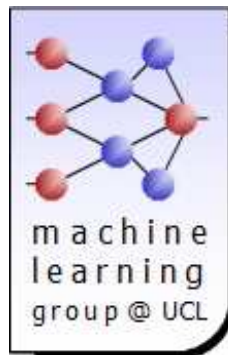


Noisy Sequence Classification with Smoothed Markov Chains

Pierre.Dupont@uclouvain.be



Motivation

Markov chains (MCs) are among the simplest models of sequential processes

They are often used to **predict**, in a probabilistic sense, the next symbol in a sequence (e.g. language modeling task)

We are interested here in **sequences** such as texts, DNA fragments, protein primary structures,... to be **classified** into a set of predefined categories

Motivation

Markov chains (MCs) are among the simplest models of sequential processes

They are often used to **predict**, in a probabilistic sense, the next symbol in a sequence (e.g. language modeling task)

We are interested here in **sequences** such as texts, DNA fragments, protein primary structures,... to be **classified** into a set of predefined categories

Our approach is intended to be

- simple conceptually
- accurate
- highly efficient computationally
- highly robust to classification noise

We are ready to trade off some level of accuracy for higher robustness

Questions

Good predictive models assigns a high likelihood $P(x|M)$ to a newly observed sequence x or, equivalently, a low perplexity $2^{-\left[\frac{1}{|x|} \log_2 P(x|M)\right]}$

According to Bayes decision theory, maximum likelihood (ML) models are optimal classifiers when class priors are uniform

Questions

Good predictive models assigns a high likelihood $P(x|M)$ to a newly observed sequence x or, equivalently, a low perplexity $2^{-\left[\frac{1}{|x|} \log_2 P(x|M)\right]}$

According to Bayes decision theory, maximum likelihood (ML) models are optimal classifiers when class priors are uniform

- In practice, smoothing ML models is required. If a smoothed model is a better predictor, how about its ability to classify new sequences?
- Is there a relation between perplexity and classification accuracy?
- What is the optimal model order?
- What about the robustness to classification noise?

Questions (ctd.)

An alternative approach to noisy sequence classification was proposed in [Sebban et al. 2004]:

- an automaton induction technique designed to deal with classification noise
- confidence of the class labels evaluated with a Markov chain model
- this information is subsequently introduced in a boosting scheme

What is the classification accuracy if the confidence oracle (= a Markov chain) is used alone?

Markov Chains

A discrete time *Markov Chain (MC)* is a stochastic process $\{X_t | t \in \mathbb{N}\}$ where the random variable X takes its value at any discrete time t in a countable set W and such that:

$$P[X_t = w | X_{t-1}, X_{t-2}, \dots, X_0] = P[X_t = w | X_{t-1}, \dots, X_{t-p}] \triangleq P(w|h)$$

where $h \triangleq X_{t-1}, \dots, X_{t-p}$ is the relevant history and p is the model order

The set W corresponds to the alphabet when modeling sequences

Markov Chains

A discrete time *Markov Chain (MC)* is a stochastic process $\{X_t | t \in \mathbb{N}\}$ where the random variable X takes its value at any discrete time t in a countable set W and such that:

$$P[X_t = w | X_{t-1}, X_{t-2}, \dots, X_0] = P[X_t = w | X_{t-1}, \dots, X_{t-p}] \triangleq P(w|h)$$

where $h \triangleq X_{t-1}, \dots, X_{t-p}$ is the relevant history and p is the model order

The set W corresponds to the alphabet when modeling sequences

Maximum likelihood estimation

$$\hat{P}(w|h) = \begin{cases} \frac{C(h,w)}{C(h)} & \text{if } C(h) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $C(h)$ (resp. $C(h, w)$) is the number of times the history h (followed by w) has been observed in the training set

Example: $C(ab) = 10; C(abc) = 3 \Rightarrow \hat{P}(c|ab) = \frac{3}{10}$

The need for smoothing

Maximum likelihood estimation assigns a zero probability to any unseen event in the training set ($C(h, w) = 0$)

$$\hat{P}(w|h) = \begin{cases} \frac{C(h,w)}{C(h)} & \text{if } C(h) > 0 \\ 0 & \text{otherwise} \end{cases}$$

A Markov chain of order $N - 1$ (also known as a N -gram model) built on an alphabet W defines $|W|^N$ possible events

⇒ the number of parameters grows exponentially with the model order

⇒ even for very large training sets, many possible events are assigned a zero probability and the probability of observed events is overestimated

