

Statistical Methods

GSLT Course, Week 2

Joakim Nivre

Outline

Part-of-Speech Tagging

Syntactic Parsing

Word Sense Disambiguation

Machine Translation

Evaluation

Outline

Part-of-Speech Tagging

Syntactic Parsing

Word Sense Disambiguation

Machine Translation

Evaluation

Part-of-Speech Tagging

- ▶ Given a word sequence $w_1 \cdots w_m$, determine the corresponding part-of-speech (tag) sequence $t_1 \cdots t_m$.
- ▶ Probabilistic view of the problem:

$$\arg \max_{t_1 \cdots t_m} P(t_1 \cdots t_m \mid w_1 \cdots w_m) =$$

$$\arg \max_{t_1 \cdots t_m} \frac{P(t_1 \cdots t_m)P(w_1 \cdots w_m \mid t_1 \cdots t_m)}{P(w_1 \cdots w_m)} =$$

$$\arg \max_{t_1 \cdots t_m} P(t_1 \cdots t_m)P(w_1 \cdots w_m \mid t_1 \cdots t_m)$$

Independence Assumptions

- ▶ Contextual model: Tags are dependent only on $n - 1$ preceding tags (tag n-grams, n-classes):

$$P(t_1 \cdots t_m) = \prod_{i=1}^m P(t_i | t_{i-(n-1)} \cdots t_{i-1})$$

- ▶ Lexical model: Word forms are dependent only on their own part-of-speech:

$$P(w_1 \cdots w_m | t_1 \cdots t_m) = \prod_{i=1}^m P(w_i | t_i)$$

Example

- ▶ Tagging *the can smells* using the triclass model:

$$P(dt \ nn \ vb \mid \textit{the can smells}) = P(dt \mid \# \#) \cdot P(nn \mid \# \ dt) \cdot P(vb \mid dt \ nn) \cdot P(\textit{the} \mid dt) \cdot P(\textit{can} \mid nn) \cdot P(\textit{smells} \mid vb)$$

- ▶ Compare:

$$\begin{aligned} P(\textit{can} \mid vb) &> P(\textit{can} \mid nn) \\ P(\textit{smells} \mid vb) &> P(\textit{smells} \mid nn) \\ P(nn \mid \# \ dt) &>> P(vb \mid \# \ dt) \\ P(vb \mid dt \ nn) &> P(nn \mid dt \ nn) (?) \end{aligned}$$

Hidden Markov Models 1

- ▶ Markov models are probabilistic finite automata that are used for many kinds of (sequential) disambiguation tasks such as:
 1. Speech recognition
 2. Spell checking
 3. Part-of-speech tagging
 4. Named entity recognition
- ▶ A (discrete) Markov model runs through a sequence of states emitting signals. If the state sequence cannot be determined from the sequence of emitted signals, the model is said to be hidden.

Hidden Markov Models 2

- ▶ A Markov model consists of five elements:
 1. A finite set of states $\Omega = \{s_1, \dots, s_k\}$.
 2. A finite signal alphabet $\Sigma = \{\sigma_1, \dots, \sigma_m\}$.
 3. Initial probabilities $P(s)$ (for every $s \in \Omega$) defining the probability of starting in state s .
 4. Transition probabilities $P(s_i | s_j)$ (for every $(s_i, s_j) \in \Omega^2$) defining the probability of going from state s_j to state s_i .
 5. Emission probabilities $P(\sigma | s)$ (for every $(\sigma, s) \in \Sigma \times \Omega$) defining the probability of emitting symbol σ in state s .

Hidden Markov Models 3

- ▶ State transitions are assumed to be independent of everything except the current state:

$$P(s_1 \cdots s_n) = \prod_{i=1}^n P(s_1) P(s_i | s_{i-1})$$

- ▶ Signal emissions are assumed to be independent of everything except the current state:

$$P(s_1 \cdots s_n, \sigma_1 \cdots \sigma_n) = \prod_{i=1}^n P(s_1) P(s_i | s_{i-1}) P(\sigma_i | s_i)$$

Hidden Markov Models 4

- ▶ The probability of a signal sequence is obtained by summing over all state sequences that could have produced that signal sequence:

$$P(\sigma_1 \cdots \sigma_n) = \sum_{s_1 \cdots s_n \in \Omega^n} \prod_{i=1}^n P(s_1) P(s_i | s_{i-1}) P(\sigma_i | s_i)$$

