

Natural Language Understanding

Unsupervised Part-of-Speech Tagging

Adam Lopez

Slide credits: Sharon Goldwater and Frank Keller

April 2, 2018

School of Informatics

University of Edinburgh

alopez@inf.ed.ac.uk

Unsupervised Part-of-Speech Tagging

Background

- Hidden Markov Models

- Expectation Maximization

Bayesian HMM

- Bayesian Estimation

- Dirichlet Distribution

- Bayesianizing the HMM

- Evaluation

Reading: Goldwater and Griffiths (2007).

Background: Jurafsky and Martin Ch. 6 (3rd edition).

Unsupervised Part-of-Speech Tagging

Part-of-speech tagging

Task: take a sentence, assign each word a label indicating its syntactic category (part of speech).

Example:

NNP	NNP	,	RB	RB	,	VBZ	RB	VB
Campbell	Soup	,	not	surprisingly	,	does	n't	have

DT	NNS	TO	VB	IN	DT	NN	.
any	plans	to	advertise	in	the	magazine	.

Uses Penn Treebank PoS tag set.

The Penn Treebank PoS tagset: one common standard

DT	Determiner
IN	Preposition or subord. conjunction
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
RB	Adverb
TO	to
VB	Verb, base form
VBZ	Verb, 3rd person singular present
...	...

Total of 36 tags, plus punctuation. English-specific. (More recent: *Univer*

Most of the time, we have no supervised training data

Current PoS taggers are highly accurate (97% accuracy on Penn Treebank). But they require *manually labelled* training data, which for many major language is not available. Examples:

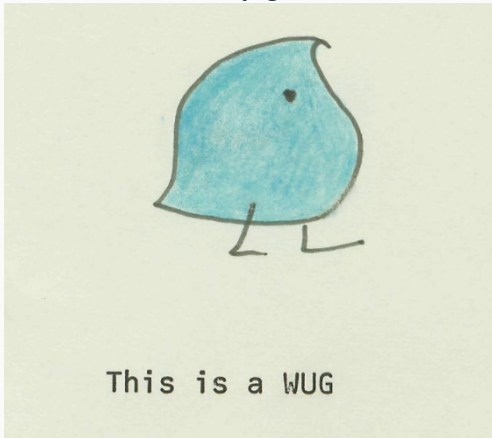
Language	Speakers
Punjabi	109M
Vietnamese	69M
Polish	40M
Oriya	32M
Malay	37M
Azerbaijani	20M
Haitian	7.7M

[From: Das and Petrov, ACL 2011 talk.]

We need models that do not require annotated training data:
unsupervised PoS tagging.

Why should unsupervised POS tagging to work at all?

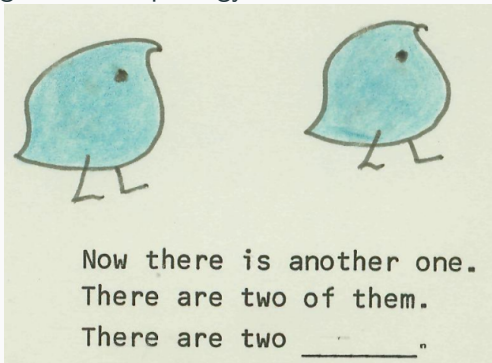
In short, because humans are very good at it. For example:



You should be able to correctly guess the PoS of “wug” even if you’ve never seen it before.

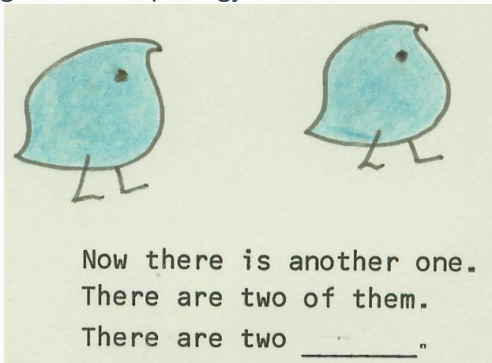
Why should unsupervised POS tagging to work at all?

You are also good at morphology:



Why should unsupervised POS tagging to work at all?