⇒ smoothing techniques are required to correct the ML estimation

Smoothed Markov Chains

$$\hat{P}(w|h) = \begin{cases} \frac{C(h,w) - d_c}{C(h)} + \gamma(h) \hat{P}_{back}(w|h) & \text{if } C(h,w) > 0, \\ \gamma(h) \hat{P}_{back}(w|h) & \text{if } C(h,w) = 0 \text{ and } C(h) > 0, \\ \hat{P}_{back}(w|h) & \text{if } C(h) = 0, \end{cases}$$

with

$$\gamma(h) = \sum_{w: C(h,w) > 0} \frac{d_c}{C(h)}$$

$\gamma(h)$ is a probability mass discounted from seen events, depending on the discounting coefficients d_c , and distributed over unseen events proportionally to a back-off distribution ($\hat{P}_{back} = \hat{P}(w|h_{-1})$, in the simplest case)

Back-off smoothing was shown to outperform significantly **Laplace smoothing**

Underlying principles

- **Symmetry:** any two symbols having the same count value c in the training set should be assigned the same probability $\hat{P}_c \Rightarrow$ events are clustered into equivalence classes based on their counts
- A **leave-one-out** (LOO) procedure simulates unseen events, for events appearing once in the original dataset. Optimal discounting coefficients can be deduced by maximum likelihood estimate on leave-one-out samples. The Turing-Good estimate derives from this principled approach.
- **Monotony:** $\hat{P}_{c-1} \leq \hat{P}_c$ (not necessarily satisfied by Turing-Good)
- **Absolute discounting** model $d_c \triangleq d, \forall c$ This model satisfies monotony and an upper bound d_* on the optimal discounting coefficient can be estimated by LOO:

$$d \leq d_* = \frac{n_1}{n_1 + 2n_2}.$$
 with n_i the number of events appearing i times
- A **modified back-off distribution** also follows from the LOO procedure with marginal constraints (see paper)

Sequence likelihood and classification rule

The likelihood $P(x|M)$ of a sequence $x = x_1 \dots x_{|x|}$ according to a Markov Chain M

$$P(x|M) = \prod_{i=1}^{|x|} P(x_i|h, M)$$

Model estimation

A distinct model $\hat{P}(x|C_i)$ is estimated from the training set associated to each class i

