Part of Speech Tagging

Probabilistic Models

Task description – more formal

- Given a sentence and its tokens t_i, assign a single label $l \in L$ with L being the tagset (e.g. Penn Treebank, STTS) to every t_i
- This is a **structural problem**, where our input is a sequence ("a list of tokens") and the output is a sequence ("a list of labels"), and:
  - Both sequences have the same length
    - (This is not the case for OCR or speech recognition)

WORD	tag
the	**DET**
koala	**N**
put	**V**
the	**DET**
keys	**N**
on	**P**
the	**DET**
table	**N**

Issues of our current model

- We modelled our sequence problem as:

$$\underset{t_1^n}{\operatorname{argmax}} \sum_i s_N(t_i, i) + s_E(t_i, t_{i-1})$$

- The Viterbi algorithm and the transducer allow us to find the best sequence by scoring sequences based on the sum of the scores of…
  - … the observations of the current word (Node-scores)
  - … the transitions (Edge-scores)

Issues of our current model

$$\underset{t_1^n}{\operatorname{argmax}} \sum_i s_N(t_i, i) + s_E(t_i, t_{i-1})$$

- Our edge score show the Markov-property, where a tag t_i is only dependent on the previous tag t_{i-1}, however this is only a crude approximation

➔Can we empower our model by relaxing the Markov-property, e.g. by adding a dependency to the tag t_{i-2}?

➔ A HMM with this property is called **Order-2 Hidden Markov Modell**

HMM of order 2

- Adapting the optimization problem is easy:

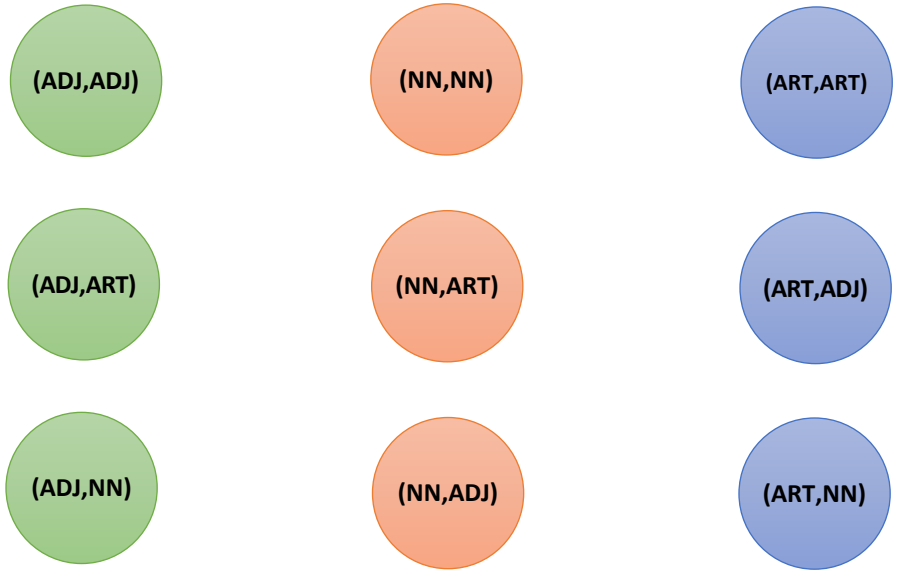
$$\underset{t_1^n}{\operatorname{argmax}} \sum_i s_N(t_i, i) + s_E(t_i, t_{i-1}, t_{i-2})$$

- And since s_E is just the log of a probability, we can also read the new edge scores from a labelled corpus:

$$s_E(t_i, t_{i-1}, t_{i-2}) = \ln \frac{C(t_i, t_{i-1}, t_{i-2})}{C(t_{i-1}, t_{i-2})}$$

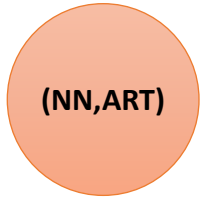
HMM of order 2

- This does however also alter our Transducer:
  - It should now also model transitions to t_i , coming from (t_{i-2}, t_{i-1}), so our states in the transducer will now be tuples!

