Prof. Dr. Andreas Hotho,
M.Sc. Janna Omeliyanenko
Lecture Chair X for Data Science, Universität Würzburg

3. Exercise for "Sprachverarbeitung und Text Mining"

19.11.2022

1 Knowledge Questions

1. What is the difference between first-order Markov models and k-order Markov models?

   In first-order Markov models, a tag t_i depends only on the previous tag t_{i-1}. In k-order Markov models, the tag t_i depends on the k previous tags t_{i-1}, t_{i-2}, ... t_{i-k}.

2. What motivates the usage of order-k Markov models?

   First-order Markov models assume that the current state t_i depends only on the previous state t_{i-1}. However, this is only a rough approximation. Higher order Markov models are more powerful models with a longer history (dependence on the previous k tags) that allow for higher prediction accuracy.

3. What is the problem concerning the data, when Markov models of too high order k are applied? What method did you learn in the lecture to address this problem? What is the intuition behind this method?

   Data sparsity. Any particular sequence of tags $ti_k, \ldots, ti_1, t_i$ that occurs in the test set may simply never have occurred in the training set. The counts for many specific sequences will be 0. This problem can be addressed with interpolation. Intuition: Estimate the probability by combining more robust, but weaker estimators. For example, if we've never seen the tag sequence `P RP V BT O`,

and so can't compute `P(T O|P RP, V B)` from this frequency, we still could rely on the bigram probability `P(T O|V B)`, or even the unigram probability `P(T O)`.

4. Is the application of the Viterbi-algorithm to Markov models of higher order more computationally expensive than to lower order models for the same POS-tag set with size T? Justify your answer.

The computational complexity of Viterbi is $O(N^2S)$, where N is the size of state space of the Markov model and S is the length of the observation sequence. The size of the state space of the Markov model of order k can be calculated depending on the model order k and the size of the POS tag set T as follows: $N = T^k$ Hence, as the model order k increases, the number of states N grows and so does the computational effort.

5. Why must the condition $\lambda_3+\lambda_2+\lambda_1 = 1$ be satisfied for the following interpolation from the lecture: $P(t_i|t_{i-1}, t_{i-2}) = \lambda_3\hat{P}(t_i|t_{i-1}, t_{i-2}) + \lambda_2\hat{P}(t_i|t_{i-1}) + \lambda_1\hat{P}(t_i)$?

This requirement ensures that the resulting P is a probability distribution.

6. In the lecture we have transformed our optimization problem of finding the most probable sequence of POS-tags given an observed word sequence into the logarithmic space, forming the sum of logarithms of probabilities instead of the product over probabilities. Due to the monotonicity property of the logarithmic function, the maximum after transformation into the logarithmic space does not change. Given that a function $f : A \to \mathbb{R}$, where A is a subset of $\mathbb{R}$, is called strictly monotonically increasing if for all $(x, y) \in D$ with $x < y$ it holds that $f(x) < f(y)$, prove that the logarithm function is strictly monotonically increasing on the interval $(0, inf)$.

$ln(x) < ln(y) \to ln(x) - ln(y) < 0$
$ln(x) - ln(y) = \frac{x}{y} < 0$
$x < y \to 0 < \frac{x}{y} < 1$ and $ln(\frac{x}{y}) < 0$, thus $ln(x) < ln(y)$

7. Why does the integration of a classifier for the estimation of the node score make for a more potent Markov model?

The integration of classifiers allows the use of many different features to describe the data samples and thus better generalization and handling of unknown words. Combining multiple features into a single probability distribution is not straightforward, but can be modelled by a classifier.

8. Given are the following table with the features, the counts observed in the dataset, and the counts predicted by two different trained classifiers CI and CII. Which of the two classifiers better reflects the data? Justify your answer.