Decision rule

$$\hat{C} = \operatorname{argmax}_i \hat{P}(C_i) \hat{P}(x|C_i).$$

The Naive Bayes classifier corresponds to the special case for which h is empty

Datasets

WF dataset: gender classification from proper name spelling

A G A T H E

A L D R I C

A N N E - G A B R I E L L E

A M Y

A M Z A

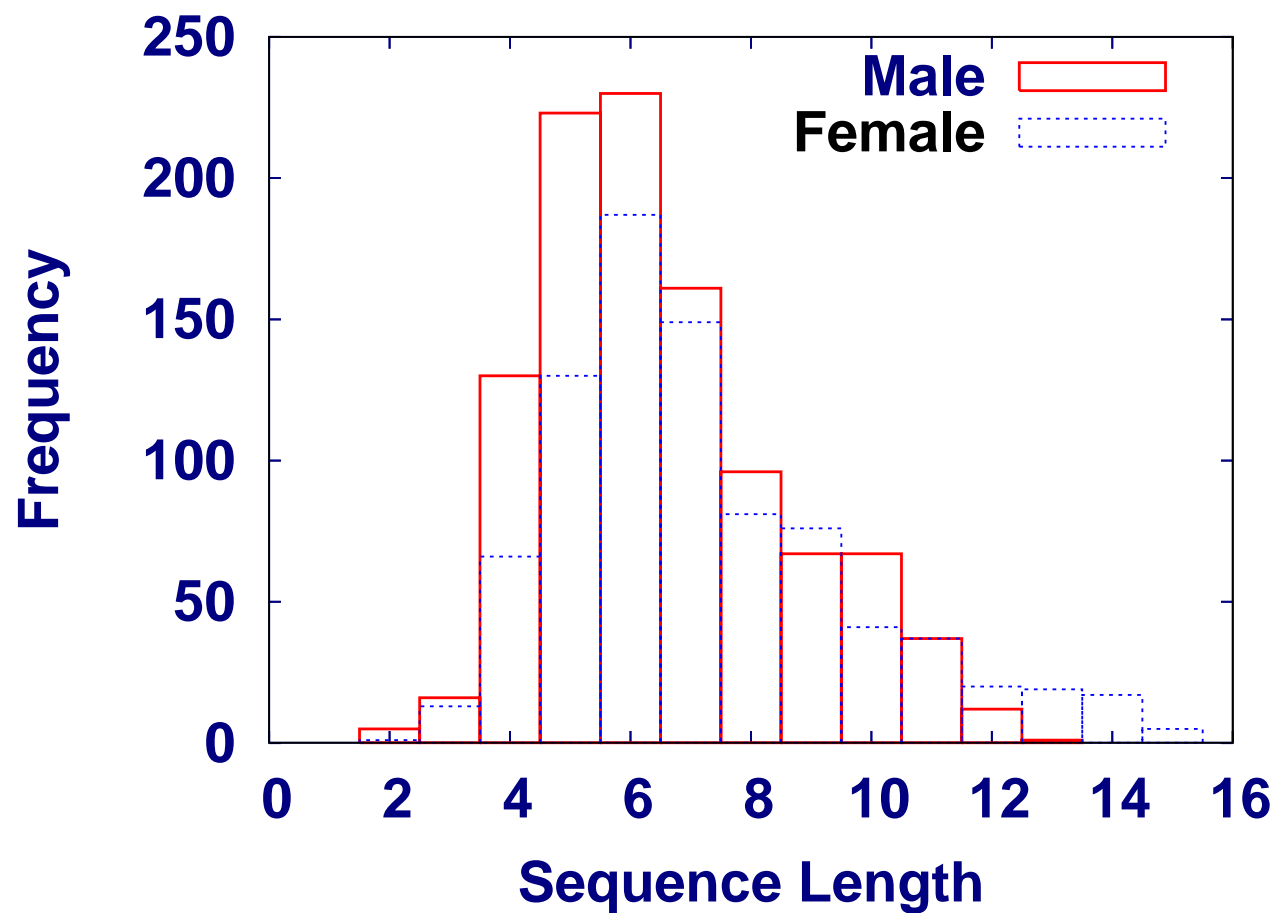
Splice dataset: splicing region classification from DNA sequences

TTCAG ... TCCCA **GG** TCTCTG ... TGTGGA
 exon intron

GACCC ... TGGCAG **G** TAGGAG ... TGCCTC
 intron exon

WF dataset

Alphabet of 27 symbols (26 letters and the symbol –)
Variable sequence length



Class distribution in WF training and test

Random splitting of the data set: 80% training, 20% test

	Dataset	Male	Female	Total
Number of sequences	Training	836	674	1510
Number of distinct sequences	Training	606	674	1280
Number of sequences	Test	209	168	377
Number of distinct sequences	Test	201	168	369

Classification **noise**: from 5% up to 50 % of the training class labels are swapped

Test labels are left unchanged

Splice dataset

Alphabet of 8 symbols (4 nucleotides + 4 ambiguous symbols):

A, T, C, G; D={A, G, T}; N={A, G, C, T}; S={C, G}; R={A, G}

Fixed sequence length of 60 nucleotides

Classify each sequence either as:

- **exon/intron** boundary (Acceptor)
- **intron/exon** boundary (Donor)
- **no** boundary (not considered here)

Class distribution in Splice training and test

Random splitting of the data set: 80% training, 20% test

	Dataset	Donor	Acceptor	Total
Number of sequences	Training	614	614	1228
Number of sequences	Test	126	127	253

