Notes on:

Jianfeng Gao and Mark Johnson. 2008. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (EMNLP '08). Association for Computational Linguistics, Stroudsburg, PA, USA, 344-352.

- read the paper here
- sample presentation slides here

Notes by:

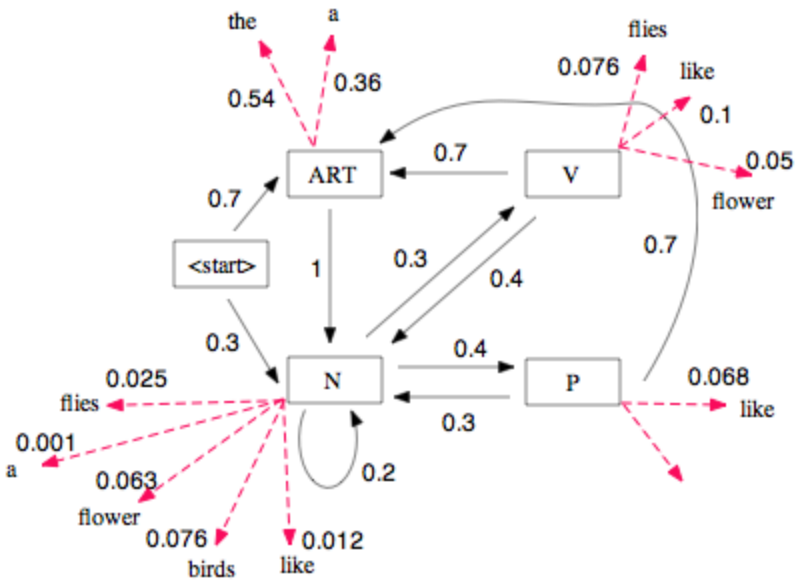
Armineh Nourbakhsh
armineh.nourbakhsh@gmail.com

1. **POS Tagging**
    a. Problem:
        i. input: a string of words: “the lead paint is unsafe”
        ii. output: same string of words, annotated by pos and “bos,” “eos” tags: <s> the/Det lead/N paint/N is/N unsafe/Adj </s>
        iii. **Q:** POS tagging usually involved some morphological analysis (e.g. truncating plurals) and capitalization analysis (e.g. separating proper nouns from general ones). The authors have mentioned that they ignored both analyses in their experiments. How would this affect their evaluation?
    b. POS tagging is slightly different from the classic coin-toss problem because in the latter, (at least in theory) there is no dependency between each toss and the previous or next one. In POS tagging, a word’s position in the context can have a big impact on its pos tag (e.g. “tasty orange juice” vs. “bright orange dress”). In other words, the stream of observations (words) are sequentially dependent on each other.
        i. supervised: training corpus is pos-tagged
        ii. unsupervised: training corpus is not tagged
        iii. semi-supervised: training corpus is not tagged, but there’s a dictionary of possible tags for each word
2. **What are HMMs**
    a. A HMM is a Markov model with hidden (unobserved) states.
    b. A Markov model is a finite state transducer that has the **Markov property**
    c. **The Markov property** indicates the probability distribution of future states of the process depends only upon the present state, not on the sequence of events that preceded it.

3. **How to use HMMs for POS-Tagging**

the HMM model	annotation	POS interpretation
a set of states	$S=s_0,s_1,...,s_m$	POS tags, BOS, EOS
a set of observations	$O=o_1,o_2,...,o_n$ from V	words from vocabulary V, <s> and </s> delimiters
a set of state-to-state transition probabilities	$A=a_{01}, \ldots,a_{mn}$ where a_{ij} is the probability of transition $q_i \rightarrow q_j$	$a_{ij}=P(t_i\|t_{i-1})$
observation (emission) probabilities	$B=\{b_i(o_t)\}$ the probability of observation o_t being generated from state i	$b_i(o_t)=P(w_t\|t_i)$
an initial probability distribution over states	the probability of starting in state i: π_i	$\pi_{BOS}=1$

$\mu=(A,B,\pi)$



- emission probabilities are rather sparse, meaning most words belong to a very limited set of POS tags.

