Tagging

- used for disambiguation between more suitable meanings of one word
- between morphology and syntax

Tagsets

- tag ~ one category

  $T \longleftrightarrow (C_1 \ldots C_n)$ 1:1 mapping

- Penn treebank
- Brown Corpus

Lemmatization ~ reduced morphological analysis

- lemma ≈ one dictionary entry ref. / lexical unit

  $L: A^+ \longrightarrow 2^L$, even though $A^+ - L$ is what is wanted → MA disambiguation

Morphological analysis

- Word form list – books: book – 2/VBZ, book – 1/NNS – it is only there/not there
- direct coding – from endings etc. $A^+ \longrightarrow 2^{(L,e)}$
- FSM ~ finite state machinery ...
- CFG ... – more linguistic than computational phenomena

FST

- implemented as FSA, one symbol: $(r:s)$ from alphabets R, S
- can just run for acceptance or analysis or synthesis

  analysis: input: s, output: r

  synthesis: input: r, output: s

Rule-based disambig.

- rules using
  - word forms
  - tags
  - combination...
  - if-then / reg. exp.
- the biggest problem is that the rules are created by hand
- trying to eliminate as much possibilities as possible to have just one (correct) tag.
- today relevant for checking simple rules with NO computational costs

HMM - tagging:

- tags can be products of „hidden" features of the language
- these features are represented by hidden states in HMM
- separation of transition and emission distributions

Transformation - tagging:

- NOT source channel view
- NOT probabilistic
- BUT statistical
  - uses training data to learn rules
- criterion-based selection of rules selection
- rules can look to the future (diff from HMM)

- Maximum Entropy principle...
  - when constraints given, the higher the entropy, the less closed doors in uncertainty.

Parsing

Phrase structure tree

- always rooted, can be enclosed by correct bracketing
- projective

Dependency tree

- one word, one node
- not always projective

PCFG

- Sometimes more parses are correct, but we still need to output just one

$P(T) = \prod_i p(r_i)$, r_i are rules used to generate the sentence.

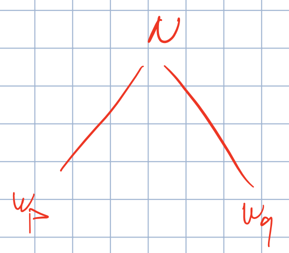
↳ very strong independence assumptions...

$\sum_{r \in R_A} p(r) = 1$ ⟶ aka. given left hand, all right hands must sum up to 1.

estimating the probs: $p(r) = c(r) / c(A)$ for $r: A \to \alpha$

$P(W) = \sum_j P(T_j)$ ~ T_j generates word W

$$\underbrace{\beta_N(p,q) = \sum_{A,B} \sum_{d=p}^{q-1} \beta_A(p,d) P(N \to A B) \beta_B(d+1, q)}_{\text{inside probability}}$$



	dog 1	saw 2	cat 3	with 4	telescope 5
1	N 0,3 NP 0,21		S 0,0441		S 0,00966
2		V 1,0	VP 0,21		VP 0,018 + 0,028 = 0,046
3			N 0,5 NP 0,35		NP 0,03
4				PREP 1,0	PP 0,2
5					N 0,2 NP 0,14

this is the probability of the final parse.

What is a Probabilistic CFG? Use five sentences at most.

A **Probabilistic Context-Free Grammar (PCFG)** is a context-free grammar where each production rule is assigned a probability. These probabilities represent how likely a rule is to be used when generating a sentence. The sum of the probabilities of all rules with the same left-hand nonterminal must equal 1. PCFGs are commonly used in natural language processing to model the likelihood of different parse trees. They allow the parser to choose the most probable interpretation of a sentence when multiple parses are possible.

Statistical machine translation

What is the source channel view of Statistical Machine Translation? Write the formula and explain it. Use five sentences at most.

The **source channel view** of Statistical Machine Translation (SMT) treats translation as a **noisy channel problem**, where we assume the target sentence e (e.g., English) generates the source sentence f (e.g., French). The goal is to find the most probable translation e given a source sentence f, formulated as:

$$\hat{e} = \arg\max_e P(e \mid f) = \arg\max_e P(f \mid e)P(e)$$

Here, $P(e)$ is the **language model** ensuring fluency of the target sentence, and $P(f \mid e)$ is the **translation model** assessing how well e explains f. This view separates translation into modeling translation probabilities and target language fluency. Decoding then finds the target sentence e that maximizes this product.

