Ambiguity Resolution: Statistical Method

Prof. Ahmed Rafea

Outline

- Estimating Probability
- Part of Speech Tagging
- Obtaining Lexical Probability
- Probabilistic Context-free Grammars
- Best First Parsing

Estimating Probability

- Example : Having corpus having 1,273,000 words. Say we find 1000 uses of the word flies, 400 is N sense, and 600 in the V sense. Then we can have the following probabilities:
  - Prob(flies) = 1000/1,273,000 = .0008
  - Prob(flies & V) = 600/ 1,273,000 = .0005
  - Prob(V|flies)= .0005/.0008 = .625
- This is called maximum likelihood estimator(MLE)
- In NL application we may have *sparse data* which means that some words may have 0 probability. To solve this problem we may add small amount say .5 to every count. This is called expected likelihood estimator (ELE)
- If a word w occurred 0 times in 40 classes (L_1,….L_{40}) then using ELE Prob(L_i|w) will be 0.5/0.5*40= .025 otherwise this probability cannot be estimated. If w appears 5 times once as a verb and 4 times as noun then using MLE Prob(N|w)= .8 and using ELE it will be 4.5/25= .18

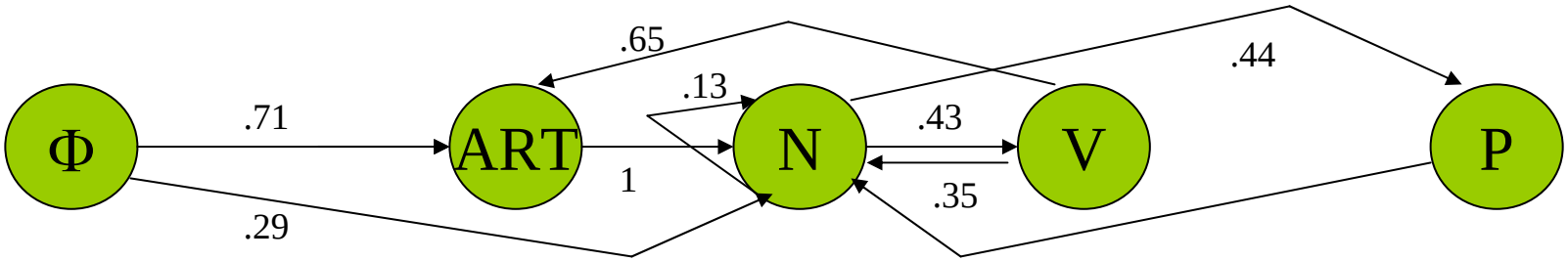
Part of Speech Tagging(1)

- Simple algorithm is to estimate the category of the word using the probability obtained from the training corpus as indicated above
- To improve reliability local context may be used as follows:
  - Prob(c_1, …c_t|w_1, …w_t), large data, not possible
  - Prob(c_1, ..c_t)* Prob(w_1,..w_t|c_1, ..c_t)/Prob(w_1, ..$w_{t)}$ Bay Rule
  - Prob(c_1, ..c_t)* Prob(w_1,..w_t|c_1, ..c_t), denominator will not affect the answer
  - $\Pi_{i=1,T}$ Prob(c_i|c_{i-1})*Prob(w_i|c_i) by approximation of Prob(c_1, ..c_t) to be the product of the bi-gram probability and the Prob(w_1,..w_t|c_1, ..c_t), to be the product of the probability that each word occurs in the indicated part of speech

Part of Speech Tagging(1)