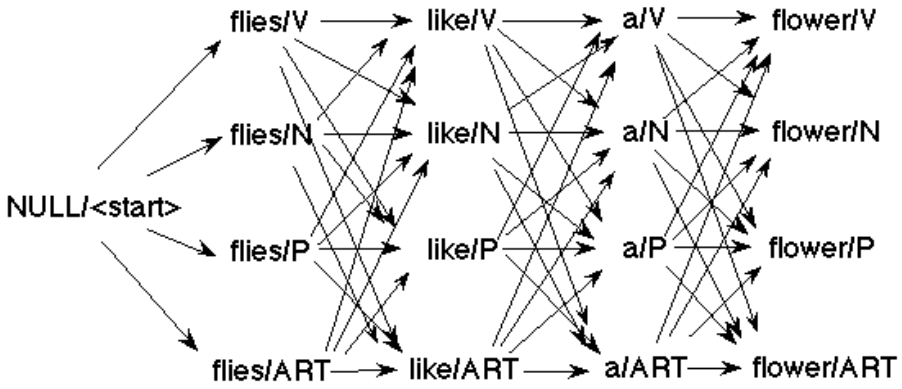
4. Definitive HMM POS problems:

Problem	HMM definition	POS interpretation	Annotation	Algorithms
Probability	Find the	Given a	P(O|μ)	Forward

estimation (Likelihood)	probability of an observation given a model	sentence what's the probability of each tag in the model	$\mu=(A,B,\pi)$	algorithm
Best path estimation (Decoding)	Find the most likely path through a model given an observed sequence	Find the most likely POS sequence given a sentence	$^{argmax}P(S\|O,\mu)$ where S is a sequence of states (tags)	Viterbi algorithm
Parameter estimation (Learning)	**Find the most likely model (params) given an observed sequence**	**Find the most likely (word to POS params) give an observed sequence of words**	**Find best transition and emission matrices for A and B**	**.Supervised: MLE .Unsupervised: EM, VB (approx.) MCMC/Gibbs (true convergence)**

5. Probability estimation:

a. Probability of each state is the product of the initial probability and all the transition and emission probabilities leading up to that state.
b. This problem can easily get computationally intractable, especially for the worst-case scenario (complete graph). Thus we turn to dynamic programming:
c. *Forward algorithm:*
    i. Buffer the probability of each state for use by the next state
    ii. Probability of state at k+1 will be equal to the transition probabilities of states at k, times the initial probability of π_k, times the emission probability $b(o_k)$
    iii. Formulation:
        1. $\alpha_j(1)=\pi_j$
        2. $\alpha_j(k+1)=\sum_{(i=1\ to N)}\alpha_i(k)\ a_{ij}\ b_j(o_k)$
        3. $P(O|\mu)=\sum_{(j=1\ to\ N)}\alpha_j(k+1)$



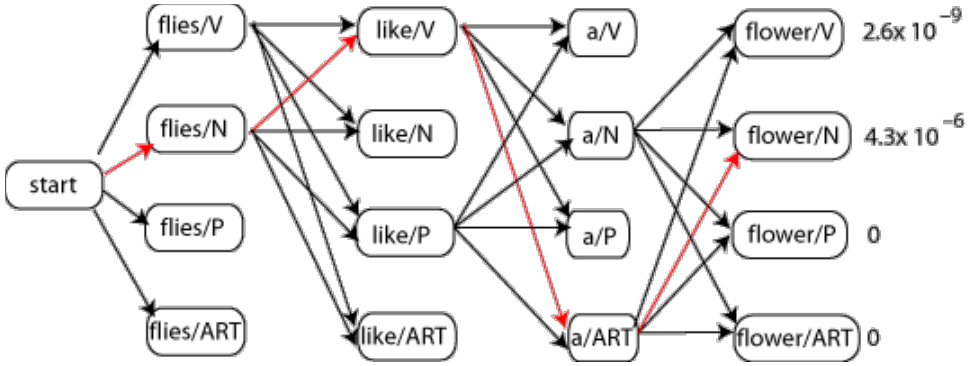
6. Best path estimation:

a. the forward algorithm buffers probabilities at each state. What if we only store the highest probability at each state?

b. *Viterbi algorithm:*

i. $\delta_j(1)=\pi_j$

ii. $\delta_j(k+1)=\max_i \delta_i(k)\, a_{ij}\, b_j(o_k)$



7. Parameter estimation:

a. We used forward probabilities to find p of observations, and used the same forward probabilities to find a best tag sequence. it turns out that backward probabilities (β) are also useful.

