This is not a machine learning course.

Grammar-based approaches to opinion mining: Part 3 (ESSLLI 2013)

Asad Sayeed

Uni-Saarland

On the menu

It's machine learning day.

- Sequence learning refresher (HMMs, CRFs, etc).
- A little bit of graphical modeling.

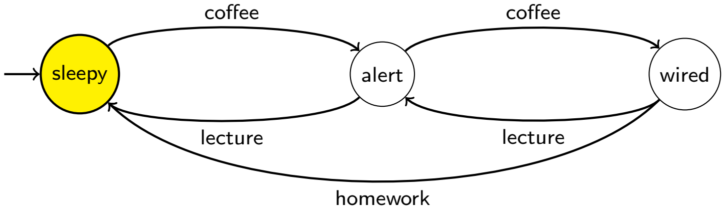
At a very high level—I assume you have basic computer science, some machine learning. Ask me to explain if something's not clear.

Disclaimer: this is not a machine learning course

- Will present only a very broad overview.
- Primarily from a “user” perspective – so the “math” will be at best shallow.
- Will focus on what people typically do with these things for sentiment analysis and other NLP tasks.

Let's talk finite state machines.

- Finite state machines represent possible sequences of strings.
    - A set of states.
    - A set of transitions between states.
    - Transitions labelled by symbols from an alphabet.
- They can be defined deterministically or non-deterministically.
- But every non-deterministic state machine can be represented deterministically.



Adding probability

Looking at it from the angle of *generating strings*:

- What strings are more likely to be generated than others?
- We can assign probabilities to transitions.
- Creates a distribution of outcomes at each step.

Markov property

Given state q_t (where t is a timestamp), state q_{t+1} is conditionally independent of $q_{t-1}, q_{t-2}, \ldots, q_0$.

From the Markov property...

We get a Markov process.

- Stochastic:

$$P(X_n|X_{n-1}, X_{n-2}, \ldots, X_0) = P(X_n|X_{n-1})$$

- Memoryless: can make predictions based on the current state as well as if we knew the entire history. (see above.)

For modelling purposes, we can assume some things to be Markov processes even when they are not.

Things that might be Markov processes

(Or we can at least pretend they are.)

- The weather.
- Lunch.
- US presidential elections.
- How I'm feeling.
- How I might **talk** about how I'm feeling...

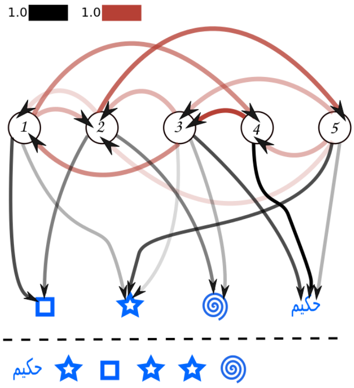
It's a two-way street

Generative models.

- What is a Markov statistical model made of?
  - The states and transitions of an FSM.
  - Probabilities on the states or transitions.
- Where do we get them from?
  - We can just assign them.
  - Or we can learn them from data.

But if we have the probability, we can use them to generate strings.

And those strings will represent a Markov process, even if they don't match the non-Markovian reality.



(Thanks, Wikipedia!)

So what does that look like in language?

- We treat human language as a potentially infinite set of strings.
- We can make a really terrible assumption about language.

A really terrible assumption about language

Human language respects the Markov property.

- Ludicrously wrong, of course, but who cares?

As though language were Markovian

- What is the FSM of a human language?
- Possible states:
    - The previous word.
    - The previous part-of-speech tag.
    - The previous word **and** POS tag combo.
    - ...
- Possible transitions:
    - The current word.
    - The word in *another* language.
    - ...

We can make things a little more complicated.

Say we wanted to model opinions as trigrams:

$$P(w_n s_n | w_{n-1} w_{n-2} s_{n-1} s_{n-2})$$

- Previous state: word (w) bigrams with word polarities (s).
  - States can represent multiple words (n-grams).
  - We can include other *features*.
- Transition: emit the next word/polarity combination.

We can make things a little more complicated.

Say we wanted to model opinions as trigrams:

$P(w_n s_n | w_{n-1} w_{n-2} s_{n-1} s_{n-2})$

- Previous state: word (w) bigrams with word polarities (s).
  - States can represent multiple words (n-grams).
  - We can include other *features*.
- Transition: emit the next word/polarity combination.

