T-(538|725)-MALV, Natural Language Processing
POS tagging – with statistics

Hrafn Loftsson[1] Hannes Högni Vilhjálmsson[1]

[1]School of Computer Science, Reykjavik University

October 2009

Outline

Outline

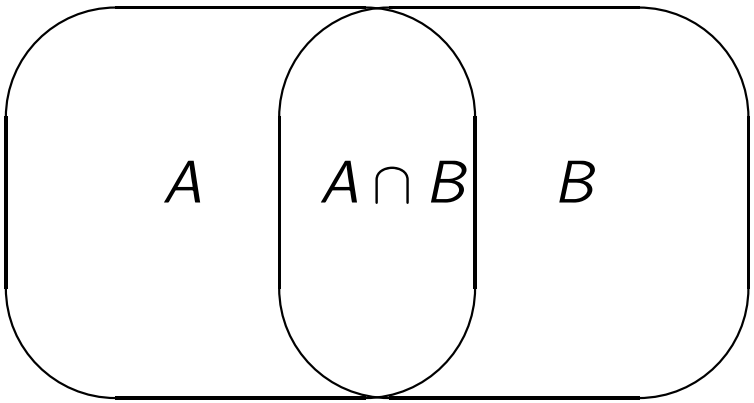
Conditional probabilities

The probability of some event A, given the occurrence of some other event B.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{1}$$

Bayes rule:

$$P(A \cap B) = P(B)P(A|B) = P(A)P(B|A) \tag{2}$$



The most probable sequence of tags

- A tag sequence: $T = t_1, t_2, \ldots, t_n$
- A word sequence: $W = w_1, w_2, \ldots, w_n$
- We want to find the tag sequence $\hat{\mathrm{T}}$, which maximises: $P(T|W)$
  - $\hat{\mathrm{T}} = max_{t_1, t_2, \ldots t_n}\ P(t_1, t_2, \ldots t_n | w_1, w_2, \ldots w_n)$
- According to Bayes rule: $\mathrm{P(W)P(T|W) = P(T)P(W|T)}$
- $$\hat{\mathrm{T}} = \arg\max_T \quad \frac{P(T)P(W|T)}{P(W)} \tag{3}$$
- For a particular word sequence, P(W) is a constant $\Rightarrow$
$$\hat{\mathrm{T}} = \arg\max_T \quad P(T)P(W|T) \tag{4}$$

Trigram estimation for the tag sequence, P(T)

$$P(T) = P(t_1, t_2, \ldots, t_n) \approx P(t_1)P(t_2|t_1)\prod_{i=3}^{n} P(t_i|t_{i-2}, t_{i-1}) \quad (5)$$

These probabilities are estimated from **maximum likilehoods** (MLE):

$$P_{MLE}(t_i|t_{i-2}, t_{i-1}) = \frac{C(t_{i-2}, t_{i-1}, t_i)}{C(t_{i-2}, t_{i-1})} \quad (6)$$

Training corpora are not large enough $\Rightarrow$ frequency counts are missing. Therefore, we often use linear interpolation:

$$P_{LinearInter}(t_i|t_{i-2}, t_{i-1}) = \lambda_1 P(t_i|t_{i-2}, t_{i-1}) + \lambda_2 P(t_i|t_{i-1}) + \lambda_3 P(t_i) \quad (7)$$

where $0 \leq \lambda_i \leq 1$ and $\sum_{i=1}^{3} \lambda_i = 1$

Estimation for P(W|T)

$$P(W|T) = P(w_1, w_2, \ldots, w_n|t_1, t_2, \ldots, t_n) \approx \prod_{i=1}^{n} P(w_i|t_i) \quad (8)$$

Each $P(w_i|t_i)$ is then estimated using MLE:

$$P_{MLE}(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)} = \frac{C(w_i, t_i)}{C(t_i)} \quad (9)$$

The final formula

This formula:

$$\hat{\mathrm{T}} = \arg\max_T \quad P(T)P(W|T) \tag{10}$$

. . . was simplified and the result is the tag sequence $\hat{\mathrm{T}}$ which maximises:

$$P(t_1)P(t_2|t_1)\prod_{i=3}^{n} P(t_i|t_{i-2}, t_{i-1})\prod_{i=1}^{n} P(w_i|t_i) \tag{11}$$

This is a **trigram model** for POS tagging.

Outline

A Markov model

- Used to describe a sequence of random variables, which are dependent in some way.
- For example, when the value of one random variable depends on the value of a previous random variable in the sequence.

Markov chain

- A sequence of random variables $\{X_1, X_2, \ldots, X_T\}$, X_t denotes a random variable at time t.
- Have their values in a finite set of states $\{q_1, q_2, \ldots, q_n\}$, the state space.
- Markov chains define random transitions from one state to another one. Can be represented as a probabilistic or weighted automata.

A Markov model

The Markov condition

- **Limited history**. The current state (value) depends only on a constant number of previous states.
  - E.g., "First order model":

$$P(X_t = q_j | X_1, X_2, \ldots, X_{t-1}) = P(X_t = q_j | X_{t-1})$$

  - E.g., "Second order model":

$$P(X_t = q_j | X_1, X_2, \ldots, X_{t-1}) = P(X_t = q_j | X_{t-2}, X_{t-1})$$

- **Independent of time; stationary**.
  - E.g., "First order model":

$$P(X_t = q_j | X_{t-1} = q_i) = a_{ij}$$

A Markov model and finite-state automata

- A Markov model can be viewed as a finite-state automaton with transition probabilities (a_{ij}) associated with each arc; see the bigram figure on the next slide.
- There are no *long distance dependencies* and the next state is only dependent on the previous state.
- In a *Visible Markov model* we know through which states the automaton traverses.