Feature-Name = Value $\wedge$ CurrentLabel	Observed counts	CI predicted counts	CII predicted counts
Current Word = $w_1\wedge$ ADJ	15	14.3	12.7
Current Word = $w_1\wedge$ N	37	32.8	32.2
Current Word = $w_1\wedge$ V	123	118.9	121
Current Word = $w_2\wedge$ ADJ	43	38	38
Current Word = $w_2\wedge$ N	1225	1222	1122
Current Word = $w_2\wedge$ V	0	0.3	11
Previous Word = $w_1\wedge$ ADJ	246	233.7	2463
Previous Word = $w_1\wedge$ N	1	0.9	111
Previous Word = $w_1\wedge$ N	587	587	7.2

error(CI) = $|15-14.3|+|37-32.8|+|123-118.9|+|43-38|+|1225-1222|+|0-0.3|+|246-233.7|+|1-0.9|+|587-587|=3.3$
error(CII) = $|15-12.7|+|37-32.2|+|123-121|+|43-38|+|1225-1122|+|0-11|+|246-2463|+|1-111|+|587-7.2|=337.2$
Classifier I reflects the data better

2 POS Tagging - but backwards

Once again, consider the following highly simplified set of generalized word types (tag-set) for part-of-speech tagging, where words are modeled as observations and associated word types as hidden variables:

DET	Determiner	the,a,...
N	Noun	year,home,costs,time,...
PRO	Pronoun	he,their,you
V	Verb	said,took,saw

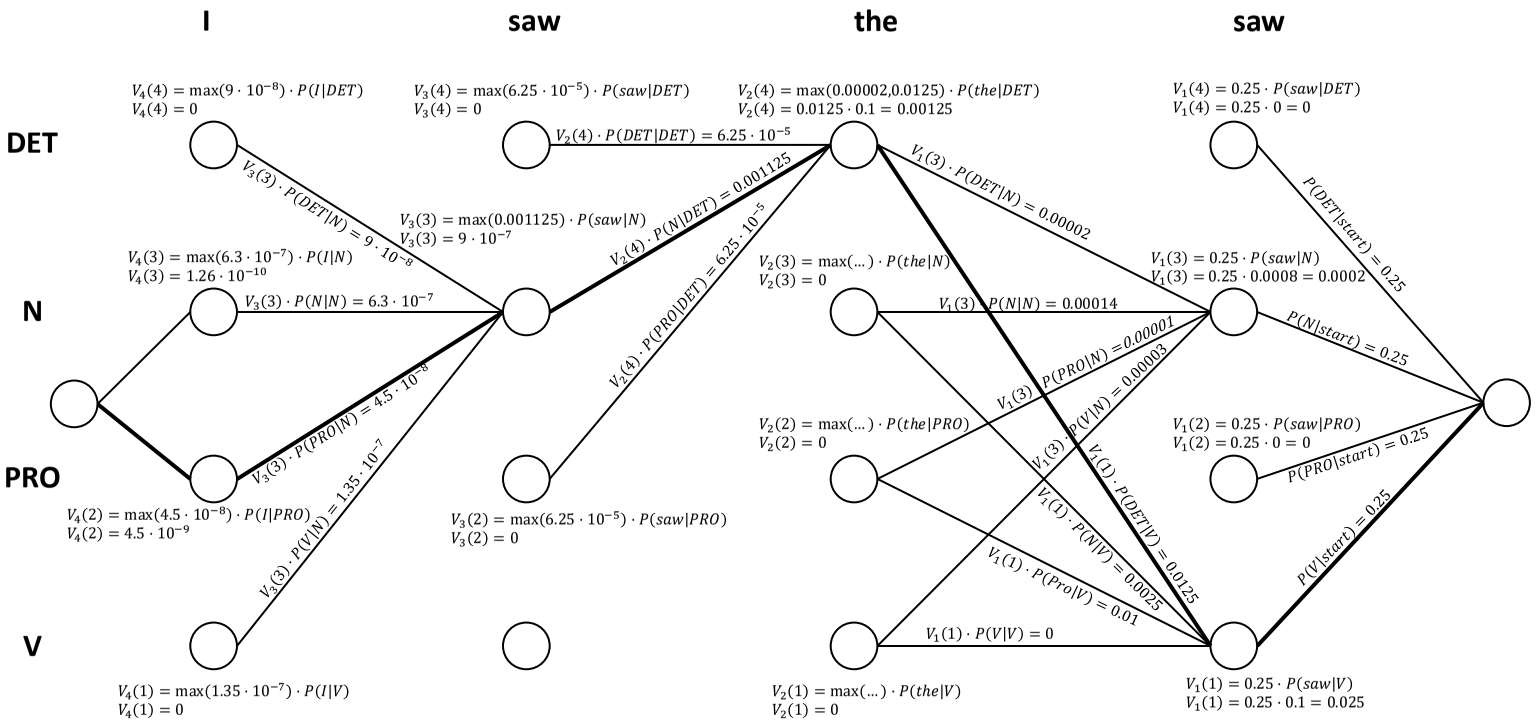
Once again, using the table below, calculate the most likely sequence of tags for the following sentence:
„I saw the saw“

1. Apply the Viterbi algorithm. This time however, apply the algorithm backwards, so that the starting state is located at the end of the sequence and transition probabilities are applied from the current to the previous observation instead of the next.
2. Which changing behavior do you observe in comparison to the last exercise sheet?