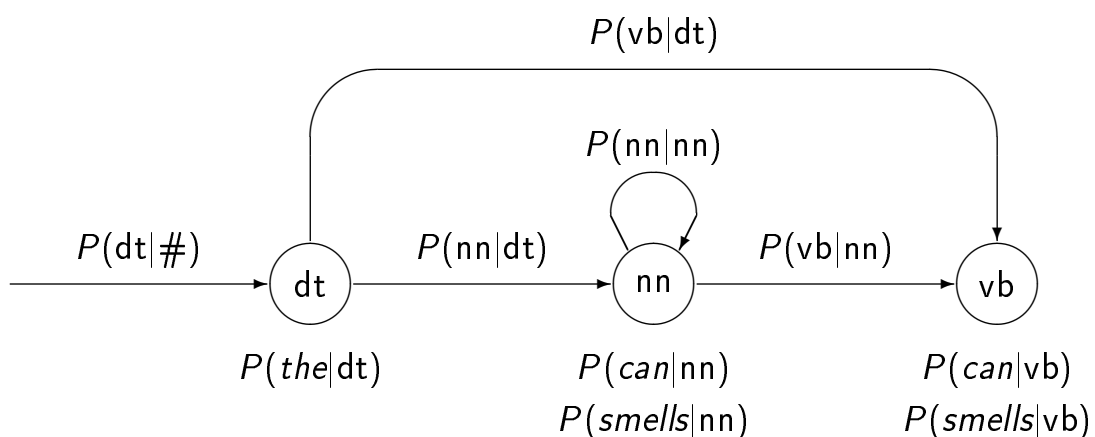
- ▶ Problems for HMMs:

1. Optimal state sequence: $\arg \max_{s_1 \cdots s_n} P(s_1 \cdots s_n | \sigma_1 \cdots \sigma_n)$.
2. Signal probability: $P(\sigma_1 \cdots \sigma_n)$.
3. Parameter estimation: $\hat{P}(s)$, $\hat{P}(s_i | s_j)$, $\hat{P}(\sigma | s)$.

HMM Tagging

- ▶ Contextual model:
 1. The biclass model can be represented by a first-order Markov model, where each state represents one tag.
 2. The triclass model can be represented by a second-order Markov model, where each state represents a pair of tags.
 3. In both cases, transition probabilities represent contextual probabilities.
- ▶ Lexical model:
 1. Signals represented word forms.
 2. Emission probabilities represent lexical probabilities.

Example: First-Order HMM



Parameter Estimation

- ▶ Two different methods for estimating probabilities (lexical and contextual):
 1. Supervised learning: Probabilities can be estimated using frequency counts in a (manually) tagged training corpus.
 2. Unsupervised learning: Probabilities can be estimated from an untagged training corpus using expectation-maximization.
- ▶ Experiments have shown that supervised learning yields superior results even with limited amounts of training data.

Supervised Learning

- ▶ Maximum likelihood estimation:
 1. Contextual probabilities:

$$\hat{P}(t_i | t_{i-2}t_{i-1}) = \frac{C(t_{i-2}t_{i-1}t_i)}{C(t_{i-2}t_{i-1})}$$

2. Lexical probabilities:

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

- ▶ Maximum likelihood estimates need to be smoothed because of sparse data (cf. language modeling).

Unsupervised Learning

- ▶ Expectation-Maximization (EM) is a method for approximating optimal probability estimates:
 1. Guess an estimate $\hat{\theta}$.
 2. Expectation: Compute expected frequencies based on training data and current value of $\hat{\theta}$.
 3. Maximization: Adjust $\hat{\theta}$ based on expected frequencies.
 4. Iterate steps 2 and 3 until convergence.
- ▶ Special case for HMM known as the Baum-Welch algorithm.
- ▶ Problem: Local maxima.

Example: Expectation-Maximization 1

- ▶ Lexicon:

<i>the</i>	dt
<i>car</i>	nn
<i>can</i>	nn vb

- ▶ Training corpus:

the can
the car

- ▶ Initial estimates:

$$\hat{P}(nn|dt) = \hat{P}(vb|dt) = 0.5$$

Example: Expectation-Maximization 2

Expectation

$$E[C(nn|dt)] = 1.5$$

$$E[C(vb|dt)] = 0.5$$

$$E[C(nn|dt)] = 1.75$$

$$E[C(vb|dt)] = 0.25$$

$$E[C(nn|dt)] = 1.875$$

$$E[C(vb|dt)] = 0.125$$

Maximization

$$\hat{P}(nn|dt) = 0.75$$

$$\hat{P}(vb|dt) = 0.25$$

$$\hat{P}(nn|dt) = 0.875$$

$$\hat{P}(vb|dt) = 0.125$$

$$\hat{P}(nn|dt) = 0.9375$$

$$\hat{P}(vb|dt) = 0.0625$$

Smoothing for Part-of-Speech Tagging

- ▶ Contextual probabilities are structurally similar to n-gram probabilities in language modeling and can be smoothed using the same methods.
- ▶ Smoothing of lexical probabilities breaks down into two sub-problems:
 1. Known words are usually handled with standard methods.
 2. Unknown words are often treated separately to take into account information about suffixes, capitalization, etc.