b. *The forward-backward algorithm:*

i. probability of whole sentence: $P(O)=\sum_i \alpha_i(k)\, \beta_i(k)$

ii. probability of being in state i at time k: $P(O,S_k=i)=\alpha_i(k)\, \beta_i(k)$, where S is a sequence of states (or tags)

c. *MLE for supervised learning:*

i. $P(w_i|t_i)=C(w_i,t_i)/C(t_i)$

ii. $P(t_i|t_{i-1})=C(t_{i-1},t_i)/C(t_{i-1})$

d. *Unsupervised learning:*

i. No fine-tuning of parameters per word

ii. Suppose the transition probability matrix A is parameterized by a set of multinomials θ and the emission probability matrix B is parameterized by another set of multinomials φ. θ_t specified the distribution over states t´

following t, and φ_t specifies the distribution over words w given state t.

iii. Since the dirichlet distribution is conjugate to multinomials, and also captures sparsity pretty well (most words have only one pos), we can say:
   1. $\theta_t \mid \alpha \sim \text{Dir}(\alpha) \Rightarrow P(\theta|\alpha) \propto \prod_{(j=1 \text{ to } m)} \theta_j^{\alpha j-1}$
   2. $\varphi_t \mid \alpha' \sim \text{Dir}(\alpha') \Rightarrow P(\varphi|\alpha') \propto \prod_{(j=1 \text{ to } m)} \varphi_j^{\alpha' j-1}$

iv. As $\alpha \rightarrow 0$, the model biases towards states that transition to fewer states

v. As $\alpha' \rightarrow 0$, the model biases towards states that emit fewer words. This latter feature is consistent with intuition and thus useful.

vi. *Expectation Maximization:*
   1. Is an iterative procedure, estimating the parameters θ and φ at the l+1st iteration based on their values at the lth iteration.
   2. It's important to note that the l variable represents an iteration, and is different from the k variable, which represented a state at time k.
   3. Process:
      a. Initialize the parameters uniformly (e.g. P(nn|dt)=P(vb|dt)=0.5)
      b. E step: Use current estimates to adjust expectations (e.g. E[C(nn|dt)]=1.5, E[C(vb|dt)]=0.5)
      c. M step: Use expectations to re-estimate the parameters so that the new estimates maximize the log-likelihood of the E Step (e.g. P(nn|dt)=0.75, P(vb|dt)=0.25)
   4. Formulation:
      a. E step:
         i. $E[n_{t',t}] = \sum_{(i=1 \text{ to } n-1)} n_{t',t}(i)$ where $n_{t',t}$: number of times t' follows t, and n_t: number of occurrences of state t
         ii. $E[n'_{w,t}] = \sum_{(i=1 \text{ to } n-1)(wk=w)} n_t(i)$ where $n'_{w,t}$: number of times w occurs with t
         iii. $E[n_t] = \sum_{(i=1 \text{ to } n-1)} n_t(k)$ where n_t: number of occurrences of state t
      b. M step:
         i. $\theta_{t'|t}^{(l+1)} = E[n_{t',t}] / E[n_t]$
         ii. $\varphi_{w|t}^{(l+1)} = E[n'_{w,t}] / E[n_t]$
         iii. you can calculate the log likelihood and expectations in (a) and (b) using the forward-backward algorithm [$O(nm^2)$]

vii. *Variational Bayes:*
   1. Find (t,θ,φ) that minimizes -log P(w) (need negative value to be able to use Jensen's inequality properly since log is a concave function in x>0)
   2. **Q:** Is P(w) our marginalization constant? Is this how we're trying to approximate it?

