PLAYER PREDICTION

Seminar am FB Informatik, FG Knowledge Engineering, SS06

Knowledge Engineering und Lernen in Spielen

18/07/2006

René Moch

Übersicht

- Vorbemerkungen
- Pattern Recognition with Sequential Prediction
  - Motivation: Vorhersage des Spielerverhaltens
  - Idee: Mustererkennung in gegebenen Sequenzen
  - Ziel: Vorhersage des Sequenz-Verlaufs
- N-Gram Statistical Models
  - Motivation: Vorhersage des Spielerverhaltens
  - Idee: Statistische Auswertung aufgezeichneter Sequenzen
  - Ziel: Vorhersage der wahrscheinlichsten Folge-Aktion

Vorbemerkungen

- Player werden komplexer
  - realistisches Verhalten
  - realistische Welten
  - Entscheidungen auf Basis komplexer Spielstrategien treffen
- Mix aus „Zufall“ und Vorhersagbarkeit
  - Randomness vs. *Predictability*
  - Ziel der AI: Player Prediction
  - Analyse von Patterns (Mustern), Statistiken

Sequential Prediction

- Strategie: Pattern Recognition

  „Sequential prediction is the problem of finding a reasonable successor for a given sequence.“ [Mom02], p. 568.
- Annahmen:
  - Es existiert eine eindeutige Elemente-Menge *(Alphabet)*
  - Es existiert ein wiederkehrendes Muster (*repetitive pattern)* in einer gegebenen Sequenz
- Menschliches Gehirn ist Spezialist auf dem Gebiet der Mustererkennung

Beispiel: IQ-Test

- Geg.: Alphabet $A = \{0,1\}$,
  Sequenz $\Sigma =$ „**1 0 1 1 0 1 1 0**“
- Ges.: Nachfolger $succ(\Sigma) = \sigma$
- Frage: *Existiert ein nachvollziehbares Muster?*
  Intuitiv: Muster = „**1 0 1**“
  => $\sigma =$ „**1**“
- Warum?
- Intuitive Suche nach der längsten Zeichenkette innerhalb der Sequenz, die nach Anhängen eines potenziellen Nachfolgers ihr Ende matcht...

String-Matching Prediction

- Geg.: Sequenz Ω über A

  Ω = „10010110111000010001101“
- Matches (von R nach L): „**01**“, „**101**“, „**1101**“, ...
- Längster Substring = „**01101**“
- => *succ*(Ω) = „**1**“
- Einfache Implementierung, jedoch schlechte Performanz falls wiederkehrende Muster enthalten sind - Laufzeit **O(N^2)**!

Enhanced String-Matching

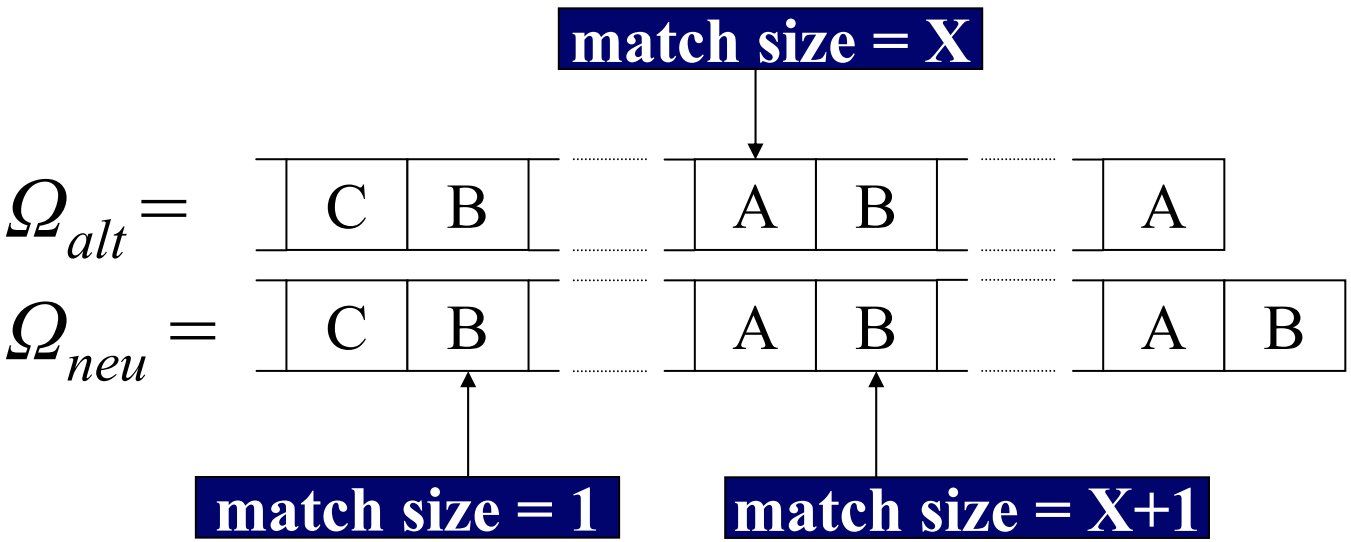
- Berechnung neuer Matches auf Basis vorheriger Matches
- Geg.: Sequenzen Ω_{alt} und Ω_{neu} über $\{A,B,C\}$

