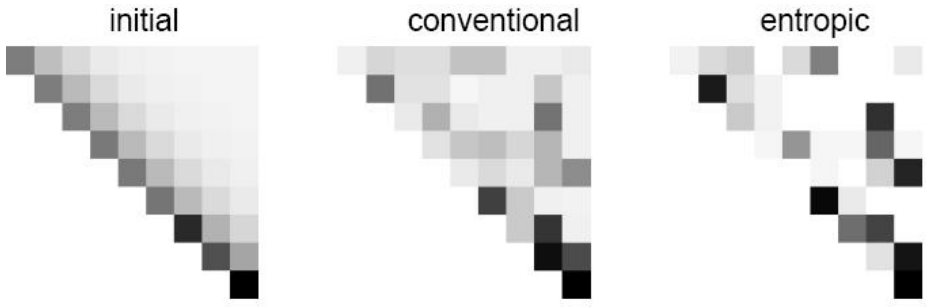

Figure 5.3. Initial, Baum-Welch, and entropically re-estimated transition matrices. Each row depicts transition probabilities from a single state; white is zero. The first two matrices are fully upper-diagonal; the rightmost is sparse.

$$\Theta_{i|j} \leqslant exp[\frac{\partial \log P(\mathbf{X}|\mathbf{\Theta})}{\partial \Theta_{i|j}}]$$

$$= exp[-\frac{\sum_{t=1}^{T-1} \alpha_j(t) p(x_{t+1}|s_i) \beta_i(t+1)}{\sum_k^N \alpha_k(T)}]$$

This test licenses a deletion when the transition is relatively improbable and the source state is seldom visited. In continuous-output HMMs, entropic training can produce two kinds of states: data-modeling, having output distributions tuned to subsets of the data; and gating, having near-zero durations ($\Theta_{i|i} \approx 0$) and often having highly non-selective output probabilities.

State trimming

One of the more interesting properties of entropic training is that it tends to reduce the occupancy rate of states that do little to direct the flow of probability mass, whether by vice of broad output distributions or non-selective exit transitions. As a result their incoming transitions become so attenuated that such states are virtually pinched off from the transition graph. As with transitions, one may detect a trimmable state s_i by balancing the prior probability of all of its incoming and exit transitions against the probability mass that flows through it:

$$\frac{P(\mathbf{X}|\mathbf{\Theta} \backslash s_i)}{P(\mathbf{X}|\mathbf{\Theta})} \geqslant \Theta_{i|i}^{\Theta_{i|i}} \Pi_{j \neq i}^N \Theta_{j|i}^{\Theta_{j|i}} \Theta_{i|j}^{\Theta_{i|j}}$$

However, this is speculative computation, which they try to avoid. They propose a non-speculative heuristic that we found equally effective: They bias transition-trimming to zero self-transitions first. Continued entropic training then drives an affected states output probabilities to extremely small values, often dropping the states occupancy low enough to lead to its being pinched off.

5.2.6 Discrete-output HMMs

Discrete-output HMMs are composed entirely of cascaded multinomials. We'l review one of the experiments they carried out to estimate both transition and output probabilities. They entropically and conventionally trained 100-state, 30-symbol discrete-output HMMs on the abstract and introduction of the original version of the article. Entropic training pinched off 4 states and trimmed 94% of the transition parameters and 91% of the output parameters, leaving states that output an average of 2.72 symbols. Some states within the HMM formed near-deterministic chains, e.g., figure 9 shows a subgraph that can output the word fragments rate, that, rotation, tradition, etc. When used to predict the next character given random text fragments taken from the body of the paper, the entropic HMM scored 27while the conventional HMM scored 12%. The subgraph in figure 5.4 probably accounts for the entropic HMMs correct prediction given the word fragment expectat. Figure 10 shows that the entropic model correctly predicts i and a range of less likely but plausible continuations. The conventionally-trained model makes less specific predictions and errs in favor of typical first-order effects, e.g., h often follows t. In predicting i over , h and e, the entropic model is using context going back at least three symbols, since expectation, demonstrate, motivated, automaton, and patterns all occurred in the training sequence.

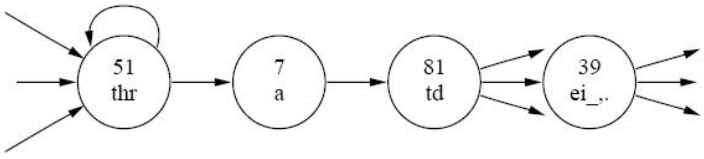

Figure 5.4. High-probability states and subgraphs of interest from a 35-state chorale HMM. Tones output by each state are listed in order of probability. Extraneous arcs are removed for clarity

5.3 Jointly Labeling Multiple Sequences: A Factorial HMM Approach [3]

This paper presents new statistical models for jointly labeling multiple sequences and apply them to the combined task of part-of-speech tagging and noun phrase chunking. They propose a solution by representing multiple sequences in a single Factorial Hidden Markov Model (FHMM).