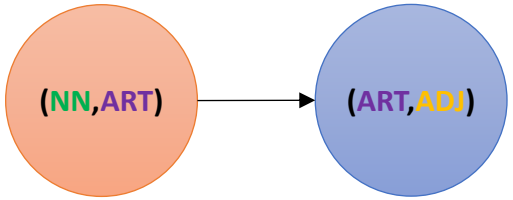


HMM of order 2

- This does however also alter our Transducer:
  - It should now also model transitions to t_i , coming from (t_{i-2}, t_{i-1}), so our states in the transducer will now be tuples!



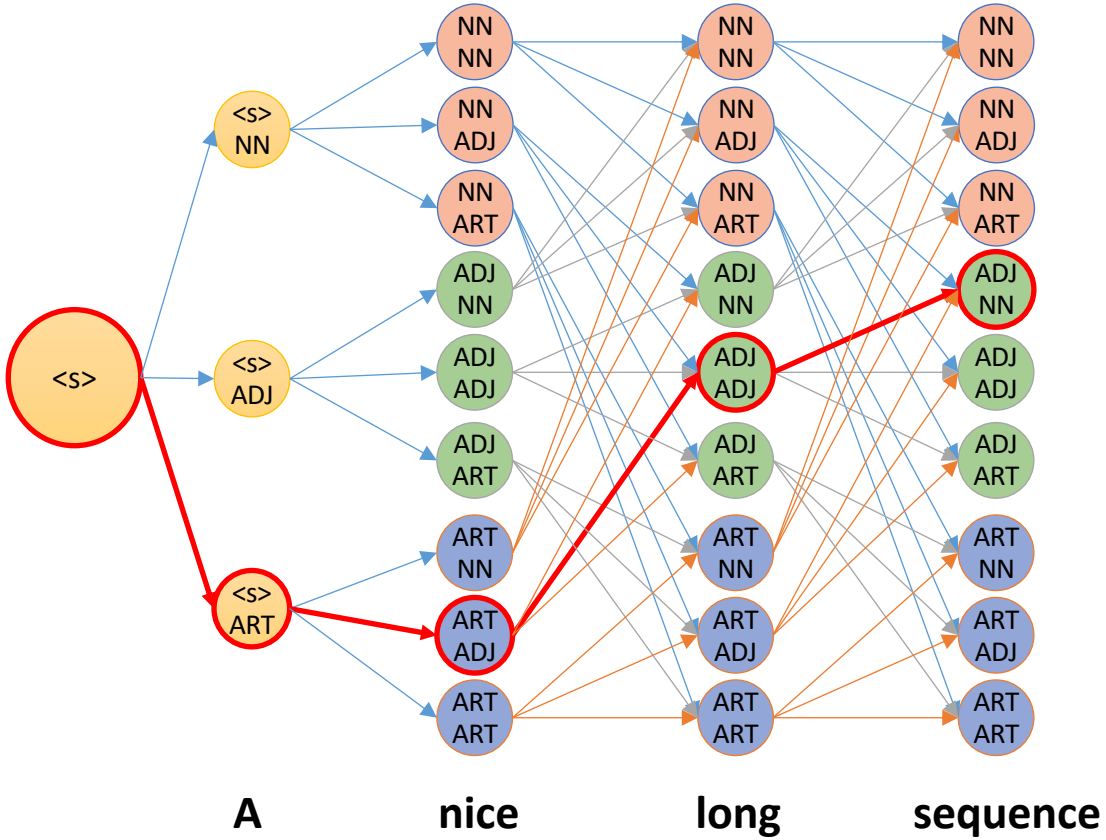
This reads as: We came from a state **NN** and are now in a state **ART**



This reads as: We transition from a state **ART** (which in turn came from a state **NN**) into a state **ADJ** (which came from a state **ART**)

HMM of order 2

- Then the Viterbi algorithm will work as expected!