$\Omega_{alt} =$ | C | B | | A | B | | A |

$\Omega_{neu} =$ | C | B | | A | B | | A | B |

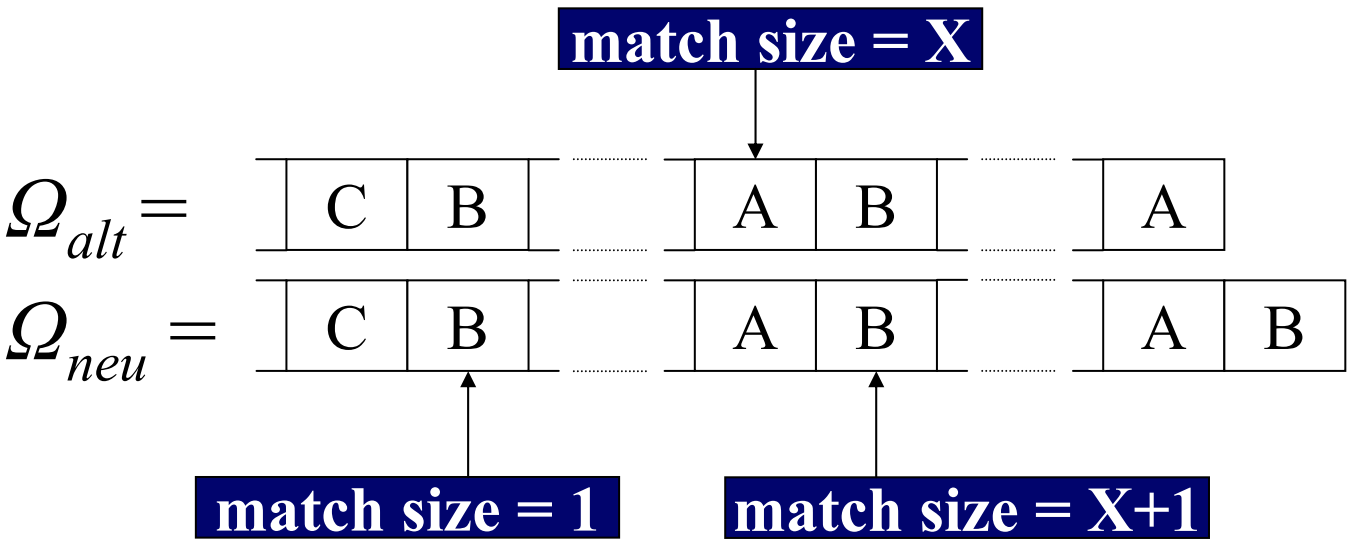
- Betrachtung aller Substrings, die das Ende von Ω_{alt} matchen; sie müssen jedoch mit dem Zeichen „A“ enden!

Enhanced String-Matching



Unter der Annahme, sämtliche match sizes für alle Vorkommen von „A“ in Ω_{alt} seien bekannt, können nun alle match sizes für „B“ in Ω_{neu} berechnet werden. **Laufzeit O(N)**.