Viterbi Tagging

- ▶ HMM tagging amounts to finding the optimal path (state sequence) through the model for a given signal sequence.
- ▶ The number of possible paths grows exponentially with the length of the input.
- ▶ The Viterbi algorithm is a dynamic programming algorithm that finds the optimal state sequence in polynomial time.
- ▶ Running time is $O(ms^2)$, where m is the length of the input and s is the number of states in the model.

Class-Based Language Models

- ▶ The n -class model for part-of-speech tagging can also be used as a language model:

$$P(w_1 \cdots w_m) = \sum_{t_1 \cdots t_m} P(t_1 \cdots t_m, w_1 \cdots w_m)$$

- ▶ Efficient algorithms for finding string probabilities:
 1. Forward algorithm
 2. Backward algorithm
 3. Forward-backward algorithm

Mini-Project

- ▶ Implement HMM tagger:
 1. Viterbi tagging
 2. Parameter estimation (supervised)
- ▶ Available data (Swedish):
 1. Training corpus (100 000 words)
 2. Test corpus (10 000 words)
- ▶ Project requires good programming skills.

Outline

Part-of-Speech Tagging

Syntactic Parsing

Word Sense Disambiguation

Machine Translation

Evaluation

Syntactic Parsing

- ▶ Given a word sequence $w_1 \cdots w_m$, determine the corresponding syntactic analysis A .
- ▶ Probabilistic view of the problem:

$$\arg \max_A P(A \mid w_1 \cdots w_m) =$$

$$\arg \max_A \frac{P(A)P(w_1 \cdots w_m \mid A)}{P(w_1 \cdots w_m)} =$$

$$\arg \max_A P(A)P(w_1 \cdots w_m \mid A) =$$

- ▶ We will assume that A is a context-free parse tree, but the same reasoning applies to any choice of syntactic analysis.

Context-Free Parse Trees

- ▶ Since a context-free parse tree T for the string $w_1 \cdots w_m$ includes the string itself, it follows that:

$$P(w_1 \cdots w_m \mid T) = \begin{cases} 1 & \text{if } \text{yield}(T) = w_1 \cdots w_m \\ 0 & \text{otherwise} \end{cases}$$

- ▶ Hence, if we restrict our attentions to parse trees with the right terminals, we simply have to search for the tree T with the highest probability:

$$\arg \max_T P(T)$$

Probabilistic Context-Free Grammar (PCFG)

- ▶ A PCFG G is a quintuple (T, N, R, S, P) , where:
 1. T is a terminal alphabet
 2. N is a nonterminal alphabet
 3. R is a set of productions $A \rightarrow \alpha_1 \cdots \alpha_n$
($A \in N, \alpha_1, \dots, \alpha_n \in N \cup T$)
 4. $S \in N$ is a start symbol
 5. P is a function assigning a probability to each production in R in such a way that:

$$\forall A \in N : \sum_{\alpha_1 \cdots \alpha_n : A \rightarrow \alpha_1 \cdots \alpha_n \in R} P(A \rightarrow \alpha_1 \cdots \alpha_n) = 1$$

Example PCFG

- ▶ Productions (with probabilities):

$S \rightarrow NP VP$	1.0	$V \rightarrow saw$	1.0
$PP \rightarrow P NP$	1.0	$NP \rightarrow astronomers$	0.1
$VP \rightarrow V NP$	0.7	$NP \rightarrow ears$	0.18
$VP \rightarrow VP PP$	0.3	$NP \rightarrow saw$	0.04
$NP \rightarrow NP PP$	0.4	$NP \rightarrow stars$	0.18
$P \rightarrow with$	1.0	$NP \rightarrow telescopes$	0.1