HMM of order k

- Why stop there? We could now easily power up our model to an arbitrary edge score of order k

- However, if we set k too high, then the counts:

$$s_E(t_i, t_{i-1}, t_{i-2}, \dots, t_{i-k}) = \ln \frac{C(t_i, t_{i-1}, t_{i-2}, \dots, t_{i-k})}{C(t_{i-1}, t_{i-2}, \dots, t_{i-k})}$$

- Will be 0 almost all the time!
  (e.g. observing the specific sequence: ADJ,ADJ,ADJ,ART,ART,NN,NN)

➔ We thought we improved our model, but we hit another wall!

HMM of order k

- Let us rewind a little

$$s_E(t_i, t_{i-1}, t_{i-2}, \dots, t_{i-k}) = \ln \mathrm{P}(t_i | t_{i-1}, t_{i-2}, \dots t_{i-k})$$

➔ We could think of a better way to calculate this probability

HMM of order k

- Interpolation: „We back off to lower N-grams“
- We can model the probability as a weighted sum as follows: (e.g. for trigrams)

3-gram estimate 2-gram estimate 1-gram estimate

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

With: $\lambda_3 + \lambda_2 + \lambda_1 = 1$

- The $\hat{P}$ refers to the probability we would get from counting

HMM of order k

- Calculating λs:
- We count from corpus again
- We reward the most accurate N-gram estimate
  - usually the estimates with larger N:
    more information means better prediction power
- Subtract 1 to prevent overfitting

function DELETED-INTERPOLATION(*corpus*) **returns** $\lambda_1, \lambda_2, \lambda_3$

$\lambda_1, \lambda_2, \lambda_3 \leftarrow 0$
foreach trigram t_1, t_2, t_3 with $C(t_1, t_2, t_3) > 0$
  depending on the maximum of the following three values
    case $\frac{C(t_1,t_2,t_3)-1}{C(t_1,t_2)-1}$: increment λ_3 by $C(t_1, t_2, t_3)$
    case $\frac{C(t_2,t_3)-1}{C(t_2)-1}$: increment λ_2 by $C(t_1, t_2, t_3)$
    case $\frac{C(t_3)-1}{N-1}$: increment λ_1 by $C(t_1, t_2, t_3)$
  end
end
normalize $\lambda_1, \lambda_2, \lambda_3$
return $\lambda_1, \lambda_2, \lambda_3$

(with division by 0 yielding 0)