You are also good at morphology:



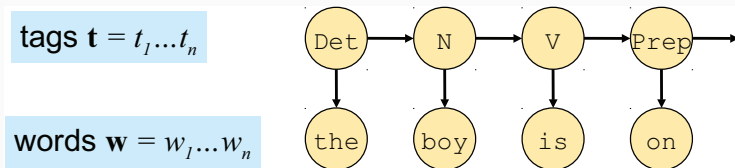
But some things are tricky:

Tom's **winning** the election was a surprise.

Background

Hidden Markov Models

All the unsupervised tagging models we will discuss are based on Hidden Markov Models (HMMs).



$$P(\mathbf{t}, \mathbf{w}) = \prod_{i=1}^n P(t_i | t_{i-1}) P(w_i | t_i)$$

The parameters of the HMM are $\theta = (\tau, \omega)$. They define:

- τ : the probability distribution over tag-tag transitions;
- ω : the probability distribution over word-tag outputs.

Hidden Markov Models

The parameters are sets of *multinomial distributions*. For tag types $t = 1 \dots T$ and word types $w = 1 \dots W$:

- $\omega = \omega^{(1)} \dots \omega^{(T)}$: the output distributions for each tag;
- $\tau = \tau^{(1)} \dots \tau^{(T)}$: the transition distributions for each tag;
- $\omega^{(t)} = \omega_1^{(t)} \dots \omega_W^{(t)}$: the output distribution from tag t ;
- $\tau^{(t)} = \tau_1^{(t)} \dots \tau_T^{(t)}$: the transition distribution from tag t .

Goal of this lecture: *introduce ways of estimating ω and τ when we have no supervision.*

Hidden Markov Models

Example: $\omega^{(NN)}$ is the output distribution for tag NN:

w

John

Mary

running

jumping

...

Hidden Markov Models

Example: $\omega^{(NN)}$ is the output distribution for tag NN:

$\omega_w^{(NN)}$	w
0.1	John
0.0	Mary
0.2	running
0.0	jumping
...	...

Hidden Markov Models

Example: $\omega^{(\text{NN})}$ is the output distribution for tag NN:

$\omega_w^{(\text{NN})}$	w
0.1	John
0.0	Mary
0.2	running
0.0	jumping
...	...

Key idea: define priors over the multinomials that are suitable for NLP tasks.

Another way to write the model, often used in statistics and machine learning:

- $t_i | t_{i-1} = t \sim \text{Multinomial}(\tau^{(t)})$
- $w_i | t_i = t \sim \text{Multinomial}(\omega^{(t)})$

This is read as: “Given that $t_{i-1} = t$, the value of t_i is drawn from a multinomial distribution with parameters $\tau^{(t)}$.”

The notation explicitly tells you how the model is parameterized, compared with $P(t_i | t_{i-1})$ and $P(w_i | t_i)$.

Inference for HMMs

For *inference* (i.e., decoding, applying the model at test time), we need to know θ and then we can compute $P(\mathbf{t}, \mathbf{w})$:

$$P(\mathbf{t}, \mathbf{w}) = \prod_{i=1}^n P(t_i | t_{i-1}) P(w_i | t_i) = \prod_{i=1}^n \tau_{t_i}^{(t_{i-1})} \omega_{w_i}^{(t_i)}$$

With this, can compute $P(\mathbf{w})$, i.e., a language model:

$$P(\mathbf{w}) = \sum_{\mathbf{t}} P(\mathbf{t}, \mathbf{w})$$

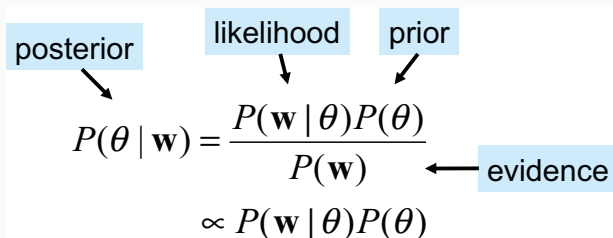
And also $P(\mathbf{t} | \mathbf{w})$, i.e., a PoS tagger:

$$P(\mathbf{t} | \mathbf{w}) = \frac{P(\mathbf{t}, \mathbf{w})}{P(\mathbf{w})}$$

Parameter Estimation for HMMs

For *estimation* (i.e., training the model, determining its parameters), we need a procedure to set θ based on data.