- ▶ Ambiguous string: *astronomers saw stars with ears.*

Independence Assumptions

- ▶ The probability of a rule $A \rightarrow \alpha_1 \cdots \alpha_n$ represents the probability of using the rule to expand a node labeled A :

$$P(A \rightarrow \alpha_1 \cdots \alpha_n) = P\left(\begin{array}{c} A \\ \swarrow \quad \searrow \\ \alpha_1 \cdots \alpha_n \end{array} \middle| A\right)$$

- ▶ The probability of using the rule in a derivation $S \Rightarrow^* w_1 \cdots w_m$ is assumed to be independent of anything before or after that step:

$$\frac{P(A \Rightarrow \alpha_1 \cdots \alpha_n \mid S \Rightarrow^* \beta A \gamma, \beta \alpha_1 \cdots \alpha_n \gamma \Rightarrow^* w_1 \cdots w_m)}{P(A \rightarrow \alpha_1 \cdots \alpha_n)} =$$

Probabilities for Parse Trees and Strings

- ▶ The probability of a parse tree is the product of the probabilities of all its independent subtrees:

$$P(T) = \prod_{t(A, \alpha_1 \cdots \alpha_n) \in T} P(A \rightarrow \alpha_1 \cdots \alpha_n)$$

where $t(A, \alpha_1 \cdots \alpha_n) \in T$ signifies that T contains a local tree with root labeled A and children labeled $\alpha_1, \dots, \alpha_n$.

- ▶ The probability of a string is the sum of the probabilities of all its parse trees:

$$P(w_1 \cdots w_m) = \sum_{T: \text{yield}(T) = w_1 \cdots w_m} P(T)$$

Example

- ▶ Parse probabilities for *astronomers saw stars with ears*:

1. PP-in-NP:

$$0.000268 \cdot P(\text{NP} \rightarrow \text{NP PP}) = 0.00268 \cdot 0.4 = 0.0009072$$

2. PP-in-VP:

$$0.000268 \cdot P(\text{VP} \rightarrow \text{VP PP}) = 0.00268 \cdot 0.3 = 0.0006804$$

- ▶ String probability:

$$P(\text{astronomers saw stars with ears}) = 0.0009072 + 0.0006804 = 0.0015876$$

Parameter Estimation

- ▶ Parameter estimation for PCFGs can be divided into two parts:

1. Nonterminal productions:

$$P(A \rightarrow \alpha_1 \cdots \alpha_n)$$

2. Terminal productions:

$$P(t \rightarrow w) = P(w|t)$$

- ▶ In practice, parsing is often preceded by a tagging phase, which eliminates the need for terminal productions.

Supervised Learning

- ▶ Maximum likelihood estimation:

1. Nonterminal productions:

$$\hat{P}(A \rightarrow \alpha_1 \cdots \alpha_n) = \frac{C(t(A, \alpha_1 \cdots \alpha_n))}{C(t(A))}$$

2. Terminal productions:

$$\hat{P}(t \rightarrow w) = \frac{C(w, t)}{C(t)}$$

- ▶ Supervised learning presupposes a parsed corpus (treebank).
- ▶ Smoothing is normally less critical for nonterminal productions.

Unsupervised Learning

- ▶ Expectation-Maximization (EM) can be used for unsupervised learning from unanalyzed data.
- ▶ Special case for PCFG known as the Inside-Outside algorithm.
- ▶ The problem of local maxima is worse than for HMMs and makes the method practically unusable.

PCFG Parsing

- ▶ Parsing algorithms for PCFGs are based on standard algorithms for ordinary CFGs such as CKY and Earley's algorithm.
- ▶ Conceptually, parsing can be seen as consisting of two phases:
 1. Computing all possible parses and storing them in a compact representation (packed parse forest).
 2. Extracting the most probable parse from the parse forest.
- ▶ For PCFGs these two phases can be efficiently computed in parallel using dynamic programming.

Limitations of the PCFG Model

- ▶ The PCFG model is a fairly blunt instrument for disambiguation because:
 1. The model does not take into account the structural context in which a phrase is expanded.
 2. The model does not exploit lexical relations, in particular relations between the lexical heads of phrases.
- ▶ Recent developments in probabilistic grammars and parsing are meant to overcome these shortcomings.

Markovization

- ▶ In a Markovized grammar context-free productions are replaced by stochastic processes that generate children one by one, conditioning on previous children, parents, grandparents, etc.
- ▶ Compare:
 1. Standard PCFG:

$$P(A \rightarrow \alpha_1 \cdots \alpha_n) = P(\alpha_1 \cdots \alpha_n | A)$$

2. Markov grammar:

$$P(A \rightarrow \alpha_1 \cdots \alpha_n) = \prod_{i=1}^n P(\alpha_i | A, p(A), \alpha_{i-1}, \dots)$$

where $p(A)$ is the parent of A (grandparent of α_j).