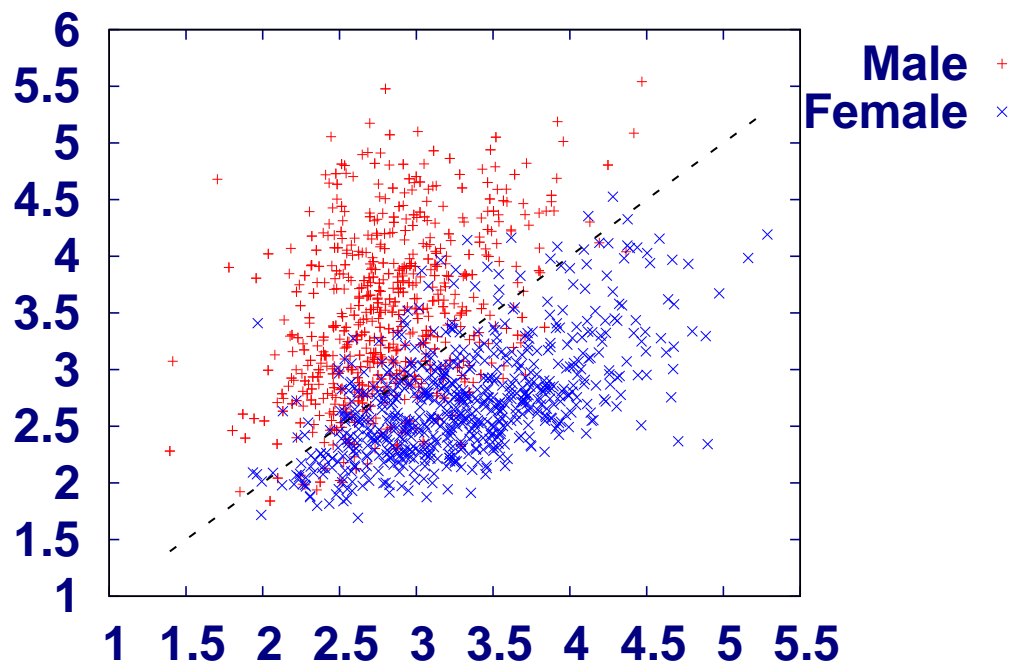
Classification **noise**: from 5% up to 50 % of the training class labels are swapped

Test labels are left unchanged

WF training classification results

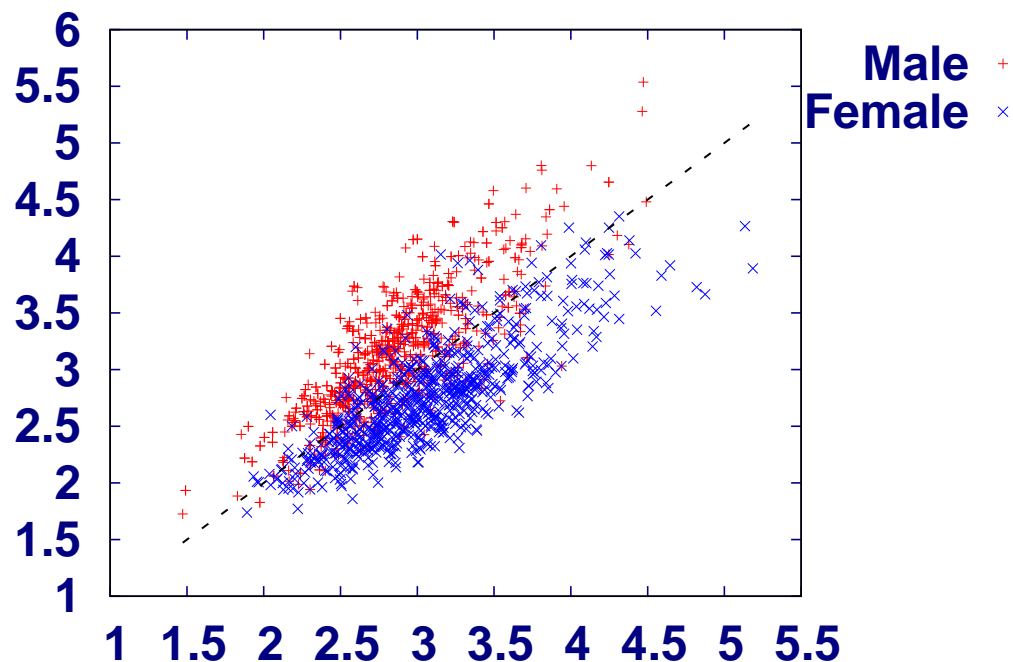
Training set classification results with 0% and 20% classification noise

2-gram Model Training Data



Accuracy: 85.76%

2-gram Model Training Data (20% Noise)