A Factorial Hidden Markov Model (FHMM) is a hidden Markov model with a distributed state representation. Let $x_{1:T}$ be a length T sequence of observed random variables (e.g. words) and $y_{1:T}$ and $z_{1:T}$ be the corresponding sequences of hidden state variables (e.g. tags, chunks). Then we define the FHMM as the probabilistic model:

$$p(x_{1:T}, t_{1:T}, z_{1:T}) \;=\; \pi_0 \Pi_{t=2}^{T} p(x_t|y_t, z_t) p(y_t|y_{t-1}, z_t) p(z_t|z_{t-1}) \tag{5.1}$$

where $\pi_0 = p(x_0|y_0, z_0)p(y_0|z_0)p(z_0)$. Viewed as a generative process, we can say that the chunk model $p(z_t|z_{t-1})$ generates chunks depending on the previous chunk label, the tag model $p(y_t|y_{t-1}, z_t)$ generates tags based on the previous tag and current chunk, and the word model $p(x_t|y_t, z_t)$ generates words using the tag and chunk at the same time-step.

FHMM parameters can be calculated via maximum likelihood (ML) estimation if the values of the hidden states are available in the training data. Otherwise, parameters must be learned using approximate inference algorithms, since exact Expectation- Maximization (EM) algorithm is computationally intractable.

5.3.1 Adding Cross-Sequence Dependencies

Statistical modeling often involves the iterative process of finding the best set of dependencies that characterizes the data effectively. As shown in Figures 5.5(a), 5.5(b), and 5.5(c), dependencies can be added between the y_t and z_{t-1}, between z_t and y_{t-1}, or both. The model in Fig. 5.5(a) corresponds to changing the tag model in Equation 5.1 to $p(y_t|y_{t-1}, z_t, z_{t-1})$; Fig. 5.5(b) corresponds to changing the chunk model to $p(z_t|z_{t-1}, y_{t-1})$; Fig. 5.5(c), corresponds to changing both tag and chunk models, leading to the probability model:

$$\Pi_{t=1}^{T} p(x_t|y_t, z_t)p(y_t|y_{t-1}, z_t, z_{t-1})p(z_t|z_{t-1}, y_{t-1}) \tag{5.2}$$

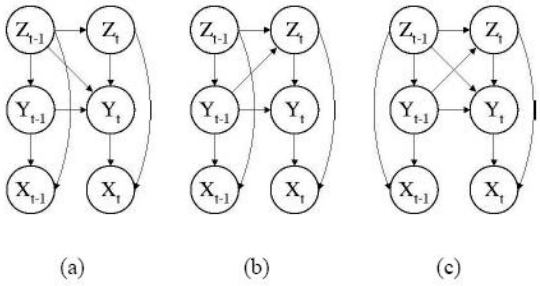

Figure 5.5. FHMMs with additional cross-sequence dependencies. The models will be referred to as (a) FHMM-T, (b) FHMM-C, and (c) FHMM-CT.

Switching Factorial HMM

A reasonable question to ask is, How exactly does the chunk sequence interact with the tag sequence? The approach of adding dependencies in Section 2.2 acknowledges the existence of cross-sequence interactions but does not explicitly specify the type of interaction.

To answer the question, we consider how the chunk sequence affects the generative process for tags: First, we can expect that the unigram distribution of tags changes depending on whether the chunk is a noun phrase or verb phrase. Similarly, a bigram distribution $p(y_t|y_{t-1})$ describing tag transition probabilities differs depending on the bigrams location in the chunk sequence, such as whether it is within a noun phrase, verb phrase, or at a phrase boundary. In other words, the chunk sequence interacts with tags by switching the particular generative process for tags. We model this interaction explicitly using a Switching FHMM:

$$p(x_{1:T}, t_{1:T}, z_{1:T}) \;=\; \pi_0 \Pi_{t=2}^{T} p(x_t|y_t, z_t)p_\alpha(y_t|y_{t-1})p_\beta(z_t|z_{t-1}) \tag{5.3}$$

In this new model, the chunk and tag are now generated by bigram distributions parameterized by α and β. For different values of α (or β), we have different distributions for $p(y_t|y_{t-1})$ (or $p(z_t|z_{t-1})$). The crucial aspect of the model lies in a function $\alpha = f(z_{1:t})$, which summarizes information in

$z_{1:t}$ that is relevant for the generation of y, and a function $\beta = g(y_{1:t})$, which captures information in $y_{1:t}$ that is relevant to the generation of z.

$$p(y_t|y_{t-1}, z_{1:t}) = \sum_{\alpha \ \alpha=1 \ or \ \alpha=2} p(y_t = \alpha|y_{t-1}, z_{1:t}) =$$

$$p(y_t|y_{t-1}, \alpha = 1)p(\alpha = 1) \ + \ p(y_t|y_{t-1}, \alpha = 2)p(\alpha = 2)$$

An idea related to the Switching FHMM is the Bayesian Multinet which allows the dynamic switching of conditional variables. It can be used to implement switching from a higher-order model to a lowerorder model, a form of backoff smoothing for dealing with data sparsity. The Switching FHMMdiffers in that it switches among models of the same order, but these models represent different generative processes.

Bibliography

[1] Brand, M. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation 11*, 5 (July 1999).

[2] Brants, T. Estimating hidden markov model topologies. In *Proceedings of Fourth International Conference on Spoken Language Processing (ICSLP 96)* (Philadelphia, USA, October 1996).

[3] Duh, K. Jointly labeling multiple sequences: A factorial hmm approach. In *43rd Annual Meeting of the Assoc. for Computational Linguistics (ACL 2005), Student Research Workshop* (Ann Arbor, Michigan, USA, June 2005).