HMM of order k

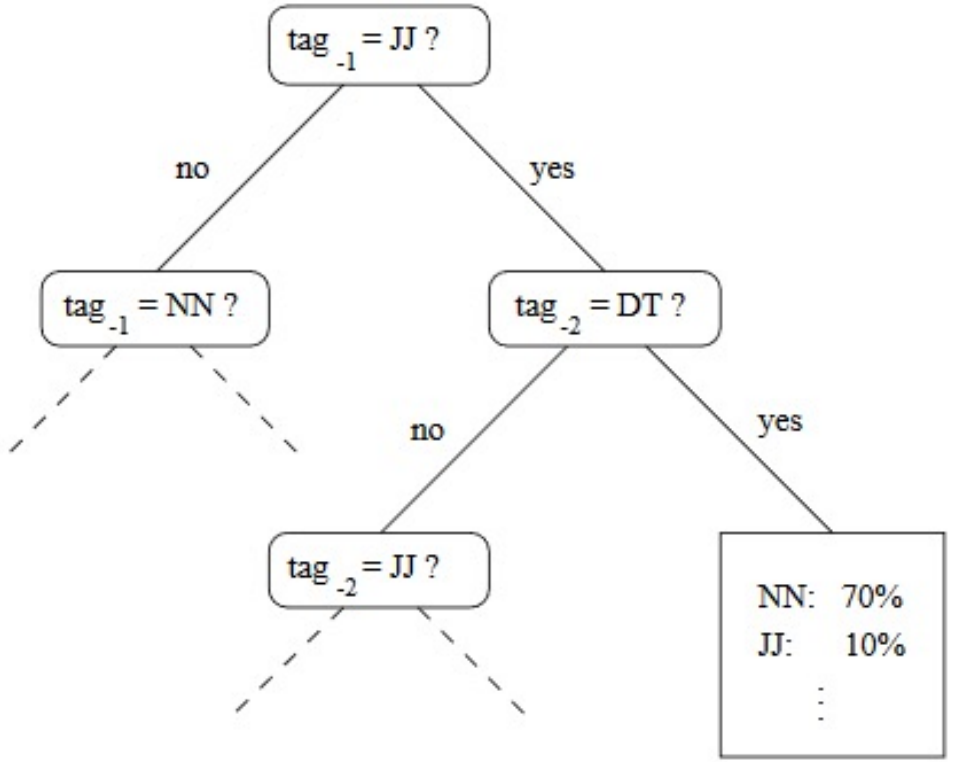
- Let us rewind a little

$$s_E(t_i, t_{i-1}, t_{i-2}, \dots, t_{i-k}) = \ln \mathrm{P}(t_i | t_{i-1}, t_{i-2}, \dots t_{i-k})$$

➔ In general, you could apply any classifier, which produces a probability

HMM of order k

- The TreeTagger uses a Decision Tree to estimate the probabilities:



Taken from: https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger2.pdf

Issues of our current model

$$\underset{t_1^n}{\operatorname{argmax}} \sum_i s_N(t_i, i) + s_E(t_i, t_{i-1})$$

- Our edge score show the Markov-property, where a tag t_i is only dependent on the previous tag t_{i-1}, however this is only a crude approximation

➔We extended this to a HMM of order k and found reliable ways to get the scores (or the probabilities)

➔ What about the node score?

Issues of our current model

$$\underset{t_1^n}{\operatorname{argmax}} \sum_i s_N(t_i, i) + s_E(t_i, t_{i-1})$$

- Since the node score is yet again just a simple probability we could replace it with any classifier of our choice, this could result in:
  - Naive Bayes HMM
  - SVM-HMM
  - Maximum Entropy HMM
  - …
- We will take a first look at the most straight forward extension here

Unknown words

$$\underset{t_1^n}{\operatorname{argmax}} \sum_i s_N(t_i, i) + s_E(t_i, t_{i-1})$$

$$s_N(t_i, i) = \ln \frac{C(t_i, w)}{C(t_i)}$$

Problematic if we deal with a word we have never seen during training, since we would get $-\infty$

Unknown words

$$s_N(t_i, i) = \ln \frac{C(t_i, w)}{C(t_i)}$$

- We can solve that in a similar fashion, using backoff, but this time we backoff using character N-grams:

e.g. unknown word
"nonconformities"