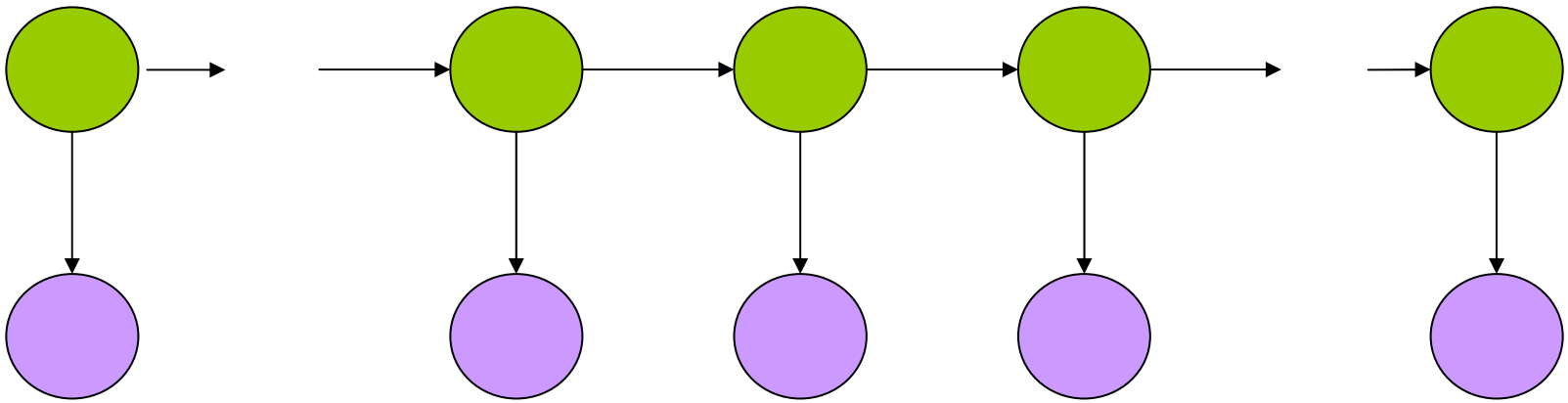
- Given all these probabilities estimates, how might you find the sequence of categories that has the highest probability of generating a specific sentence?
- The brute force method can generate N^T possible sequence where N is the number of categories and T is the number of words
- We can use Markov chain which is a special form of probabilistic finite state machine, to compute the bi-gram probability the $\Pi_{i=1,T}$ Prob($c_i|c_{i-1}$)

Markov Chain



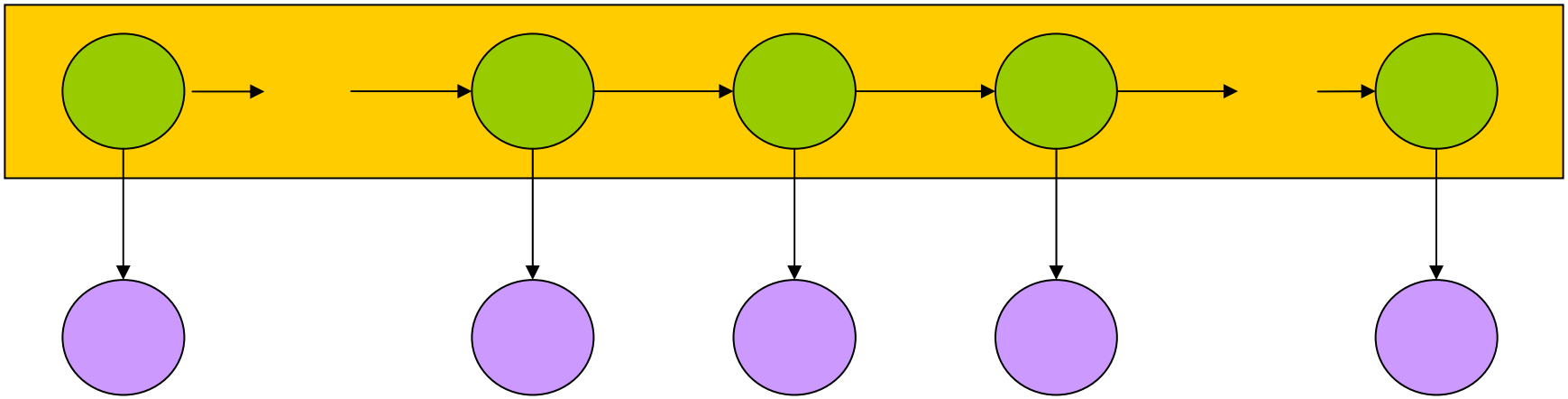
A Markov Chain capturing the bi-gram probabilities

What is an HMM?