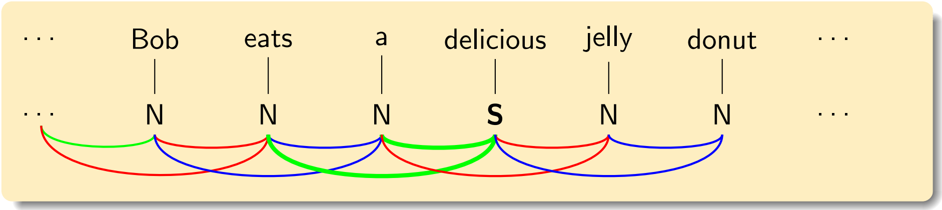
(As a side note, how plausible is this for sentiment? How would you use these probabilities?)

So how do we use Markov models in practice?

- How do we choose the probabilities?
    - We need *evidence* to decide what probabilities are on the transitions.
- But worse: we don't know the right relationship between states and outputs.
    - Language, it is big: are POS tags enough? Are bigrams of POS tags enough? (No.) To predict what?
- Not knowing state puts the "Hidden" in the Hidden Markov Model.

Does this example diagram make sense to you?

This wants to be an illustration of a trigram model. Assume we're trying to label sentiment-bearing text with (S) and non-sentiment with (N).



To save us from the ugly details...

From an application perspective, fortunately there are many tools to help us learn the models:

- Need to know or guess the number of states (e.g. # of POS tags), and then few other parameters.
- **Learning**: Maximum likelihood estimation via Baum-Welch algorithm.
    - Filtering: the "forward" algorithm gives us $P(X = q|Y_1 \dots Y_k)$ – ie, estimating prob. of states given observations.
    - Smoothing: the "forward-backward" algorithm gives us the probability of having seen a state in the past

And then we can “decode”.

What is the most likely sequence of states that led to our observations?

- (e.g., most likely set of POS tags – then we get a POS tagger).
- We need to find the the path through the FSM that gives the most likely sequence of states.
- This is known as the “Viterbi algorithm”.

Lots of tools for it

You can just search the web for a billion of these, but:

- SRI Language Modeling toolkit (srilm) – very widely used
- Cambridge HTK.
- Various packages for python, matlab, R, etc.

Not all of them were meant specifically for language.

But there are limitations. . .

. . . to which I've already alluded.

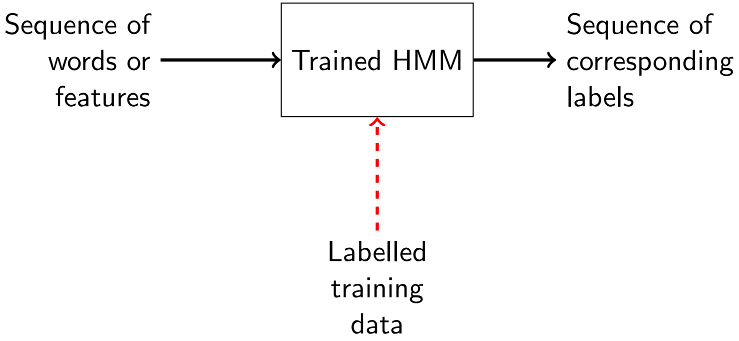
- For language, only models limited lengths of words/features.
    - But there are sometimes relationships that are larger than the *n* in our n-grams.
- Typically used to model only *sequences* – is language always so tidily sequential?
- Have to know or guess the state space.

The subtler we get, the more we want

HMMs are very successful.

- But the more we want to squeeze from language, the longer the distances in relations.
- There is a hierarchy, and it's not just grammar – from the document on down.
- So there's a progression of algorithms that do increasingly more.

For working with opinions, what you need to know:


Sequence of words or features
Trained HMM
Sequence of corresponding labels
Labelled training data

- Pretty much everything, not just HMMs.
- It's all about what features matter.

There's an assumption we've glossed over.

So we can model e.g. POS tagging via an HMM, **but**:

- HMMs contain an assumption of independence between features.
- Say we have some morphological features in the model, does that make sense?

An example

*Bob wants to return some failings tools.

We know that some morphological features can't appear in some places.

But fortunately someone fixed it for us.

Maximum-entropy Markov Models (MEMM):

- Practically, a slight variation on HMM.
  - Training and decoding are analogous.
- Predicts label based on previous label and weighted combination of features.
- ie, which class produces the most informative combination of features?
- Can be used to cover for missing labels during training.

Q: So why did I start with HMMs?