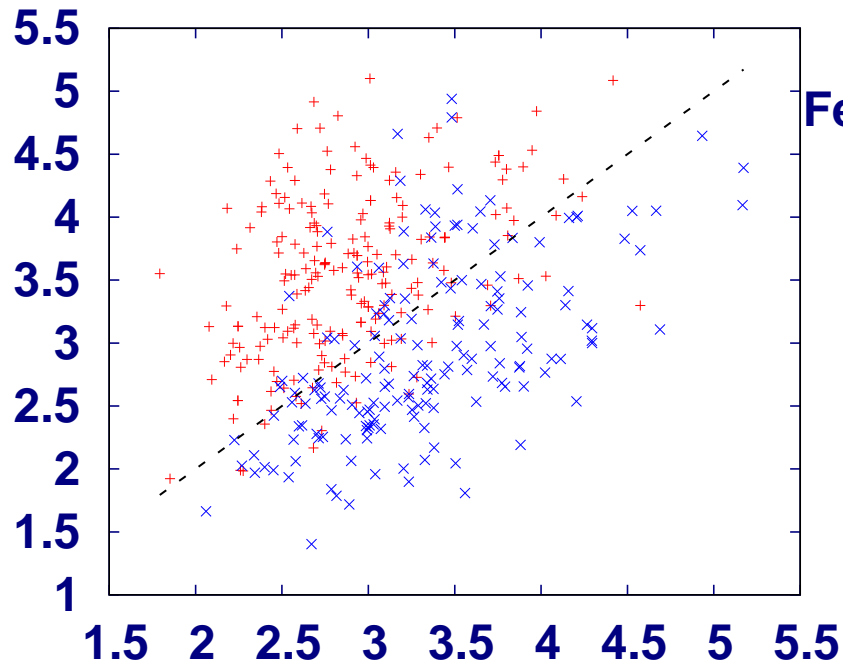


83.18%

WF test classification results

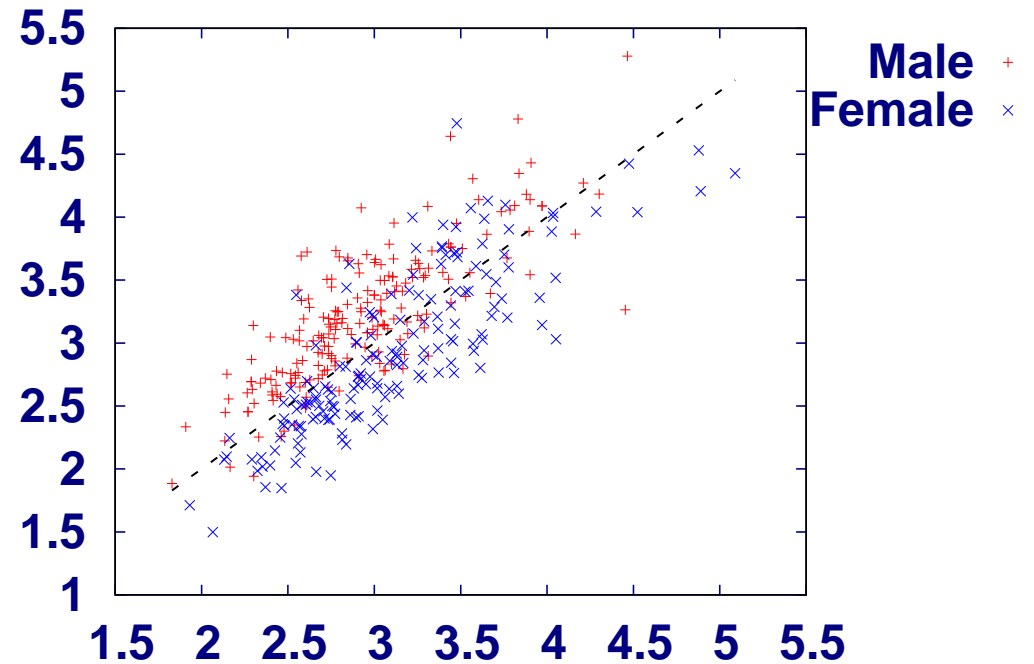
Test set classification results with 0% and 20% classification noise

2-gram Model Test Data



Accuracy: 81.96%

2-gram Model Test Data (20% Noise for Training)