Lexicalization

- ▶ In a lexicalized grammar, nonterminals carry information about lexical heads.
- ▶ Compare:
 1. Standard PCFG:

$$P(A \rightarrow \alpha_1 \cdots \alpha_n) = P(\alpha_1 \cdots \alpha_n | A)$$

2. Lexicalized grammar:

$$P(A(h_h) \rightarrow \alpha_1 \cdots \alpha_n) = P(\alpha_1(h_1) \cdots \alpha_h(h) \cdots \alpha_n(h_n) | A(h))$$

where h_j is the lexical head of α_j and α_h is the head child of A .

Generative and Discriminative Models

- ▶ A PCFG is a *generative* model, because it models the *joint* distribution $P(T, w_{1,n})$ of input and output.
- ▶ A *discriminative* model instead models the *conditional* distribution $P(T|w_{1,n})$ of output given input.
- ▶ Comparison:
 - ▶ Generative models are more informative, because the conditional distribution can be derived from the joint distribution (but not vice versa).
 - ▶ Discriminative models are more flexible, because they allow arbitrary features without independence assumptions.

Log-Linear Models

- ▶ Let $Gen(w_{1,n}) = \{T_1, \dots, T_m\}$ be a function that generates candidate parse trees for a given input sentence $x = w_1 \cdots w_n$.
- ▶ Conditional log-linear parsing model (a.k.a. MaxEnt):

$$P(T|w_{1,n}) = \frac{\exp\left(\sum_{i=1}^k \lambda_i f_i(w_{1,n}, T)\right)}{\sum_{T' \in Gen(w_{1,n})} \exp\left(\sum_{i=1}^k \lambda_i f_i(w_{1,n}, T')\right)}$$

where

- ▶ each f_i is a feature of the input and output,
- ▶ each λ_i is the weight of feature f_i .

Mini-Project

- ▶ PCFG for artificial blocks world English:

put the red cone on the square

put the block on the blue square on the circle

- ▶ Simple parser implementation (Prolog tutorial available)
- ▶ Parameter estimation from treebank (supervised learning)
- ▶ Model evaluated as parser and/or as language model

Outline

Part-of-Speech Tagging

Syntactic Parsing

Word Sense Disambiguation

Machine Translation

Evaluation

Word Sense Disambiguation

- ▶ Given a word w with possible senses s_1, \dots, s_m occurring in context C , determine the correct sense s_j .
- ▶ Probabilistic view of the problem:

$$\begin{aligned} \arg \max_{s_j} P(s_j | C) &= \\ \arg \max_{s_j} \frac{P(s_j)P(C | s_j)}{P(C)} &= \\ \arg \max_{s_j} P(s_j)P(C | s_j) &= \end{aligned}$$

- ▶ We will assume that C is represented as a sequence of words $w_{-n} \cdots w_{-1} [w] w_1 \cdots w_n$ surrounding w .

Bayesian Classifiers

- ▶ A Bayesian classifier assigns to each new instance w the most probable classification s_j given the context C and the previously seen training data D .
- ▶ The Bayes optimal classifier combines the prediction of all possible models M_j :

$$\arg \max_{s_j} \sum_j P(s_j | C, M_j)P(M_j | D)$$

- ▶ The Bayes optimal classification is optimal on average. However, it is usually very costly to apply, since it requires computing all possible models.

Naive Bayes

- ▶ The Naive Bayes classifier is based on two assumptions:
 1. Only the best model given the training data is considered:

$$\arg \max_{s_i} P(s_i | C, M_{best})$$

2. Attributes of the instance (context) are assumed to be independent of each other:

$$\arg \max_{s_i} P(s_i)P(C | s_i) = \arg \max_{s_i} P(s_i) \prod_{w_j \in C} P(w_j | s_i)$$

Parameter estimation

- ▶ In order to use the naive Bayes classifier for word sense disambiguation, we need to train two models:
 1. Sense model: $P(s_i)$
 2. Context model: $P(w_j | s_i)$
- ▶ Both models can be trained using MLE and smoothing using a sense-tagged corpus. Alternative data sources:
 1. Dictionaries
 2. Thesauri
 3. Parallel corpora

Parallel Corpora

- ▶ Lexical ambiguities are sometimes resolved in translation:

English	French	Swedish
duty	droit	skatt
	devoir	plikt
sentence	peine	dom
	phrase	mening
interest	intérêt	intresse
		ränta

- ▶ A word-aligned parallel corpus can be viewed as a partially sense-tagged corpus (Gale et al. 1992).