Explain the concept of alignment in Statistical Machine Translation. Write the formula for a translation model using the alignment. How is it simplified in the classical IBM models? Use five sentences at most.

In Statistical Machine Translation, **alignment** refers to the mapping between words in the source sentence and their corresponding words in the target sentence. The **translation model** with alignment is expressed as:

$$P(f \mid e) = \sum_a P(f, a \mid e)$$

where a represents a specific alignment between the words of the source sentence f and the target sentence e. In classical IBM models (e.g., IBM Model 1), this is simplified by assuming **independence between word alignments** and positions, making $P(f, a \mid e)$ a product of individual word translation probabilities $t(f_j \mid e_{a_j})$. This simplification allows efficient training via the Expectation-Maximization (EM) algorithm.

Entropy:

$$H(P) = -\sum_x P(x) \cdot \log P(x) \qquad \text{NLL}$$

Perplexity $= 2^{H(P)}$

$$H(P_{\text{bigram}}) = -\sum_x P(x,y) \cdot \log P(x \mid y)$$

Cross Entropy:

$$H(P,Q) = -\sum_x P(x) \cdot \log Q(x)$$

$$H(P,X) \geq H(P) \quad \forall X$$

$$H(P,P) = H(P)$$

Maximum entropy methods

The form of the constraints for the Maximum Entropy model is defined as

Vyberte jednu nebo více možností:

a. $1/|T| \sum_{t=1..T} \sum_{y \in Y} p(y|x_t) f_i(y,x_t) - d_i = 0$, where T is the training data (and |T| its size), p(y|x) the conditional model probability distribution, y the predicted variable and x the context (x_t is the concrete context at t-th data item), and f_i the i-th feature. d_i is the true feature count as extracted from the training data.

Yes, this is the approximation formula using the data-oriented expected value computation due to the complexity of summing over all possible xs (which is often impossible to enumerate).

b. $\sum_{y,x} p(y|x) f_i(y,x) - d_i = 0$, where p is the conditional model distribution, f_i are the features, and d_i is the true count as extracted from the training data.

No. The weight in the expected value formula computation must always be the joint distribution, not the conditional one.

c. $E_p(f_i(y,x)) - d_i = 0$, where E is the expected value of feature count expressed in terms of the probability distribution p as $E_p(f_i) = \Sigma_{y,x} p(x,y) f_i(y,x)$, with p being the model joint distribution, and d_i is the true count as extracted from the training data.

Yes. It simply says that the (joint) distribution p must be such that it models the feature count in such a way that it equals to the true count as found in the data, by using the standart optimization technique known as Lagrange multipliers (where the "multipliers" then serve as the feature weights).

?

The model (distribution) that fulfills the constraints while maximizing the entropy has the following form:

Vyberte jednu nebo více možností:

a. $p(y|x) = 1/Z(x)\, e^{\sum_{i=1..N} \lambda_i f_i(y,x)}$, where y is the predicted variable, x is the context of y used (together with y) for computing the feature values, N is the number of features used in the model, $f_i(y,x)$ are the feature functions, and λ_i are their weights. Z(x) is the normalization factor for a given context x computed either from the special f_0 feature or simply by normalizing to <0..1>.

b. $E_p(f_i) = \sum_{y,x} p(x,y) f_i(y,x)$, where E is the expected value for each f_i, and p(x,y) is the joint distribution.

No, this is the expected count of the features used in the constraints where this value (expressed by means of the distribution) must equal to the empirical count from the training data (the d_i constant).

c. $p(y,x) = (1/Z)\, e^{\sum_{i=1..N} \lambda_i f_i(y,x)}$, where y is the predicted variable, x is the context of y used (together with y) for computing the feature values, N is the number of features used in the model, $f_i(y,x)$ are the feature functions, and λ_i are their weights. Z is the normalization factor, computed either from the special f_0 feature or simply by normalizing to <0..1>.

Yes. This is the joint probability "version". The normalization is needed for the model to be a probability distribution (and in fact that's what makes the computation, especially at training time, slow in terms of complexity based on N).

?