Enhanced String-Matching



Fallunterscheidung in allen Substrings, die nun mit B enden:

- *B geht **kein** A voran:* match size := 1 *(entspr. X+1 mit X=0)*
- *B geht ein A voran:* match size := X+1

Enhanced String-Matching at work

Eingabe = „ A B A B B A B “

	Index:	0	1	2	3	4	5	6
N=1	Sequenz	**A**						
	Match size	-						
N=2	Sequenz	A	**B**					
	Match size	-	-					
N=3	Sequenz	A	B	**A**				
	Match size	**1**	-	-				
N=4	Sequenz	A	B	A	**B**			
	Match size	-	**2**	-	-			
N=5	Sequenz	A	B	A	B	**B**		
	Match size	-	**1**	-	**1**	-		
N=6	Sequenz	A	B	A	B	B	**A**	
	Match size	**1**	-	**2**	-	-	-	
N=7	Sequenz	A	B	A	B	B	A	**B**
	Match size	-	**2**	-	**3**	**1**	-	-

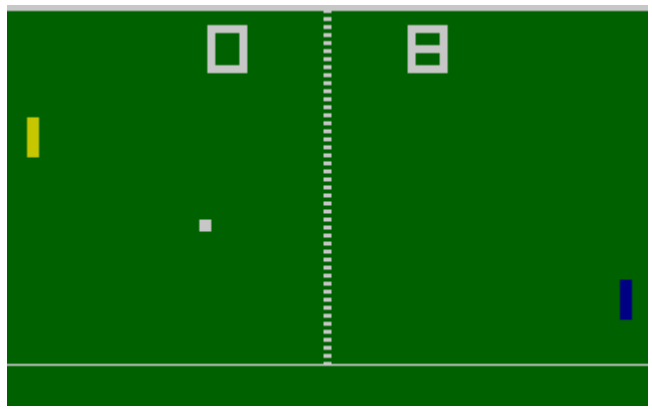
Enhanced String-Matching at work

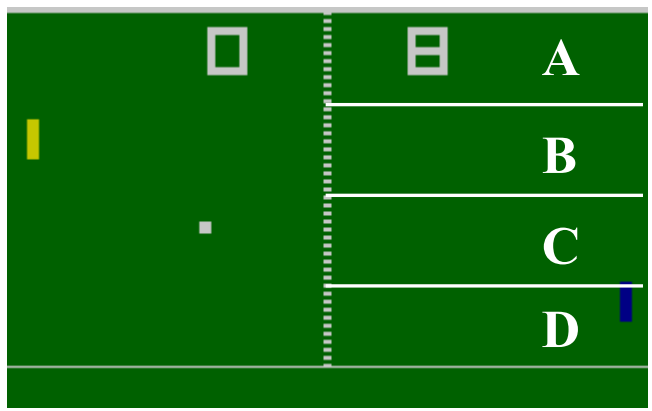
Eingabe = „ A B A B B A B “

	Index:	0	1	2	3	4	5	6
N=1	Sequenz	A						
	Match size	-						
N=2	Sequenz	A	B					
	Match size	-	-					
N=3	Sequenz	A	B	A				
	Match size	**1**	-	-				
N=4	Sequenz	A	B	A	B			
	Match size	-	**2**	-	-			
N=5	Sequenz	A	B	A	B	B		
	Match size	-	**1**	-	**1**	-		
N=6	Sequenz	A	B	A	B	B	A	
	Match size	**1**	-	**2**	-	-	-	
N=7	Sequenz	A	**B**	**A**	**B**	B	A	B
	Match size	-	**2**	-	**3**	**1**	-	-

Patterns in Games

- Positioning Pattern
  - PONG: *Bot* vs. *Player*. Modellierung eines professionellen Gegners anhand von Heuristiken:
    - *Move back to center!*
    - *Move to where ball is expected!*
  - Einteilung des Spielfeldes in Blöcke
  - Abbildung der Blöcke auf Elemente eines Alphabets
  - Impact-Block bestimmt somit das nächste Element der generierten Sequenz





Patterns in Games

- Anticipation Pattern
  - Kampfspiel mit den Moves:
    - *kick*
    - *punch*
    - *block*
  - Player entwickelt bevorzugte Move-Kombinationen
  - Bot antizipiert die Moves und reagiert mit geeigneten Countermoves
- Analogon: „Schere, Stein, Papier“