For this, we can rely on *Bayes Rule*:



The diagram illustrates the components of Bayes' Rule for parameter estimation. It features the equation
$$P(\theta | \mathbf{w}) = \frac{P(\mathbf{w} | \theta)P(\theta)}{P(\mathbf{w})}$$
 with a second line
$$\propto P(\mathbf{w} | \theta)P(\theta)$$
 below it. Labels in light blue boxes with arrows point to specific parts: 'posterior' points to $P(\theta | \mathbf{w})$, 'likelihood' points to $P(\mathbf{w} | \theta)$, 'prior' points to $P(\theta)$, and 'evidence' points to $P(\mathbf{w})$.

$$P(\theta | \mathbf{w}) = \frac{P(\mathbf{w} | \theta)P(\theta)}{P(\mathbf{w})}$$
$$\propto P(\mathbf{w} | \theta)P(\theta)$$

Maximum Likelihood Estimation

Choose the θ that makes the data most probable:

$$\hat{\theta} = \operatorname{argmax}_{\theta} P(\mathbf{w}|\theta)$$

Basically, we ignore the prior. In most cases, this is equivalent to assuming a uniform prior.

In supervised systems, the *relative frequency estimate* is equivalent to the maximum likelihood estimate. In the case of HMMs:

$$\tau_{t'}^{(t)} = \frac{n(t, t')}{n(t)} \qquad \omega_w^{(t)} = \frac{n(t, w)}{n(t)}$$

where $n(e)$ is the number of times e occurs in the training data.

Maximum Likelihood Estimation

In unsupervised systems, can often use the *expectation maximization* (EM) algorithm to estimate θ :

- *E-step*: use current estimate of θ to compute expected counts of hidden events (here, $n_{(t,t')}$, $n_{(t,w)}$).
- *M-step*: recompute θ using expected counts.

Examples: forward-backward algorithm for HMMs, inside-outside algorithm for PCFGs, k-means clustering.

Maximum Likelihood Estimation

Estimation Maximization sometimes works well:

- word alignments for machine translation;
- ... and speech recognition

But it often fails:

- probabilistic context-free grammars: highly sensitive to initialization; F-scores reported are generally low;
- for HMMs, even very small amounts of training data have been show to work better than EM;
- similar picture for many other tasks.

Bayesian HMM

Bayesian Estimation

We said: to train our model, we need to *estimate θ from the data*.
But is this really true?

- for language modeling, we estimate $P(w_{n+1}|\theta)$, but what we actually need is $P(w_{n+1}|\mathbf{w})$;
- for PoS tagging, we estimate $P(\mathbf{t}|\theta, \mathbf{w})$, but we actually need is $P(\mathbf{t}|\mathbf{w})$.

Bayesian Estimation

We said: to train our model, we need to *estimate θ from the data*.
But is this really true?

- for language modeling, we estimate $P(w_{n+1}|\theta)$, but what we actually need is $P(w_{n+1}|\mathbf{w})$;
- for PoS tagging, we estimate $P(\mathbf{t}|\theta, \mathbf{w})$, but we actually need is $P(\mathbf{t}|\mathbf{w})$.

So we are not actually interested in the value of θ . We could simply do this:

$$P(w_{n+1}|\mathbf{w}) = \int_{\Delta} P(w_{n+1}|\theta)P(\theta|\mathbf{w})d\theta \quad (1)$$

$$P(\mathbf{t}|\mathbf{w}) = \int_{\Delta} P(\mathbf{t}|\mathbf{w}, \theta)P(\theta|\mathbf{w})d\theta \quad (2)$$

We don't estimate θ , we integrate it out.

Bayesian Integration

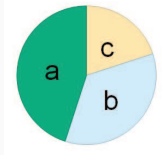
This approach is called *Bayesian integration*.

Integrating over θ gives us an *average* over all possible parameters values. Advantages:

- accounts for uncertainty as to the exact value of θ ;
- models the shape of the distribution over θ ;
- increases robustness: there may be a range of good values of θ ;
- we can use priors favoring sparse solutions (more on this later).

Bayesian Integration

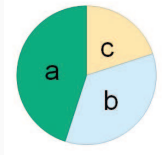
Example: we want to predict: will spinner result be “a” or not?



- Parameter θ indicates spinner result: $P(\theta = a) = .45$, $P(\theta = b) = .35$, $P(\theta = c) = .2$;
- define $t = 1$: result is “a”, $t = 0$: result is not “a”;
- make a prediction about one random variable (t) based on the value of another random variable (θ).