Thesaurus Classes

- ▶ The different senses of an ambiguous word often belongs to different semantic categories:

Word	Roget category
star	Universe
	Entertainer
	Insignia

- ▶ By grouping together all words belonging to a given category c , we can estimate $P(c)$ and $P(w_j | c)$ and use these as estimates for $P(s_i)$ and $P(w_j | s_j)$ if $s_i \in c$ (Yarowsky 1992).

Unsupervised Methods

- ▶ Unsupervised learning of a naive Bayes classifier is possible using Expectation-Maximization (Schütze 1998):
 1. Random initialization of $P(s_i)$ and $P(w_j | s_i)$.
 2. E-step: Compute expected frequencies based on current estimates and training corpus.
 3. M-step: Revise estimates using MLE with respect to expected frequencies.
 4. Repeat step 1 and 2 until convergence.
- ▶ Problem: How many senses per word?

Variations on the Bayesian Approach

- ▶ The context model in the Bayes classifier can be varied in different ways:
 1. Window size
 2. Weighting in proportion to distance
 3. Stop word filtering

One Sense per Collocation

- ▶ The naive Bayes classifier assumes independence between contextual features and uses an (unweighted) combination of their evidence for disambiguation.
- ▶ Yarowsky (1995) instead proposes:
 1. Each feature f should be taken as an indicator of one sense only ($\arg \max_{s_i} P(s_i|f)$).
 2. Features can be ranked according to how strong indicators they are (e.g. using the ratio $\frac{P(s_i|f)}{P(\neg s_i|f)}$).
 3. When classifying a new instance, we should only consider the evidence of the strongest indicator.

One Sense per Discourse

- ▶ The sense of a given word w is normally very consistent within any given document.
- ▶ This can be used as a powerful heuristic in word sense disambiguation by only allowing each word to have one sense within a given document.
- ▶ Yarowsky (1995) combines “one sense per collocation” and “one sense per discourse” to achieve very good results with (almost) unsupervised learning.

Yarowsky's Algorithm

- ▶ For each sense s_i of w , let F_i be the set of contextual features (collocations) indicative of s_i according to some seed text.
- ▶ Repeat until convergence:
 1. For each s_i , let C_i be all the occurrences (contexts) of w that are assigned sense s_i given the current collocation sets (and their ranking).
 2. Optionally: Apply “one sense per discourse” to reclassify minority senses of w in a given document.
 3. For each s_i , let F_i be the set of collocations for s_i whose strength (based on the currently disambiguated tokens) exceeds a certain threshold.

Word Sense Discrimination

- ▶ Traditional word sense disambiguation can be described as sense tagging, since it requires classification using a predefined set of labels (sense tags).
- ▶ In word sense discrimination, the goal is instead to discriminate senses based on the similarity of contexts, without having a predefined set of senses to choose from.
- ▶ Word sense discrimination typically uses unsupervised learning methods to group tokens/contexts into clusters based on some similarity metric.

Mini-Project

- ▶ Word sense disambiguation for Swedish:
 - resa 1 to raise
 - 2 to travel
 - 3 a journey
- ▶ Probabilistic model of your choice (e.g., naive Bayes)
- ▶ Parameter estimation from corpus (supervised)
- ▶ Evaluation with respect to test corpus

Outline

Part-of-Speech Tagging

Syntactic Parsing

Word Sense Disambiguation

Machine Translation

Evaluation

Statistical Methods in Translation

- ▶ Statistical methods are used in two areas related to translation:
 1. Text alignment: Given a parallel (translation) corpus, determine which paragraphs, sentences and (possibly) words correspond to each other in the source and target texts.
 2. Machine translation: Use statistical methods to map texts (sentences) in a source language to their translations in a target language.
- ▶ In addition, statistical methods can be used in any submodule (tagger, parser, word sense disambiguator, etc.) of an ordinary machine translation system.

Sentence Alignment

- ▶ We want to partition a source text $S_{1,m}$ and its translation $T_{1,n}$, into paired subsequences of sentences (or beads) such that:
 1. For each bead, the subsequence $T_{i,j}$ of $T_{1,n}$ is a translation of the subsequence $S_{k,l}$ of $S_{1,m}$.
 2. For each bead, it is impossible to break it up into smaller beads for which the translation correspondence still holds.
- ▶ Example:

$S_{1,3}$: John came. Mary came. Bill left.