3. Use the Q function to factorize -log P(w). Use Jensen's inequality:
    a. $- \log P(w) = - \log \int P(w,t,\theta,\varphi)\, dt\, d\theta\, d\varphi \leq - \int Q(t,\theta,\varphi) \log P(w,\theta,\varphi)/Q(t,\theta,\varphi)\, dt\, d\theta\, d\varphi$
    b. **Q for groups to work on:** Why are we trying to minimize -log P(w)? That's not something we're interested in. What we're really interested in is P(t,θ,φ|w) (answer in next step).
4. Try to minimize the KL distance between Q(t,θ,φ) and P(t,θ,φ|w)
    a. $-\log P(w) = -\int Q(t,\theta,\varphi) \log [P(w,t,\theta,\varphi)/Q(t,\theta,\varphi) \times Q(t,\theta,\varphi)/P(t,\theta,\varphi|w)]\, dt\, d\theta\, d\varphi = -\int Q(t,\theta,\varphi) \log [P(w,t,\theta,\varphi)/Q(t,\theta,\varphi)]\, dt\, d\theta\, d\varphi - \int Q(t,\theta,\varphi) \log [Q(t,\theta,\varphi)/P(t,\theta,\varphi|w)]\, dt\, d\theta\, d\varphi$
    b. The first phrase is called the "Variational Free Energy" and the second phrase is what makes the inequality in step 2. It is the KL distance between Q(t,θ,φ) and P(t,θ,φ|w) which we'd like to minimize. In other words we want the integral to be close to zero, thus we want: P(t,θ,φ|w) ≈ Q(t,θ,φ)
5. We'll use a mean field assumption, which allows us to factorize Q so that t does not covary with the parameters θ,φ. This factorization will make it easier for us to calculate the derivative and local minimum of the KL distance:
    a. $P(t,\theta,\varphi|w) \approx Q(t,\theta,\varphi) = Q_1(t)\, Q_2(\theta,\varphi)$
6. Now we try to find Q_1 and Q_2 by reducing the KL divergence between the desired posterior distribution and the factorized approximation.
    a. The factorization simplifies the differentiation and minimization of the KL distance: $\int Q(t,\theta,\varphi) \log [Q(t,\theta,\varphi)/P(t,\theta,\varphi|w)]\, dt\, d\theta\, d\varphi = \int Q_1(t)\, Q_2(\theta,\varphi) \log [Q_1(t)\, Q_2(\theta,\varphi)/P(t,\theta,\varphi|w)]\, dt\, d\theta\, d\varphi$
7. Good news: if the likelihood and conjugate priors (α and α´) belong to exponential families, so do the optimal Q_1 and Q_2 (**Q:** why?), thus there is a locally optimal method of calculating the parameters which is very similar to EM and has a complexity of $O(nm^2)$ likewise:
    a. $\theta_{t'|t}^{(l+1)} = \psi(E[n_{t',t}] + \alpha) / \psi(E[n_t]+m\alpha)$ where m is the number of word types
    b. $\varphi_{w|t}^{(l+1)} = \psi(E[n'_{w,t}]+\alpha') / \psi(E[n_t]+m'\alpha')$ where m´ is the number of states
    c. $\psi(x)=\Gamma'(x)/\Gamma(x)$

viii. *MCMC and Gibbs samplers*

1. MCMC methods have their roots in the Metropolis algorithm, an attempt by physicists to compute complex integrals by expressing

them as expectations for some distribution and then estimate this expectation by drawing samples from that distribution.

2. Start in an arbitrary state, use the Markov Chain to do a random walk for a while, and stop and output the current state t_i. If you use the Metropolis-Hastings algorithm, you move from one state to the next with the probability min (1, [p*(t´) q(t;t´) / p(t) q(t´;t)]).
3. MCMC produces a stream of samples from the posterior distribution P(t|w,α,α´),and eventually converges to the posterior distribution.
    a. $P(\mathbf{t}|\mathbf{w},\alpha,\alpha') \propto \int P(\mathbf{w},\mathbf{t}|\theta,\varphi)\, P(\theta|\alpha_t)\, P(\varphi|\alpha'_w)\, d\theta\, d\varphi$
    b. Because θ and φ are conjugate to dirichlet distributions, we can integrate them out to find the conditional distribution for t.
4. Gibbs sampler is similar to MCMC, but simpler. It updates each t_i by sampling from $P(t_i|\mathbf{t}_{-i})$, and only moves along a single dimension at a time.
    a. $P(t_i|\mathbf{w},\mathbf{t}_{-i},\alpha,\alpha') \propto \int P(\mathbf{w},\mathbf{t}_{-i}|\theta,\varphi)\, P(\theta|\alpha_t)\, P(\varphi|\alpha'_w)\, d\theta\, d\varphi$
5. **Q:** But isn't that what the MCMC does already? It moves per state (or per t_i)
6. Four types of Gibbs sampler:
    a. pointwise collapsed, O(nm):
        i. $P(t_i|\mathbf{w},\mathbf{t}_{-i},\alpha,\alpha') \propto$
        $[(n'_{wi,ti}+\alpha')/(n_{ti}+m'\alpha')]\ [(n_{ti,ti-1}+\alpha)/(n_{ti-1}+m\alpha)]$
        $[(n_{ti+1,ti}+I(t_{i-1}=t_i=t_{i+1})+\alpha)/(n_{t'}+I(t_{i-1}=t_i)+m\alpha)]$
    b. blocked collapsed, $O(nm^2)$: same algorithm but resample labels for all words in a sentence at a time, using forward-backward method
        i. $\theta^*_{t'|t} = (n_{t',t}+\alpha)/(n_t+m\alpha)$ where n is state-to-state transition count for other sentences in the corpus
        ii. $\varphi^*_{w|t} = (n'_{w,t}+\alpha')/(n_t+m'\alpha)$ where n´ is state-to-word emission count for other sentences in the corpus
        iii. use the above and forward-backward alg to produce t* for the words in current sentence
        iv. use Metropolis-Hastings accept/reject step to decide weather to update current state sequence (in current sentence) to t* or not.
    c. pointwise explicit, O(nm):
        i. Resample θ and φ given state-to-state transition and state-to-word emission counts
        ii. Resample each state t_i given word w_i and