79.31%

WF confusion matrix

Noise = 0%

	Male	Female
Male (predicted)	179	38
Female (predicted)	30	130

Confusion matrix

	Male	Female
Male (predicted)	7.4	10.1
Female (predicted)	12.0	7.3

Perplexity

Noise = 20%

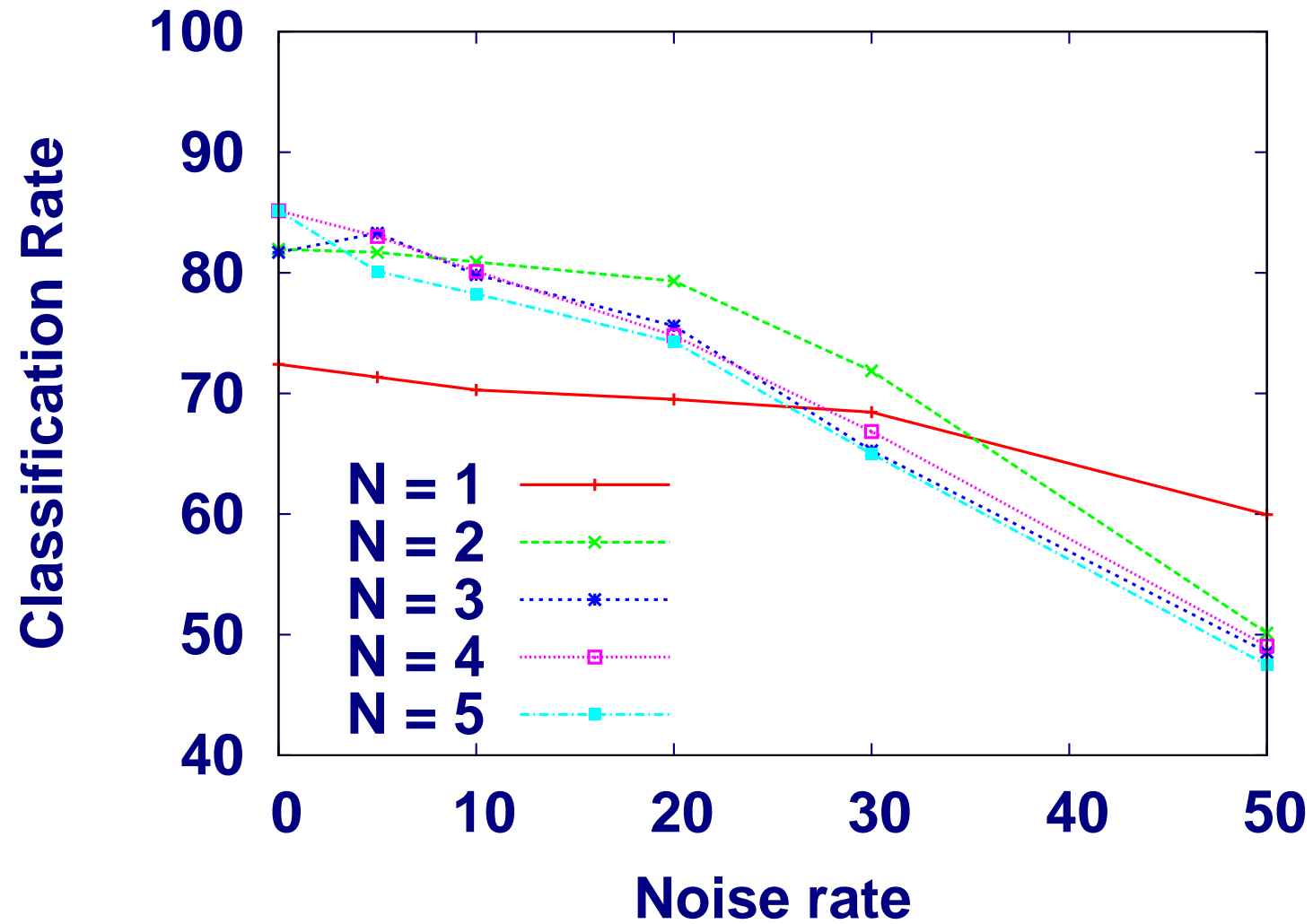
	Male	Female
Male (predicted)	172	41
Female (predicted)	37	127