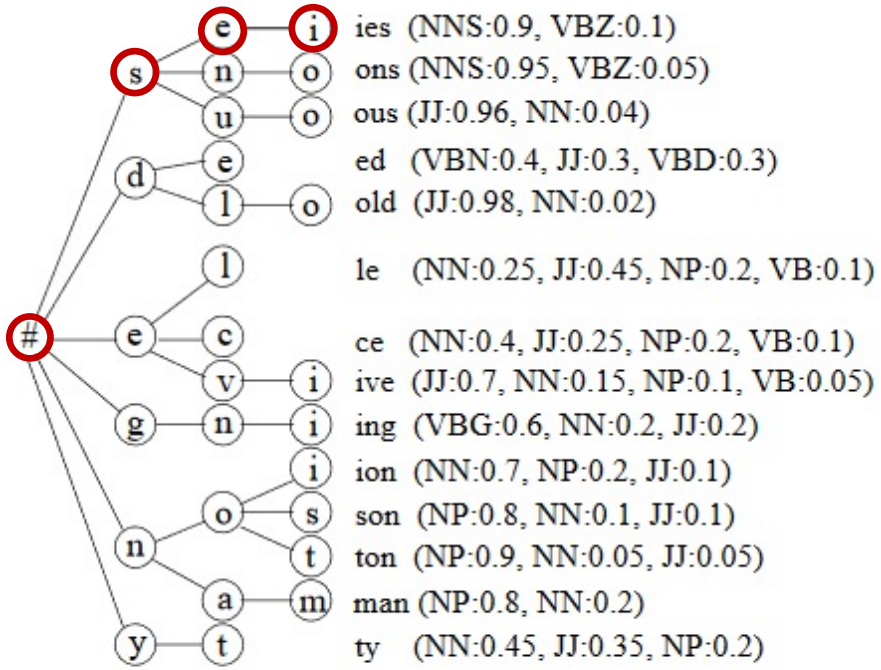


Figure 2: A sample suffix tree of maximal length 3.

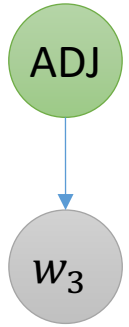
Taken from: https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger2.pdf

A more sophisticated extension of the node score

$$\underset{t_1^n}{\operatorname{argmax}} \sum_i s_N(t_i, i) + s_E(t_i, t_{i-1})$$

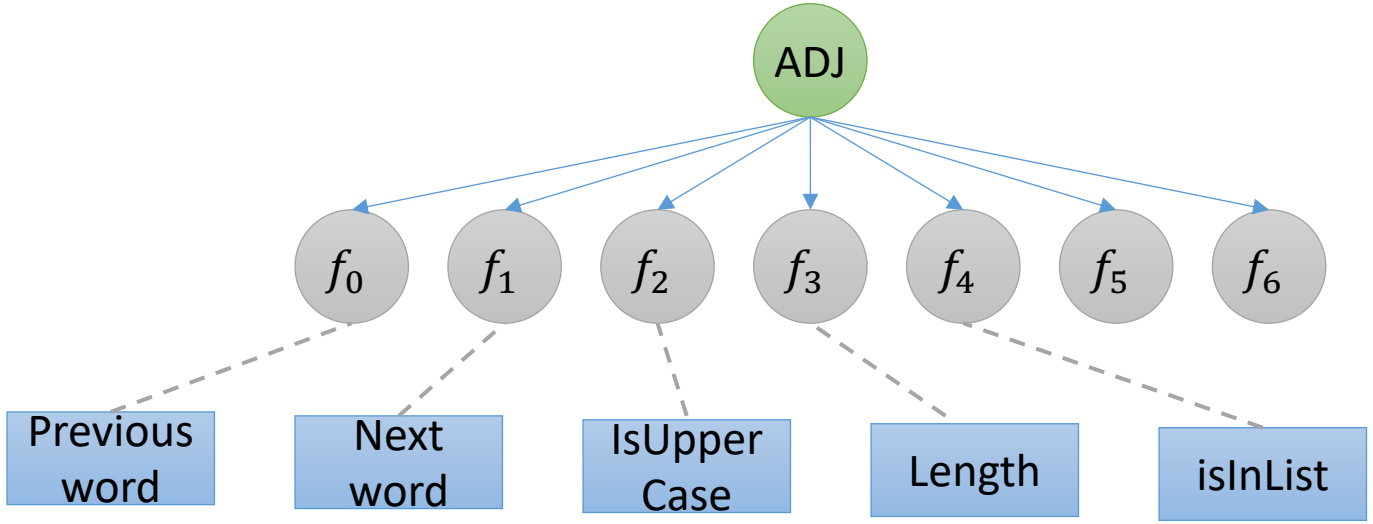
- Our current tree approach only comes into play, if we encounter an unknown word
- We could yet again experiment with different backoff or **smoothing** techniques, but being able to integrate a classifier is the more potent approach
- But why is it more potent?