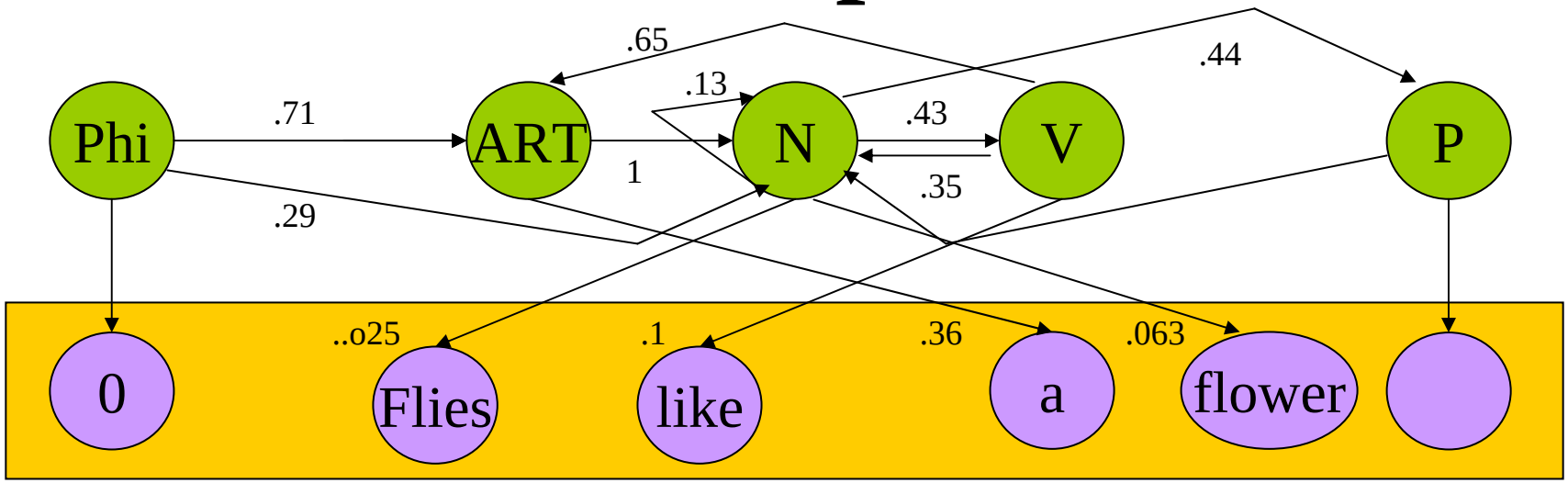
- Graphical Model
- Circles indicate states
- Arrows indicate probabilistic dependencies between states

What is an HMM?



- Green circles are ***hidden states***
- Dependent only on the previous state

Example



- Purple nodes are ***observed states***
- Dependent only on their corresponding hidden state
- Example: Flies like a flower N V ART N
  - $Prob(w_1,\ldots\ldots w_T|c_1,\ldots\ldots c_T) = \Pi_{i=1,T}\ Prob(c_i|c_{i-1})*Prob(w_i|c_i)$
    = (.29*.43*.65*1)*(.025*.1*.36*.063)= 0.081*0.0000567= 0.0000045927

Viterbi Algorithm

	Flies	like	a	flower
V	.0000076	***.000312***	0	.0000000026
N	***.00725***	.000013	.00000012	***.0000043***
P	0	.00022	0	0
ART	0	0	***.000072***	0

Obtaining Lexical Probability

- Context Independent probability of w
  - $Prob(L_j,w)= count(L_j \& w)/\Sigma_{i=1,N}\ count(L_i \& w)$
- This estimate is not reliable because it does not take context into account
- Example for taking context into account:

The flies like flowers

Prob(flies/N|The flies)= Prob(flies/N&The flies)/Prob(The flies)

Prob(flies/N&Theflies)=Prob(the|ART)*Prob(flies|N)*Prob(ART|Φ)Prob(N|ART)+
Prob(the|N)*Prob(flies|N)*Prob(N|Φ)Prob(N|N)+
Prob(the|P)*Prob(flies|N)*Prob(P|Φ)Prob(N|P)

Prob(The flies)= Prob(flies/N & The flies)+Prob(flies/V & The flies)

(see page 206 for numeric values)

Forward Probability

- $\alpha_i(t) = Prob(w_t/L_i, w_1, \ldots w_t)$

  e.g. with the sentence The flies like flowers $\alpha_2(3)$ would be the sum of values computed for all sequences ending in V (2^{nd} category) in position 3 given the input *The flies like.*

- Using conditional probability
  - $Prob(w_t/L_i|w_1,\ldots w_t) = prob(w_t/L_i, w_1,\ldots w_t)/Prob(w_1,\ldots w_t)$

$$= \alpha_i(t) / \Sigma_{j=1,N}\, \alpha_j(t)$$

Backward Probability

- $\beta_i(t)$ is the probability of producing the sequence w_t,….. w_T beginning from state w_t/L_j
- A better method of estimating the lexical probability for word w_t would be to consider the entire sentence:
  - $\text{Prob}(w_t/L_i) = (\alpha_i(t)*\beta_i(t))/\Sigma_{j=1,N}(\alpha_j(t)*\beta_j(t))$