$T_{1,2}$: John och Mary kom. Bill gick.

Alignment: $(S_{1,2}, T_{1,1}), (S_{3,3}, T_{3,3})$

Statistical Methods for Sentence Alignment

- ▶ Probabilistic formulation of the problem:

$$\arg \max_A P(A|S, T) = \arg \max_A P(A, S, T)$$

- ▶ Independence assumption: For an alignment consisting of beads B_1, \dots, B_k , the probabilities of different beads are independent of each other:

$$\arg \max_A P(A, S, T) = \prod_{i=1}^k P(B_k)$$

Length-Based Methods

- ▶ The standard model of bead probabilities is based on two factors:
 1. The probability of the alignment type (1:1, 1:2, 2:1, etc.).
 2. The probability of the difference in length between the aligned sequences (given the alignment type).

- ▶ Model:

$$P(B = (S_{1,m}, T_{1,n})) = P(m : n) \cdot P(\text{len}(T_{1,n}) - \text{len}(S_{1,m}) | m : n)$$

- ▶ Dynamic programming is used to compute the most probable alignment efficiently.

Word Alignment

- ▶ Word alignment normally presupposes that the texts are aligned on the sentence level and combines different information sources:
 1. Cooccurrence statistics: Words that often occur in aligned sentences are candidates for alignment.
 2. Relative positions: Words that occur in (roughly) the same position in aligned sentences are candidates for alignment.
 3. Alignment types: 1:1 alignments are more probable than 2:1 and 1:2 alignments (in most cases).
- ▶ In practice, word alignment is much more difficult than sentence alignment.

Statistical Machine Translation

- ▶ Given a source language text S , determine the most probable translation T in the target language:

$$\begin{aligned} \arg \max_T P(T | S) &= \arg \max_T \frac{P(T)P(S | T)}{P(S)} \\ &= \arg \max_T P(T)P(S | T) \end{aligned}$$

- ▶ Two models:
 1. Language model: $P(T)$
 2. Translation model: $P(S|T)$

The IBM Model

- ▶ Language model: Trigram model
- ▶ Translation model based on word alignment:

$$P(S|T) = \sum_A P(A, S|T)$$

$$P(A, S|T) = \prod_{(w_S, w_T) \in A} P(w_S|w_T) \cdot P(p_{w_S}|p_{w_T}, l_S) \prod_{w_T \in T} P(N_{w_T}|w_T)$$

where p_w is the position of w in the sentence and N_w is the number of words aligned to w in the other sentence.

Example

- ▶ Aligned sentence pair:

S = he saw the car

T = han såg bilen

A = (han, he) (såg, saw) (bilen, the) (bilen, car)

- ▶ Factors:

1. Translation: $P(\text{he}|\text{han}) \cdot P(\text{saw}|\text{såg}) \cdot P(\text{the}|\text{bilen}) \cdot P(\text{car}|\text{bilen})$
2. Distortion: $P(1|1, 4) \cdot P(2|2, 4) \cdot P(3|3, 4) \cdot P(3|4, 4)$
3. Fertility: $P(1|\text{han}) \cdot P(1|\text{såg}) \cdot P(2|\text{bilen})$

Decoding

- ▶ Finding the most probable translation is in general an intractable computational problem because:
 1. We need to maximize over all possible target sentences.
 2. We need to sum over all possible alignments for each target sentence.
- ▶ Even assuming that we know the words of the target sentence, the number of possible permutations and alignments (both) grow superexponentially with sentence length.
- ▶ The standard implementation uses stack search, pruning less likely hypotheses at an early stage.

Phrase-Based Statistical Translation

- ▶ Phrase-based translation models are a generalization of the word-based models, where translation, distortion and fertility models are defined on phrases instead of words (see, e.g., Koehn, Och and Marcu 2003).
- ▶ Phrases need not be linguistically motivated but can be arbitrary n-grams in the two languages.
- ▶ A variety of methods have been proposed for extracting phrases:
 1. Phrases from word alignments
 2. Phrases from phrase alignments
 3. Syntactic phrases (filtered by parsing)

Syntax-Based Statistical Translation

- ▶ Syntax-based translation models estimate translation probabilities based on syntactic parses of the source and/or target language sentences.
- ▶ For example, Yamada and Knight (2001) proposes a model where parse trees in the two languages are related by stochastic operations on tree nodes capturing linguistic differences such as word order and case marking.
- ▶ In addition, the n-gram language model can be replaced by a statistical parsing model, so that candidate translations can be ranked according to the probabilities of their parse trees, as proposed by Charniak, Knight and Yamada (2003).