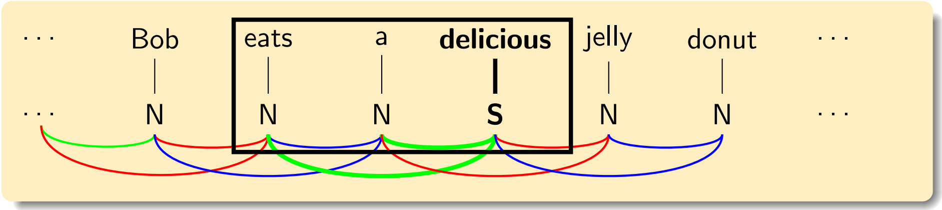
A: Because everything else can be seen as trying to escape the tyranny of the sequence.

One way to do this is to abandon the ability to generate.

Q: What is the defining feature of an HMM? (For our purposes.)

A: States representing n previous items.

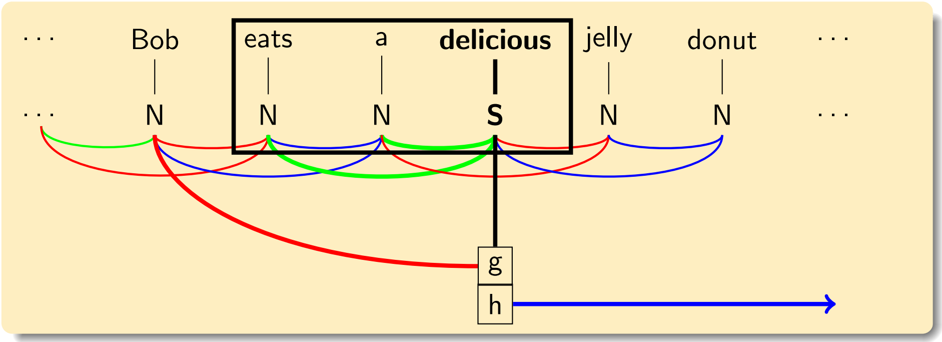
You can interpret this in terms of binary functions.



Let's define a binary function f that behaves kind of like this:

- $f($"eats a", "a delicious", "S"$) = 1$
- $f($"eats a", "a delicious", "N"$) = 0$
- $f($"Bob eats", "eats a", "N"$) = 1$
- ...

But we might actually want other kinds of functions.



Such as, say, grammar functions—or other arbitrary things.

- g(“delicious is connected to a proper name”, “S”) = 1
- h(“delicious is four words left of a preposition governing an NP with an Agent role”, “S”) = 0

When we allow this, we get. . .

. . . linear-chain Conditional Random Fields (CRFs).

- Allow arbitrary functions that can expect the input anywhere (with respect to current state.
- Goal of training: find weights on these binary functions to obtain “clique potentials” – the “energy” states of configurations of the model.
- But in practice it looks rather like training an HMM.
- No longer generative – why?

But we don't have to stay linear.

- In fact, HMMs can be thought of as a special case of CRF.
- Now that we can go beyond sequence features, we can get features based on tree and graph structures.
- Tools: again, lots of them. GRMM, CRF++.

But there's another angle on all of this.

Abandon the sequence entirely.

- Get back to generativeness.
- How is text produced?
  - Everything is in a context.
  - Those contexts come from further contexts – not just the neighbours.

The true story

- It's not just grammatical context (beyond Markovian, even).
- We return once again to *people* and *motives* for text.

Now it gets complicated to draw. . .

. . . and so I will shamelessly copypasta from the literature.

Fang et al. (2012)

- "Contrastive Opinion Modeling"
  - Given a query and some documents, discover individual perspectives on the query topic.
  - Quantify the differences in opinion.
- Bayesian approach – topic modeling
  - Assume that the words in the documents are generated by some unknown distribution of unknown topics.
  - Sample the data in order to infer the distribution, based on some assumptions about how the data came to exist.

Their theory is represented by a convenient graphical model.