Bayesian Integration

Example: we want to predict: will spinner result be “a” or not?



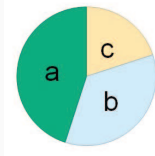
- Parameter θ indicates spinner result: $P(\theta = a) = .45$, $P(\theta = b) = .35$, $P(\theta = c) = .2$;
- define $t = 1$: result is “a”, $t = 0$: result is not “a”;
- make a prediction about one random variable (t) based on the value of another random variable (θ).

Maximum likelihood approach: choose most probable θ :

$\hat{\theta} = a$, and $P(t = 1|\hat{\theta}) = 1$, so we predict $t = 1$.

Bayesian Integration

Example: we want to predict: will spinner result be “a” or not?



- Parameter θ indicates spinner result: $P(\theta = a) = .45$, $P(\theta = b) = .35$, $P(\theta = c) = .2$;
- define $t = 1$: result is “a”, $t = 0$: result is not “a”;
- make a prediction about one random variable (t) based on the value of another random variable (θ).

Maximum likelihood approach: choose most probable θ :

$\hat{\theta} = a$, and $P(t = 1|\hat{\theta}) = 1$, so we predict $t = 1$.

Bayesian approach: average over θ :

$P(t = 1) = \sum_{\theta} P(t = 1|\theta)P(\theta) = 1(.45) + 0(.35) + 0(0.2) = .45$,
so we predict $t = 0$.

Dirichlet Distribution

Choosing the right prior can make integration easier.

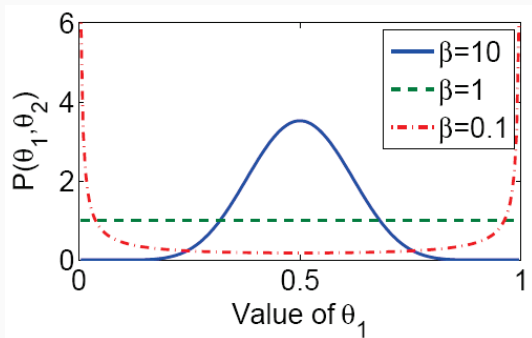
This is where the *Dirichlet distribution* comes in. A K -dimensional Dirichlet with parameters $\alpha = \alpha_1 \dots \alpha_K$ is defined as:

$$P(\theta) = \frac{1}{Z} \prod_{j=1}^K \theta_j^{\alpha_j - 1}$$

We usually only use symmetric Dirichlets, where $\alpha_1 \dots \alpha_K$ are all equal to β . We write $\text{Dirichlet}(\beta)$ to mean $\text{Dirichlet}(\beta, \dots, \beta)$.

Dirichlet Distribution

A 2-dimensional symmetric Dirichlet(β) prior over $\theta = (\theta_1, \theta_2)$:



$\beta > 1$: prefer uniform distributions

$\beta = 1$: no preference

$\beta < 1$: prefer sparse (skewed) distributions

Bayesianizing the HMM

To Bayesianize the HMM, we augment with it with symmetric Dirichlet priors:

$$t_i | t_{i-1} = t, \tau^{(t)} \sim \text{Multinomial}(\tau^{(t)})$$

$$w_i | t_i = t, \omega^{(t)} \sim \text{Multinomial}(\omega^{(t)})$$

$$\tau^{(t)} | \alpha \sim \text{Dirichlet}(\alpha)$$

$$\omega^{(t)} | \beta \sim \text{Dirichlet}(\beta)$$

To simplify things, we will present a bigram version of the Bayesian HMM; Goldwater and Griffiths use trigrams.

If we integrate out the parameters $\theta = (\tau, \omega)$, we get:

$$P(t_{n+1}|\mathbf{t}, \alpha) = \frac{n(t_n, t_{n+1}) + \alpha}{n(t_n) + T\alpha}$$

$$P(w_{n+1}|t_{n+1}, \mathbf{t}, \mathbf{w}, \beta) = \frac{n(t_{n+1}, w_{n+1}) + \beta}{n(t_{n+1}) + W_{t_{n+1}}\beta}$$

with T possible tags and W_t possible words with tag t .

We can use these distributions to find $P(\mathbf{t}|\mathbf{w})$ using an estimation method called *Gibbs sampling*.