neighboring states t_{i-1} and t_{i+1}

iii. $\theta_t \mid n_t, \alpha \sim Dir(n_t+\alpha)$

iv. $\varphi_t \mid n'_t, \alpha' \sim Dir(n'_t+\alpha')$

v. $P(t_i|w_i, \mathbf{t}_{-i}, \boldsymbol{\theta}, \boldsymbol{\varphi}) \propto \theta_{ti|ti-1}\ \varphi_{wi|ti}\ \theta_{ti+1|ti}$

d. blocked explicit, $O(nm^2)$:

i. same algorithm but resample labels for all words in a sentence at a time, using forward-backward method.

7. Evaluation:

a. *Variation of Information (the lower the better)*

i. The VI measure between two clusterings C (gold standard POS tags) and C´(our output) is a sum of the amount of information lost in moving from C to C´and the amount that must be gained.

ii. H(C|C´)=H(C)-I(C;C)

iii. VI(C,C´)=H(C|C´)+H(C´|C)

iv. Problem: a tagger that assigns all words the same POS tag has good VI measure (**Q:** why? A: sparsity)

v. **Q:** Isn't this a somewhat unfair metric to use, since the whole points of Variational Bayes optimize based on this metric. In other words isn't Variation of Information a metric biased towards the Variational Bayes method?

b. *Cross-Validation Accuracy (the higher the better)*

i. Map each state to the tag it co-occurs with most frequently

ii. Problem: you can achieve a perfect score by assigning each state its unique tag (**Q:** why?)

iii. Solution: divide the corpus into two equal sets, use first set to map, use second set to validate

c. *Greedy 1-to-1 Accuracy (the higher the better)*

i. At most one state can be assigned to any POS tag

ii. Haghighi and Klein (2006) propose constraining the mapping from hidden states to POS tags so that at most one hidden state maps to any POS tag. This mapping is found by greedily assigning hidden states to POS tags until either the hidden states or POS tags are exhausted (note that if the number of hidden states and POS tags differ, some will be unassigned). We call the accuracy of the POS sequence obtained using this map its *1-to-1 accuracy*.

9. Experiments:

a. 8 different combinations of hyperparameters α and α´(0.0001 to 1)

b. Data sets of sizes 24K, 120K and 1174K words

c. Different tag sets: Noah Smith's 17 tag set, Penn Treebank tag set

d. Run each setting 10 times with at least 1000 iterations

10. Observations:

a. *Steps per sample:*
    i. Pointwise samplers need O(m) steps per sample. EM, VB and blocked Gibbs need $O(m^2)$ steps per sample.
b. *Data size:*
    i. On small data sets, Bayesian estimators outperform EM and VB (less so for VB)
        1. **Q:** why?
        2. A: when data is small, priors become important. Also VB only gives an approximation which can be inaccurate with small data
    ii. On large data, EM's cross validation score is high
c. *Convergence:*
    i. VB converges faster. Larger hyperparameters cause faster convergence.
        1. **Q:** why?
        2. A: the hyperparameters specify the mobility of the sampler, allowing it to consider novel tags, but smaller ones are rather more accurate.
    ii. Gibbs:
        1. blocked converges faster than pointwise
        2. explicit converges faster than collapsed
        3. pointwise initially converges faster than blocked ones thus it might be good to begin with a pointwise sampler.
d. **Accuracy:**
    i. **Q:** some data sets proved to be sensitive to hyperparameters, what are some examples?
    ii. Another set of experiments by Johnson (2007) proved that reducing the number of hidden states in the HMM significantly improves the 1-to-1 accuracy at the cost of missing less frequent POS tags.