Probabilistic Context Free Grammar

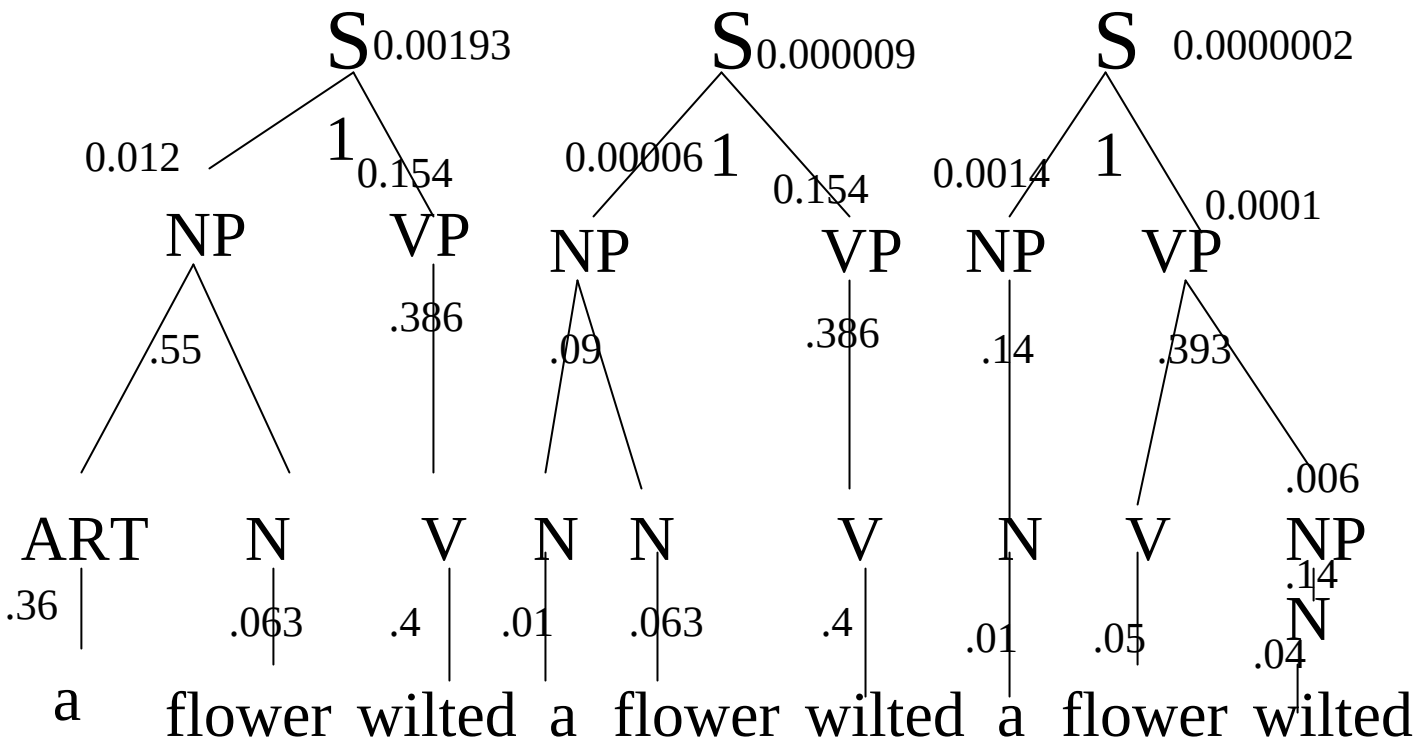
- $Prob(R_j|C) = Count(\#times\ R_j\ used)/\Sigma_{i=1,m}(\#times\ R_j\ used)$
  Where the grammar contains m rules: R_1, …. R_m with the left hand side C
- Parsing is to find the most likely parse tree that could have generated a sentence
- Independent assumption should be made about rule use, e.g. NP rules probabilities are the same whether the NP is a subject, the object of a verb, or the object of a preposition.
- Inside probability which is the probability that a constituent C generates a sequence of words w_i, w_{i+1},…. w_j ($w_{i,j}$) : $Prob(w_{i,j})|C)$
- Example the inside probability of the NP *a flower (*using Rule 6 and Rule 8 in Grammar 7.17 page 209) is given by

  Prob(a flower| NP)= Prob(R8|NP)*Prob(a|ART)*Prob(flower|N)+
  Prob(R6|NP)*Prob(a|N)*Prob(flower|N)

Example of a PCFG

	Rule	Count of LHS	Count of Rule	Probability
1.	S → NP VP	300	300	1
2.	VP → V	300	116	.386
3.	VP → V NP	300	118	.393
4.	VP → V NP PP	300	66	.22
5.	NP → NP PP	1023	241	.24
6.	NP → N N	1023	92	.09
7.	NP → N	1023	141	.14
8.	NP → ART N	1023	558	.55
9.	PP → P NP	307	307	1

Example of PCFG Parse Trees



Best First Parsing

- Best First parsing leads to significant improvement in efficiency
- One implementation problem is that if you use multiplicative method to combine the scores , the scores of constituent tend to fall quickly and consequently the search will be like breadth first search. Some algorithms use a different function to compute the score for constituents such as

  Score (C) = Min (Score (C→ C1,…Cn), Score(C1)… Score (Cn))