Confusion matrix

	Male	Female
Male (predicted)	7.6	8.5
Female (predicted)	9.2	7.4

Perplexity

Influence of model order



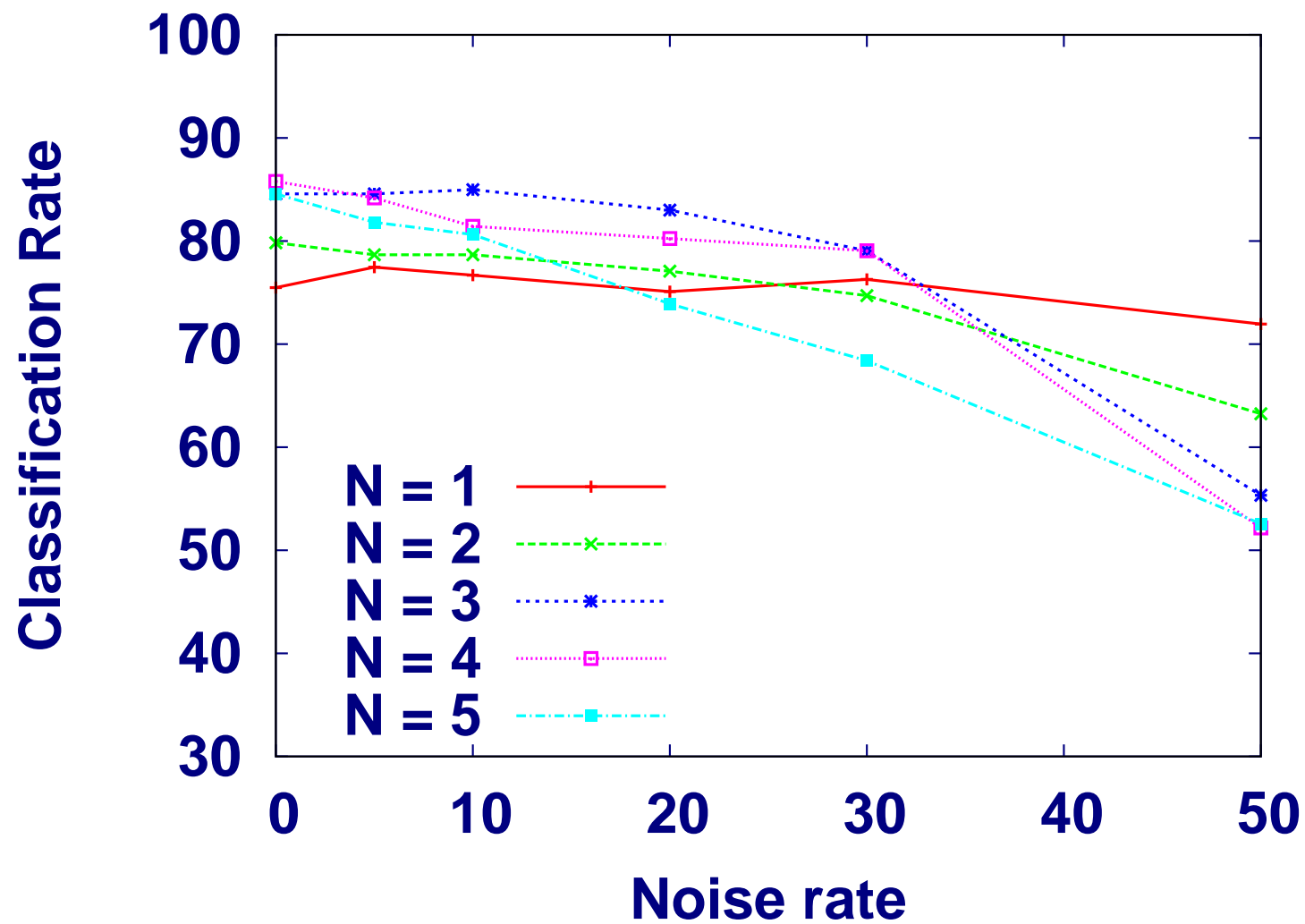
Smoothed Markov Chains versus Automata with Confidence Oracle and Boosting

BOOST refers to the approach described in [Sebban et al. 2004]

Noise rate	2-gram	BOOST
5%	18.3%	21.4%
10%	19.1%	22.7%
20%	20.7%	31.7%
30%	28.1%	?

Test set classification errors on the WF dataset

Splice classification results



Conclusion

Our approach appears to be

- simple conceptually
- accurate (80 % ... 85 % classification accuracy)
- highly efficient computationally (a few seconds to estimate and test a model)
- highly robust to classification noise (stable up to 20 % ... 30 % noise)

