
Advanced Natural Language Processing

Lecture 8

Maximum Entropy Models

Frank Keller (slides by Philipp Koehn)

7 October 2011



Tagging as supervised learning

- Tagging is a supervised learning problem
 - given: some annotated data (words annotated with POS tags)
 - build model (based on features, i.e. representation of example)
 - predict unseen data (POS tags for words)
- Issues in supervised learning
 - there is no data like more data
 - feature engineering: how best represent the data
 - overfitting to the training data?
- There are many algorithms for supervised learning (naive Bayes, decision trees, maximum entropy, neural networks, support vector machines, ...)

One tagging method: Hidden Markov Models

- HMMs make use of two conditional probability distributions
 - tag sequence model $p(t_n | t_{n-2}, t_{n-1})$
 - tag-word prediction model $p(w_n | t_n)$
- Given these models, we can find the best sequence of tags for a sentence using the Viterbi algorithm

How good is HMM tagging?

- Labeling a sequence is very fast
- Viterbi algorithm outputs best label sequence (previous tags affect labeling of next tag), not just best tag for each word in isolation
- It is easy to get 2nd best sequence, 3rd best sequence, etc.
- But: uses only a very small window around word (n previous tags)

More features

- Consider a larger window

w_{n-4}	w_{n-3}	w_{n-2}	w_{n-1}	w_n	w_{n+1}	w_{n+2}	w_{n+3}	w_{n+4}
t_{n-4}	t_{n-3}	t_{n-2}	t_{n-1}	t_n	t_{n+1}	t_{n+2}	t_{n+3}	t_{n+4}

- Examples for useful features

- if one of the previous tags is **MD**, then **VB** is likelier than **VBP** (basic verb form instead of verb in singular present)
- if next tag is **JJ**, then **RBR** is likelier than **JJR** (adverb instead of adjective)

More features (2)

- Lexical features
 - if one of the previous tags is **not**, then **VB** is likelier than **VBP**
- Morphological features
 - if word ends in **-tion** it is most likely an **NN**
 - if word ends in **-ly** it is most likely an adverb

Using additional features

- Using more features in a conditional probability distribution?

$$p(t_i | w_i, f_0, \dots, f_n)$$

⇒ sparse data problems
(insufficient statistics for reliable estimation of the distribution)

Defining Features

- Each tagging decision for a word occurs in a specific context
- For tagging, we consider as context the history h_i
 - the word itself
 - morphological properties of the word
 - other words surrounding the word
 - previous tags
- We can define a feature f_j that allows us to learn how well a specific aspect of the history h_i is associated with a tag t_i

Defining Features (2)

- We observe in the data patterns such as:

the word *like* has in 50% of the cases the tag *VB*

- Previously, in HMM models, this led us to introduce probabilities (as part of the tag sequence model) such as

$$p(VB|like) = 0.5$$

Defining Features (3)

- In a maximum entropy model, this information is captured by a feature

$$f_j(h_i, t_i) = \begin{cases} 1 & \text{if } w_i = \text{like and } t_i = VB \\ 0 & \text{otherwise} \end{cases}$$

- The importance of a feature f_j is defined by a parameter λ_j

Defining Features (4)

- Features may consider morphology

$$f_j(h_i, t_i) = \begin{cases} 1 & \text{if suffix}(w_i) = \text{"ing"} \text{ and } t_i = VB \\ 0 & \text{otherwise} \end{cases}$$

- Features may consider tag sequences

$$f_j(h_i, t_i) = \begin{cases} 1 & \text{if } t_{i-2} = DET \text{ and } t_{i-1} = NN \text{ and } t_i = VB \\ 0 & \text{otherwise} \end{cases}$$

Features in Ratnaparkhi [1996]

frequent w_i	$w_i = X$
rare w_i	X is prefix of w_i , $ X \leq 4$
	X is suffix of w_i , $ X \leq 4$
	w_i contains a number
	w_i contains uppercase character
	w_i contains hyphen
all w_i	$t_{i-1} = X$
	$t_{i-2}t_{i-1} = XY$
	$w_{i-1} = X$
	$w_{i-2} = X$
	$w_{i+1} = X$
	$w_{i+2} = X$