Assume uniformly distributed start transition probabilities.

from\to	DET	N	PRO	V
DET	0.05	0.9	0.05	0
N	0.1	0.7	0.05	0.15
PRO	0.05	0.5	0.05	0.4
V	0.5	0.1	0.4	0

w\t	DET	N	PRO	V
I	0	0.0002	0.1	0
saw	0	0.0008	0	0.1
the	0.1	0	0	0


I
saw
the
saw
DET
N
PRO
V
$V_4(4) = \max(9 \cdot 10^{-8}) \cdot P(I|DET)$
$V_4(4) = 0$
$V_3(4) = \max(6.25 \cdot 10^{-5}) \cdot P(saw|DET)$
$V_3(4) = 0$
$V_2(4) = \max(0.00002, 0.0125) \cdot P(the|DET)$
$V_2(4) = 0.0125 \cdot 0.1 = 0.00125$
$V_1(4) = 0.25 \cdot P(saw|DET)$
$V_1(4) = 0.25 \cdot 0 = 0$
$V_2(4) \cdot P(DET|DET) = 6.25 \cdot 10^{-5}$
$V_3(3) \cdot P(DET|N) = 9 \cdot 10^{-8}$
$V_3(3) = \max(0.001125) \cdot P(saw|N)$
$V_3(3) = 9 \cdot 10^{-7}$
$V_2(4) \cdot P(N|DET) = 0.001125$
$V_1(3) \cdot P(DET|N) = 0.00002$
$P(DET|start) = 0.25$
$V_4(3) = \max(6.3 \cdot 10^{-7}) \cdot P(I|N)$
$V_4(3) = 1.26 \cdot 10^{-10}$
$V_3(3) \cdot P(N|N) = 6.3 \cdot 10^{-7}$
$V_2(3) = \max(\ldots) \cdot P(the|N)$
$V_2(3) = 0$
$V_1(3) \cdot P(N|N) = 0.00014$
$V_1(3) = 0.25 \cdot P(saw|N)$
$V_1(3) = 0.25 \cdot 0.0008 = 0.0002$
$P(N|start) = 0.25$
$V_2(4) \cdot P(PRO|DET) = 6.25 \cdot 10^{-5}$
$V_3(3) \cdot P(PRO|N) = 4.5 \cdot 10^{-8}$
$V_1(3) \cdot P(PRO|N) = 0.00001$
$V_1(3) \cdot P(V|N) = 0.00003$
$V_2(2) = \max(\ldots) \cdot P(the|PRO)$
$V_2(2) = 0$
$V_1(2) = 0.25 \cdot P(saw|PRO)$
$V_1(2) = 0.25 \cdot 0 = 0$
$P(PRO|start) = 0.25$
$V_4(2) = \max(4.5 \cdot 10^{-8}) \cdot P(I|PRO)$
$V_4(2) = 4.5 \cdot 10^{-9}$
$V_3(2) = \max(6.25 \cdot 10^{-5}) \cdot P(saw|PRO)$
$V_3(2) = 0$
$V_1(1) \cdot P(DET|V) = 0.0125$
$V_1(1) \cdot P(N|V) = 0.0025$
$V_1(1) \cdot P(Pro|V) = 0.01$
$P(V|start) = 0.25$
$V_3(3) \cdot P(V|N) = 1.35 \cdot 10^{-7}$
$V_1(1) \cdot P(V|V) = 0$
$V_4(1) = \max(1.35 \cdot 10^{-7}) \cdot P(I|V)$
$V_4(1) = 0$
$V_2(1) = \max(\ldots) \cdot P(the|V)$
$V_2(1) = 0$
$V_1(1) = 0.25 \cdot P(saw|V)$
$V_1(1) = 0.25 \cdot 0.1 = 0.025$

> The algorithm can be applied backwards all the same, but some of our transition probabilities from the given table probably require updating since they not match this scenario (i.e. verbs cant come bevore determiners).