We did not have to trade off a lot of accuracy for high robustness

Conclusion

Our approach appears to be

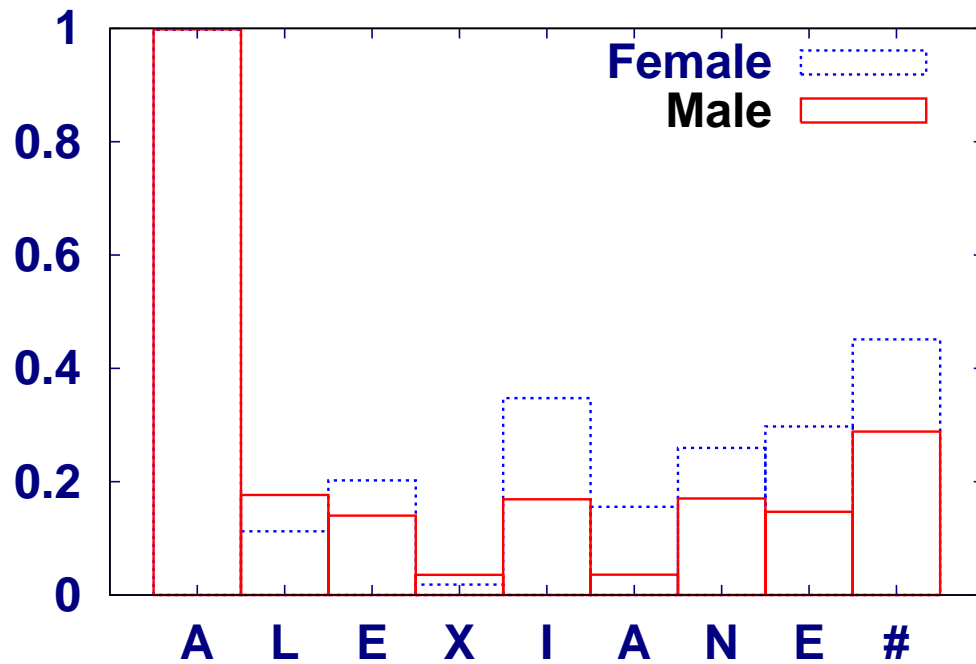
- simple conceptually
- accurate (80 % ... 85 % classification accuracy)
- highly efficient computationally (a few seconds to estimate and test a model)
- highly robust to classification noise (stable up to 20 % ... 30 % noise)

We did not have to trade off a lot of accuracy for high robustness

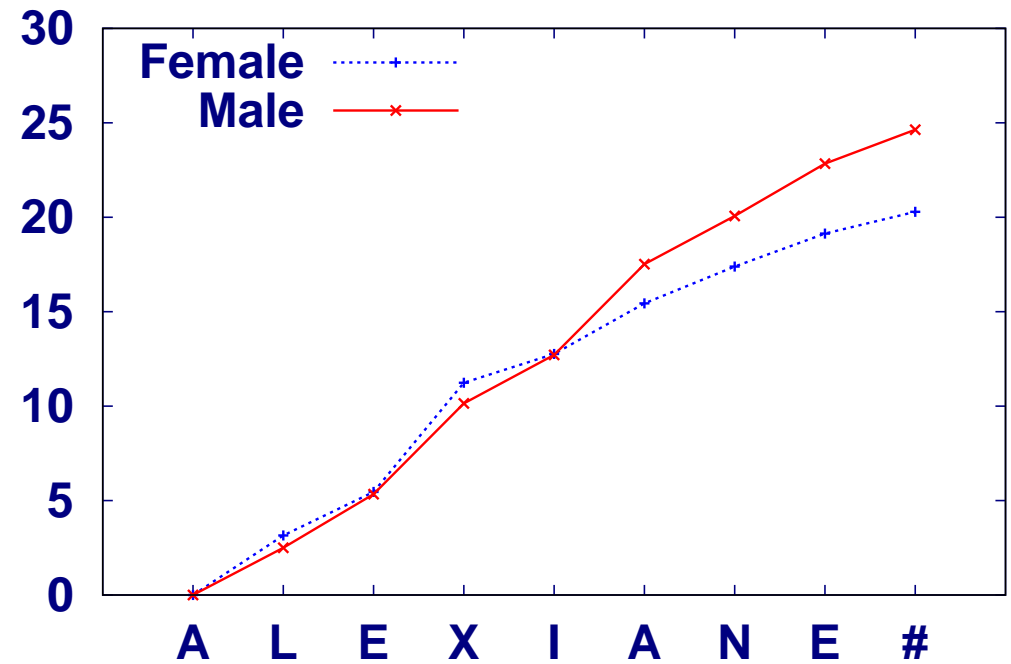
- interpretable

Interpretability

Order 1 Markov Chain probabilities



- Log likelihood



Perspectives

A Markov chain defines a probability $\hat{P}(x_i|h)$ for each symbol x_i in a sequence

These probabilities are combined to form a MAP classifier which extends Naive Bayes by considering a non-empty relevant history

Alternatively, we can consider each feature as a dimension in a new feature space and define an appropriate kernel

A gain in accuracy is expected...

at the price of an increased computational complexity