Log-linear model

- Features f_j and parameters λ_j are used to compute the probability $p(h_i, t_i)$:

$$p(h_i, t_i) = \prod_j \lambda_j^{f_j(h_i, t_i)}$$

- These types of models are called log-linear models, since they can be reformulated into

$$\log p(h_i, t_i) = \sum_j f_j(h_i, t_i) \log \lambda_j$$

- There are many learning methods for these models, maximum entropy is just one of them

Conditional probabilities

- We defined a model $p(h_i, t_i)$ for the joint probability distribution for a history h_i and a tag t_i
- Conditional probabilities can be computed straightforwardly as

$$p(t_i|h_i) = \frac{p(h_i, t_i)}{\sum_{i'} p(h_i, t_{i'})}$$

Tagging a sequence

- We want to tag a sequence w_1, \dots, w_n
- This can be decomposed into:

$$p(t_1, \dots, t_n | w_1, \dots, w_n) = \prod_{i=1}^n p(t_i | h_i)$$

- The history h_i consist of all words w_1, \dots, w_n and previous tags t_1, \dots, t_{i-1}
- We cannot use Viterbi search \Rightarrow heuristic beam search is used (more on beam search in a future lecture on machine translation)

Questions for training

- Feature selection
 - given the large number of possible features, which ones will be part of the model?
 - we do not want redundant features
 - we do not want unreliable and rarely occurring features (avoid overfitting)
- Parameter values λ_j
 - λ_j are positive real numbered values
 - how do we set them?

Feature selection

- Feature selection in Ratnaparkhi [1996]
 - Feature has to occur 10 times in the training data
- Other feature selection methods
 - use features with high mutual information
 - add feature that reduces training error most, retrain

Setting the parameter values λ_j : Goals

- The empirical expectation of a feature f_j occurring in the training data is defined by

$$\tilde{E}(f_j) = \frac{1}{n} \sum_{i=1}^n f_j(h_i, t_i)$$

- The model expectation of that feature occurring is

$$E(f_j) = \sum_{h,t} p(h, t) f_j(h, t)$$

- We require that $\tilde{E}(f_j) = E(f_j)$

Empirical expectation

- Consider the feature

$$f_j(h_i, t_i) = \begin{cases} 1 & \text{if } w_i = \text{like and } t_i = VB \\ 0 & \text{otherwise} \end{cases}$$

- Computing the empirical expectation $\tilde{E}(f_j)$:
 - if there are 10,000 words (and tags) in the training data
 - ... and the word **like** occurs with the tag **VB** 20 times
 - ... then

$$\tilde{E}(f_j) = \frac{1}{n} \sum_{i=1}^n f_j(h_i, t_i) = \frac{1}{10000} \sum_{i=1}^{10000} f_j(h_i, t_i) = \frac{20}{10000} = 0.002$$

Model expectation

- We defined the model expectation of a feature occurring as

$$E(f_j) = \sum_{h,t} p(h,t) f_j(h,t)$$

- Practically, we cannot sum over all possible histories h and tags t
- Instead, we compute the model expectation of the feature on the training data:

$$E(f_j) \approx \frac{1}{n} \sum_{i=1}^n p(t|h_i) f_j(h_i, t)$$

Note: theoretically we have to sum over all t , but $f_j(h_i, t) = 0$ for all but one t

Goals of maximum entropy training

- Recap: we require that $\tilde{E}(f_j) = E(f_j)$, or

$$\frac{1}{n} \sum_{i=1}^n f_j(h_i, t_i) = \frac{1}{n} \sum_{i=1}^n p(t|h_i) f_j(h_i, t)$$

- Otherwise we want maximum entropy, i.e. we do not want to introduce any additional order into the model (Occam's razor: simplest model is best)
- Entropy:

$$H(p) = \sum_{h,t} p(h, t) \log p(h, t)$$

Improved Iterative Scaling [Berger, 1993]

Input: Feature functions f_1, \dots, f_m , empirical distribution $\tilde{p}(x, y)$

Output: Optimal parameter values $\lambda_1, \dots, \lambda_m$

1. Start with $\lambda_i = 0$ for all $i \in \{1, 2, \dots, n\}$
2. Do for each $i \in \{1, 2, \dots, n\}$:
 - a. $\Delta\lambda_i = \frac{1}{C} \log \frac{\tilde{E}(f_i)}{E(f_i)}$
 - b. Update $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$
3. Go to step 2 if not all the λ_i have converged

Note: This algorithm requires that $\forall t, h : \sum_i f_i(t, h) = C$, which can be ensured with an additional filler feature