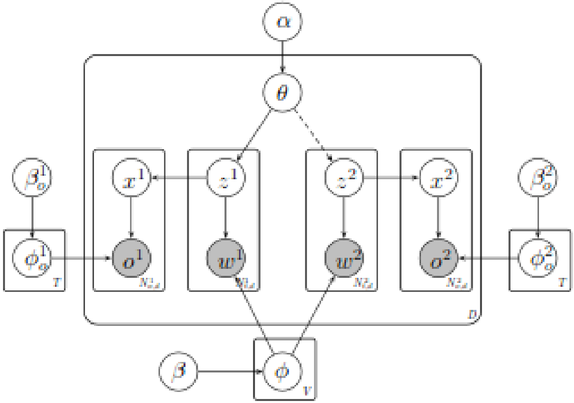
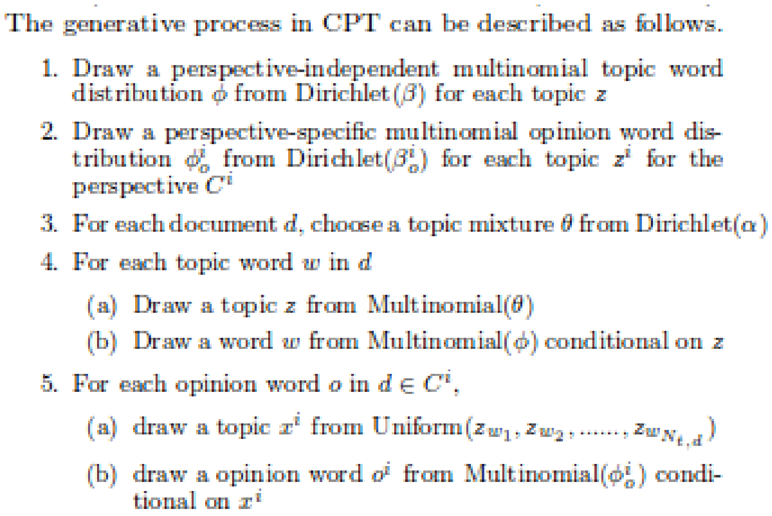


Figure 1: The plate notation of the Cross-Perspective Topic model. The shaded nodes are observed variables.

And their model has a handy little "story".



The generative process in CPT can be described as follows.

1. Draw a perspective-independent multinomial topic word distribution ϕ from Dirichlet(β) for each topic z
2. Draw a perspective-specific multinomial opinion word distribution ϕ_o^i from Dirichlet(β_o^i) for each topic z^i for the perspective C^i
3. For each document d, choose a topic mixture θ from Dirichlet(α)
4. For each topic word w in d
   (a) Draw a topic z from Multinomial(θ)
   (b) Draw a word w from Multinomial(ϕ) conditional on z
5. For each opinion word o in $d \in C^i$,
   (a) draw a topic x^i from Uniform$(z_{w_1}, z_{w_2}, \ldots\ldots, z_{w_{N_{t,d}}})$
   (b) draw a opinion word o^i from Multinomial(ϕ_o^i) conditional on x^i

So what did they do?

Contrastive opinions are relatively easy to find.

- Statement records of US senators (Democrat vs. Republican)
- News sources from US, India, China from Jan 2009 to Dec 2010.

Sample the data via "Gibbs sampling"

- "Markov chain Monte Carlo" procedure.
- Iteratively sample the distribution.
  - Pick variable x_i, find $p(x|X_{-i})$ in the data.
  - Put x_i back, pick next i.
  - Shake vigorously.

The sampler requires sampling equations.

- Sampling equation of the topic variable z for each topic word w_i:

$$p(z_i = k | w_i = v, \mathbf{z}_{-i}, \mathbf{w}_{-i}, \alpha, \beta)$$

$$\propto \frac{n_{kd,-i} + \alpha}{\sum_{k=1}^{K} n_{kd,-i} + K\alpha} \times \frac{n_{vk} + \beta}{\sum_{v=1}^{V} n_{vk} + V\beta}$$

- Sampling equation of the opinion topic variable x^1 in the perspective C^1 (the similar equation can be derived for C^2):

$$p(x_i^1 = s | o_i = r, \mathbf{x}_{-i}^1, \mathbf{o}_{-i}, \beta, \beta_o)$$

$$\propto \frac{n_{rs,-i} + \beta_o^1}{\sum_{r=1}^{T} n_{rs,-i} + T\beta_o^1} \times \frac{n_{sd}}{N_{t,d}}$$