$P(X_1, X_2, \ldots, X_T) = P(X_1) * P(X_2|X_1) * P(X_3|X_2, X_1), \ldots,$
$P(X_T|X_1, X_2, \ldots, X_{T-1})) =$
$P(X_1) * P(X_2|X_1) * P(X_3|X_2), \ldots, P(X_T|X_{T-1})$
(limited history; first order)

A Markov chain

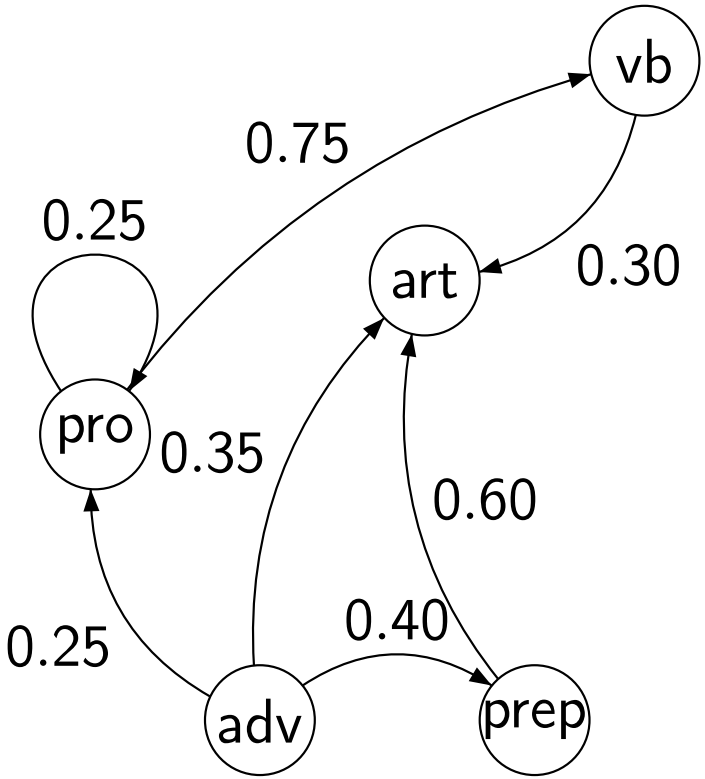


Figure: A Markov chain representing bigram probabilities.

Hidden Markov Model (HMM)

- In the case of POS tagging we represent tags as states in the automaton.
- The exact sequence of states is however not known (hidden), because we only know the words.
- But what about the words themselves?
  - Each states emits a word with certain emission probability.
- The HMM is then represented with underlying hidden states (tags) which generate (emit) words according to some probabilites.
- P(T): transition probabilities (í. færslulíkur)
- P(W|T): emission probabilities (í. frálagslíkur)

Hidden Markov Model (HMM)

- We view a sequence of tags in text as an HMM.
- We assume that the tag for a particular word only depends on the previous one or two tags (limited horizon)
- And this is independent of time (stationary)
  - For example, if the probability of observing a verb following a pronoun in the beginning of a sentence is 0.2, then this probability does not change when the rest of the sentence is tagged, or when a new sentence is tagged.
- The goal is to find the most probable sequence of tags for a particular sequence of words.
- In other words, to find the most probable sequence of states which the model goes through for a particular sequence of words.

Outline

How do we find the most probable tag sequence?

$$P(t_1)P(t_2|t_1)\prod_{i=3}^{n}P(t_i|t_{i-2},t_{i-1})\prod_{i=1}^{n}P(w_i|t_i)$$

Calculate the probabilities for all possibilities of $t_1, t_2, \ldots, t_n$?

- Inefficient, because the number of possibilities can be very large.
- The time complexity is (worst case) n^T, where T is the number of tags in the tagset.

Therefore, dynamic programming (í. kvik bestun) is usually applied – **Viterbi** algorithm.

The Viterbi algorithm

- Instead of checking all possible paths in the automaton to calculate the optimal state sequence ...
- the Viterbi algorithm determines the optimal subpaths for each state in the automaton while it traverses the automaton.
- The subpaths that are not most probable are discarded.
- See Fig. 7.2 page 166.

Outline

A trigram tagger

- Is based on an HMM.
- States stands for a pair of tags.
- Transition probabilitiy: The probability of a particular tag given the two preceeding tags.
  - transition/contextual probabilities
- Emission probability: The probability of a word given particular tags.
  - emission/lexical probabilities
- The probabilities are estimated from a training corpus using MLE.
- The tag sequence $t_1, t_2, \ldots, t_n$ is found which maximises:

$$P(t_1)P(t_2|t_1)\prod_{i=3}^{n} P(t_i|t_{i-2}, t_{i-1})\prod_{i=1}^{n} P(w_i|t_i)$$

A trigram tagger – Unknown words

- How are unknown words handled, i.e. words that are not found during training?
- One method is to perform ending analysis.
- A probability distribution is derived for a particular ending by using words in the training corpus having the same ending of some maximum length (e.g. 5)
- $P(w_i|t_i)$ then becomes $P(e_i|t_i)$, where e_i stands for the ending of word i of some given length.

A trigram tagger – TnT

- TnT – Trigrams 'n Tags
- Brants 2000.
- An efficent and effective trigram tagger based on a HMM (implemented in C)
- Implemented in the manner we have discussed.
- You will experiment with a reimplementation of it (in Java) in Assignment II.