Evaluation

Goldwater and Griffiths evaluate the BHMM in a standard experimental set-up for unsupervised PoS tagging (Merialdo, 1994):

- use a dictionary that lists possible tags for each word:

run: NN, VB, VBN

- the dictionary is actually derived from WSJ corpus;
- train and test on the unlabeled corpus (24,000 words of WSJ):

53.6% of word tokens have multiple possible tags.

Average number of tags per token = 2.3.

Goldwater and Griffiths evaluate tagging accuracy against the gold-standard WSJ tags and compare to:

- HMM with maximum-likelihood estimation using EM (MLHMM);
- Conditional Random Field with contrastive estimation (CRF/CE).

They also experiment with reducing/eliminating dictionary information.

Results

MLHMM	74.7
BHMM ($\alpha = 1, \beta = 1$)	83.9
BHMM (best: $\alpha = .003, \beta = 1$)	86.8
CRF/CE (best)	90.1

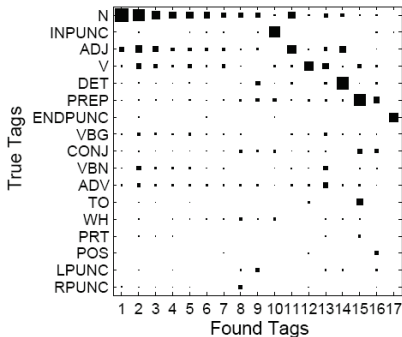
- Integrating over parameters is useful in itself, even with uninformative priors ($\alpha = \beta = 1$);
- better priors can help even more, though do not reach the state of the art.

Syntactic clustering: input are the words only, no dictionary is used:

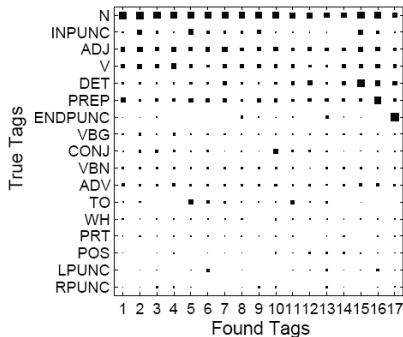
- collapse 45 treebank tags onto smaller set of 17;
- hyperparameters (α , β) are inferred automatically using Metropolis-Hastings sampler;
- standard accuracy measure requires labeled classes, so measure accuracy using best matching of classes.

Results

BHMM: 63% acc.

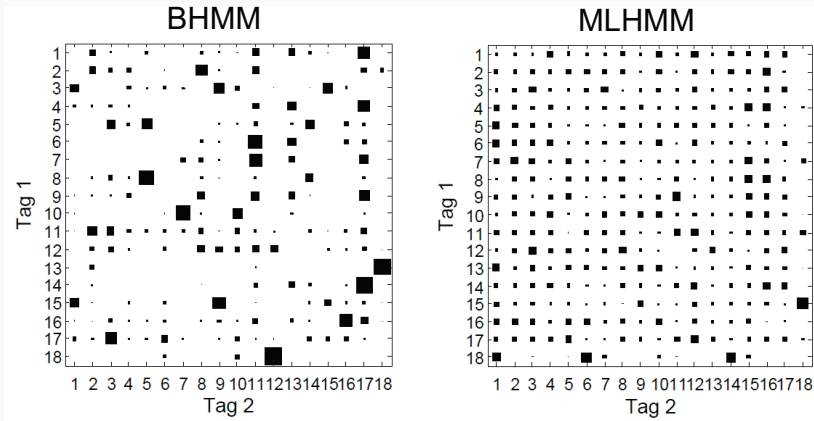


MLHMM: 35% acc.



- MLHMM groups instances of the same lexical item together;
- BHMM clusters are more coherent, more variable in size.

Results



BHMM transition matrix is sparse, MLHMM is not.

Summary

- Unsupervised PoS tagging is useful to build lexica and taggers for new language or domains;
- maximum likelihood HMM with EM performs poorly;
- Bayesian HMM with Gibbs sampling can be used instead;
- the Bayesian HMM improves performance by averaging out uncertainty;
- it also allows us to use priors that favor sparse solutions as they occur in language data.
- Other types of discrete latent variable models (e.g. for syntax or semantics) use similar methods.