Mini-Project

- ▶ Translation of blocks world commands (English-Swedish or Swedish-English):

put the red cone on the square
ställ den röda konen på kvadraten

- ▶ Tasks:
 1. Construct a language model for computing $P(T)$.
 2. Construct a translation model for computing $P(S|T)$.
 3. Construct an algorithm that finds $\arg \max_T P(T|S)$.

Outline

Part-of-Speech Tagging

Syntactic Parsing

Word Sense Disambiguation

Machine Translation

Evaluation

Evaluation

- ▶ Basic questions in this lecture:
 1. Why is statistical evaluation useful in language technology?
 2. What kind of statistical methods are relevant?
- ▶ Focus: Empirical evaluation of accuracy.

Aspects of Evaluation

- ▶ Software product evaluation (ISO 9126):
 1. Functionality
 2. Reliability
 3. Usability
 4. Efficiency
 5. Maintainability
 6. Portability
- ▶ These quality characteristics apply to natural language processing systems as well as other software systems.

Statistical Evaluation

- ▶ Many aspects of natural language processing systems can be evaluated by performing series of (more or less controlled) experiments.
- ▶ The results of such experiments are often quantitative measurements which can be summarized and analyzed using statistical methods.
- ▶ Evaluation methods are (in principle) independent of whether the systems evaluated are statistical or not.

Empirical Evaluation of Accuracy

- ▶ Most natural language processing systems make errors even when they function perfectly.
- ▶ Accuracy can be tested by running systems on representative samples of inputs.
- ▶ Three kinds of statistical methods are relevant:
 1. Descriptive statistics: Measures
 2. Estimation: Confidence intervals
 3. Hypothesis testing: Significant differences

Test Data

- ▶ Requirements on test data set:
 1. Distinct from any training data
 2. Unbiased (random sample)
 3. As large as possible
- ▶ These requirements are not always met.
- ▶ Testing may be supervised or unsupervised depending on whether the test data set contains solutions or not.
- ▶ Gold standard: Solutions provided by human experts.

Descriptive Statistics

- ▶ Descriptive measures such as sample means and proportions are used to summarize test results.
- ▶ Examples:

1. Accuracy rate (percent correct): $\frac{1}{n} \sum_{i=1}^n x_i$

2. Recall: $\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$

3. Precision: $\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$

4. Logprob: $\frac{1}{n} - \sum_{i=1}^n \log_2 \hat{P}(x_i)$

Confidence Intervals

- ▶ Descriptive sample measures can be used to estimate the accuracy of systems when applied to new data.
- ▶ It is good scientific practice to provide confidence intervals for such measures.
- ▶ Example: 95% confidence interval for accuracy rate 0.92 with sample size 1000:

$$0.92 \pm 1.96 \sqrt{\frac{0.92 \cdot 0.08}{1000}}$$

- ▶ Percent correct: 92% \pm 1.7%

Hypothesis Testing

- ▶ Given descriptive sample measures for several systems tested on the same data, we can use hypothesis tests to assess the significance of differences.
- ▶ Example: Paired t-test to compare accuracy rates 0.92 and 0.94 with sample size 1000, where the variance of the difference is 0.13:

$$t = \frac{0.02}{\sqrt{\frac{0.13}{1000}}} = 1.75$$

- ▶ Result: Difference not significant at 0.05 level.

Example 1: Language Modeling

- ▶ Evaluation of language models as such are always unsupervised (no gold standards for string probabilities).
- ▶ Evaluation measures:
 1. Corpus probability
 2. Corpus entropy (logprob)
 3. Corpus perplexity

Example 2: PoS Tagging and WSD

- ▶ Supervised evaluation with gold standard is the norm.
- ▶ Evaluation measures:
 1. Percent correct
 2. Recall
 3. Precision

Example 3: Syntactic Parsing

- ▶ Supervised evaluation with gold standard (treebank).
- ▶ Evaluation measures:
 1. Percent correct
 2. Labeled/bracketed recall
 3. Labeled/bracketed precision
 4. Zero crossing

Summary – Evaluation

- ▶ Why?
 1. Experimental evaluation (often) yields quantitative results.
 2. These results can be analyzed using statistical methods.
- ▶ What?
 1. Descriptive statistics to provide measures
 2. Statistical estimation to derive confidence intervals
 3. Hypothesis testing to assess differences