A classifier for the node score



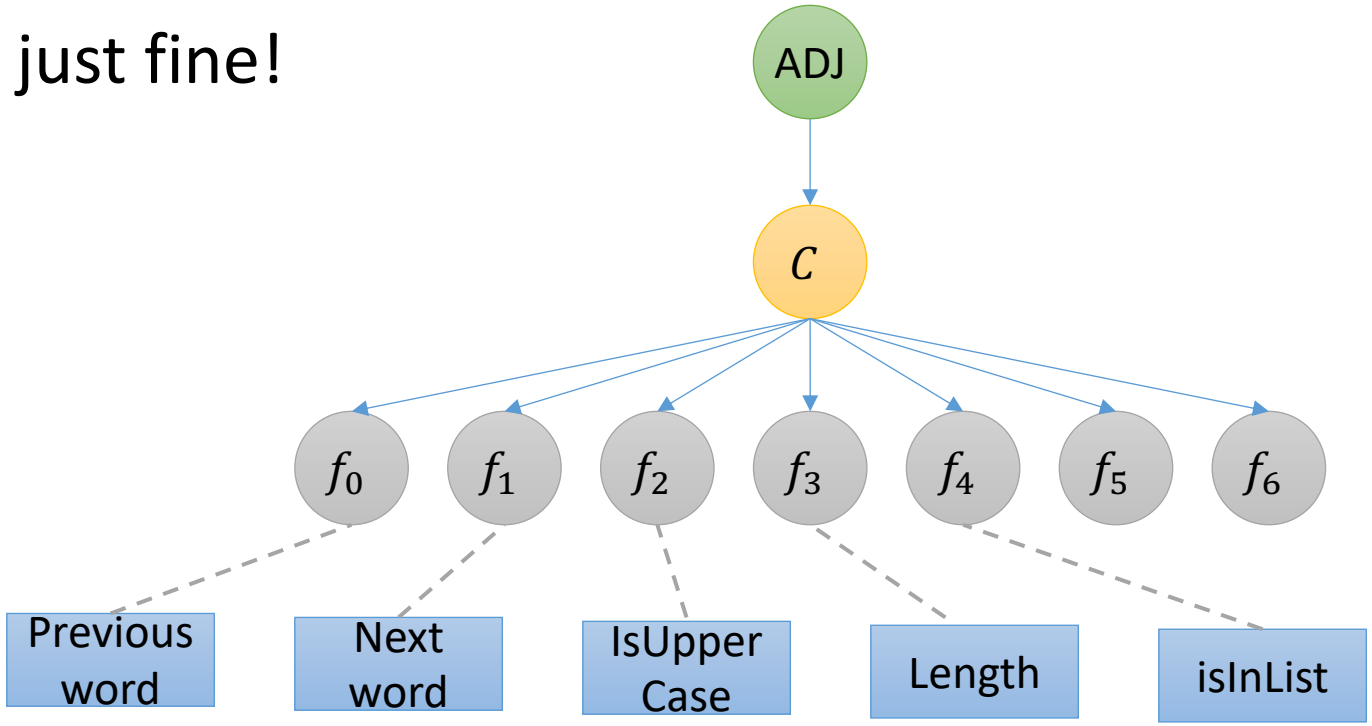
- For the node score, we currently consider this:

- But we would prefer to add many features f_i



A classifier for the node score

- Combining them into a single probability distribution is not easy
- But a classifier C can do this just fine!



Outlook

- In the next lecture, we will introduce the Maximum Entropy classifier

➔ can be directly plugged in as our node score

- We will then extend HMM to deal with the structured problem we are facing here, and we will arrive at
  - Maximum Entropy Markov Models (MEMM)
  - Conditional Random Fields (CRF)

Basic Machine Learning

- At this point, we assume Master's students are familiar with the basic Machine Learning principle of learning parameters through Gradient Descent, as taught in courses such as "Data Mining" and "Machine Learning"

- We provide additional slides and video material regarding Gradient Descent and related basic Machine Learning concepts for students to (re)familiarize themselves