Which in turn make use of these variables

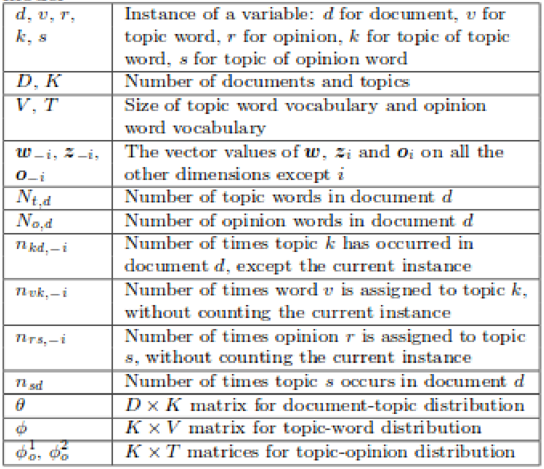
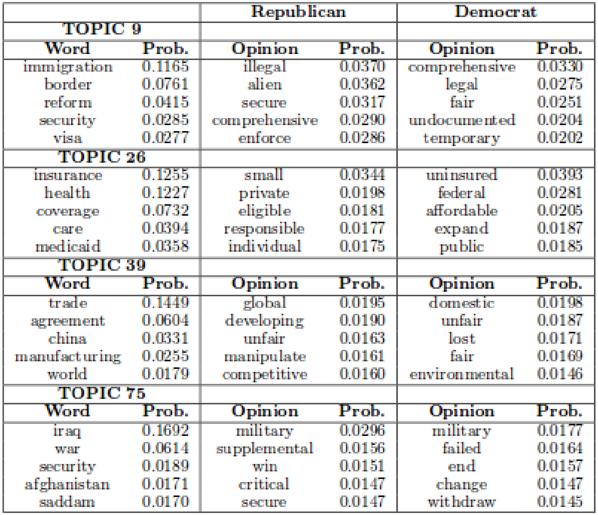


Table 1: Notations in the Cross-Perspective Topic model

d, v, r, k, s	Instance of a variable: d for document, v for topic word, r for opinion, k for topic of topic word, s for topic of opinion word
D, K	Number of documents and topics
V, T	Size of topic word vocabulary and opinion word vocabulary
$\mathbf{w}_{-i}$, $\mathbf{z}_{-i}$, $\mathbf{o}_{-i}$	The vector values of $\mathbf{w}$, $\mathbf{z}_i$ and $\mathbf{o}_i$ on all the other dimensions except i
$N_{t,d}$	Number of topic words in document d
$N_{o,d}$	Number of opinion words in document d
$n_{kd,-i}$	Number of times topic k has occurred in document d, except the current instance
$n_{vk,-i}$	Number of times word v is assigned to topic k, without counting the current instance
$n_{rs,-i}$	Number of times opinion r is assigned to topic s, without counting the current instance
n_{sd}	Number of times topic s occurs in document d
θ	$D \times K$ matrix for document-topic distribution
ϕ	$K \times V$ matrix for topic-word distribution
ϕ_o^1, ϕ_o^2	$K \times T$ matrices for topic-opinion distribution

Q: So, does it work?

A: You actually get coherent topics from it!



		Republican		Democrat	
TOPIC 9					
Word	Prob.	Opinion	Prob.	Opinion	Prob.
immigration	0.1165	illegal	0.0370	comprehensive	0.0330
border	0.0761	alien	0.0362	legal	0.0275
reform	0.0415	secure	0.0317	fair	0.0251
security	0.0285	comprehensive	0.0290	undocumented	0.0204
visa	0.0277	enforce	0.0286	temporary	0.0202
TOPIC 26					
insurance	0.1255	small	0.0344	uninsured	0.0393
health	0.1227	private	0.0198	federal	0.0281
coverage	0.0732	eligible	0.0181	affordable	0.0205
care	0.0394	responsible	0.0177	expand	0.0187
medicaid	0.0358	individual	0.0175	public	0.0185
TOPIC 39					
Word	Prob.	Opinion	Prob.	Opinion	Prob.
trade	0.1449	global	0.0195	domestic	0.0198
agreement	0.0604	developing	0.0190	unfair	0.0187
china	0.0331	unfair	0.0163	lost	0.0171
manufacturing	0.0255	manipulate	0.0161	fair	0.0169
world	0.0179	competitive	0.0160	environmental	0.0146
TOPIC 75					
Word	Prob.	Opinion	Prob.	Opinion	Prob.
iraq	0.1692	military	0.0296	military	0.0177
war	0.0614	supplemental	0.0156	failed	0.0164
security	0.0189	win	0.0151	end	0.0157
afghanistan	0.0171	critical	0.0147	change	0.0147
saddam	0.0170	secure	0.0147	withdraw	0.0145

This is all just to get you into the right mindset.

Next couple of days we'll try to apply this mindset.