Anwendung

Patterns in Games

- Tracker Pattern
  - Multi-Player Shooter: systematische Planung einer Route zum Aufsammeln von:
    - *Waffen, Munition*
    - *Medikits*
    - *etc.*
  - Gefahr hierbei: Hinterhalt oder Verfolgung, falls ein Bot zur Erkennung von Patterns fähig ist...
  - Einteilung der Spielfeldkarte in Aufenthaltsräume, in denen sich ein Player befinden kann
  - Abbildung der Aufenthaltsräume auf ein Alphabet und Generierung von Sequenzen

Beispiel: Diskrete Verteilung

- Geg.: Sequenz Ω über $A = \{a, b\}$

  Ω = „abababbabb“
- Häufigkeit (*frequence*) des Nachfolgers x von b in Ω sei $F_{succ}(\Omega, b, x)$

x	$F_{succ}(\Omega, b, x)$	$P(succ(\Omega) = x)$
a	3	0.75
b	1	0.25

N-Gram Definition

- Definition: *Sei A ein Alphabet und N eine natürliche Zahl. Dann ist ein N-Gram ein Wort der Länge N über A.*
- N-Grams erlauben Interpretation als
  - probabilistische, gerichtete, antizyklische Graphen
  - geordnete Mengen mit N Elementen
- N-Gram Analyse: Von der NSA patentiertes Verfahren zur Kontextsuche (1995)
  - Bsp.: *„Atombombe in Nordkorea“* in einer großen Anzahl von E-Mails

N-Gram Spezialfälle

- Bigram (*bzw. 2-Gram mit N = 2*)
  - besteht aus genau 2 Zeichen
  - z.B. Markov-Ketten
- Trigram (*bzw. 3-Gram mit N = 3*)
  - besteht aus genau 3 Zeichen
  - Geeignet zur Modellierung von:
    - Spracherkennung
    - Rechtschreibkorrektur
    - Information Retrieval

„The N-Gram language model is usually derived from large training texts that share the same language characteristics as expected input.“ [W3C01]

Beispiel: Rechtschreibkorrektur

- 3-Gram für Korrekturvorschläge
- Schreibfehler: „*wirk*“ statt „*work*“
- 3-Gram Zerlegungen („■“ Füllzeichen für Leersymbol)
- 3-Gram Zerlegung von ***string*** wird repräsentiert durch T(***string***)

T(„wirk“)	{ ■■**w**, ■**wi**, **wir**, **irk**, **rk**■, **k**■■ }
T(„work“)	{ ■■**w**, ■**wo**, **wor**, **ork**, **rk**■, **k**■■ }

Beispiel: Rechtschreibkorrektur

3-Gram Zerlegungen der Worte *wirk, work*

			w	**i**	**r**	**k**					**w**	**o**	**r**	**k**		
1	■	■	**w**						■	■	**w**					
2		■	w	i						■	w	o				
3			w	i	r						w	o	r			
4				i	r	k						o	r	k		
5					**r**	**k**	■						**r**	**k**	■	
6						**k**	■	■						**k**	■	■

Beispiel: Rechtschreibkorrektur

- Ähnlichkeitsmaß für Wörter: *Dice*-Koeffizient

$$d(a,b) = \frac{2|T(a) \cap T(b)|}{|T(a)| + |T(b)|}$$

$$|T("wirk") \cap T("work")| = 3$$

$$|T("wirk")| = |T("work")| = 6$$

$$d("wirk","work") = \frac{2 \cdot 3}{6+6} = 0.5$$

Motivation für N-Grams in Spielen

- Game world constraints
  - In einigen Spielen sind bestimmte Moves nur unter gewissen Bedingungen möglich
- Player styles
  - In Flugsimulatoren oder Kampfspielen entwickeln Spieler eigene Techniken und Sequenzen
- Bonuses
  - Spieler erhalten Extras für bestimmte Kombinationen von Spielzügen
- Game controls
  - Button Konfigurationen machen bestimmte Move-Sequenzen leichter und/oder schneller ausführbar

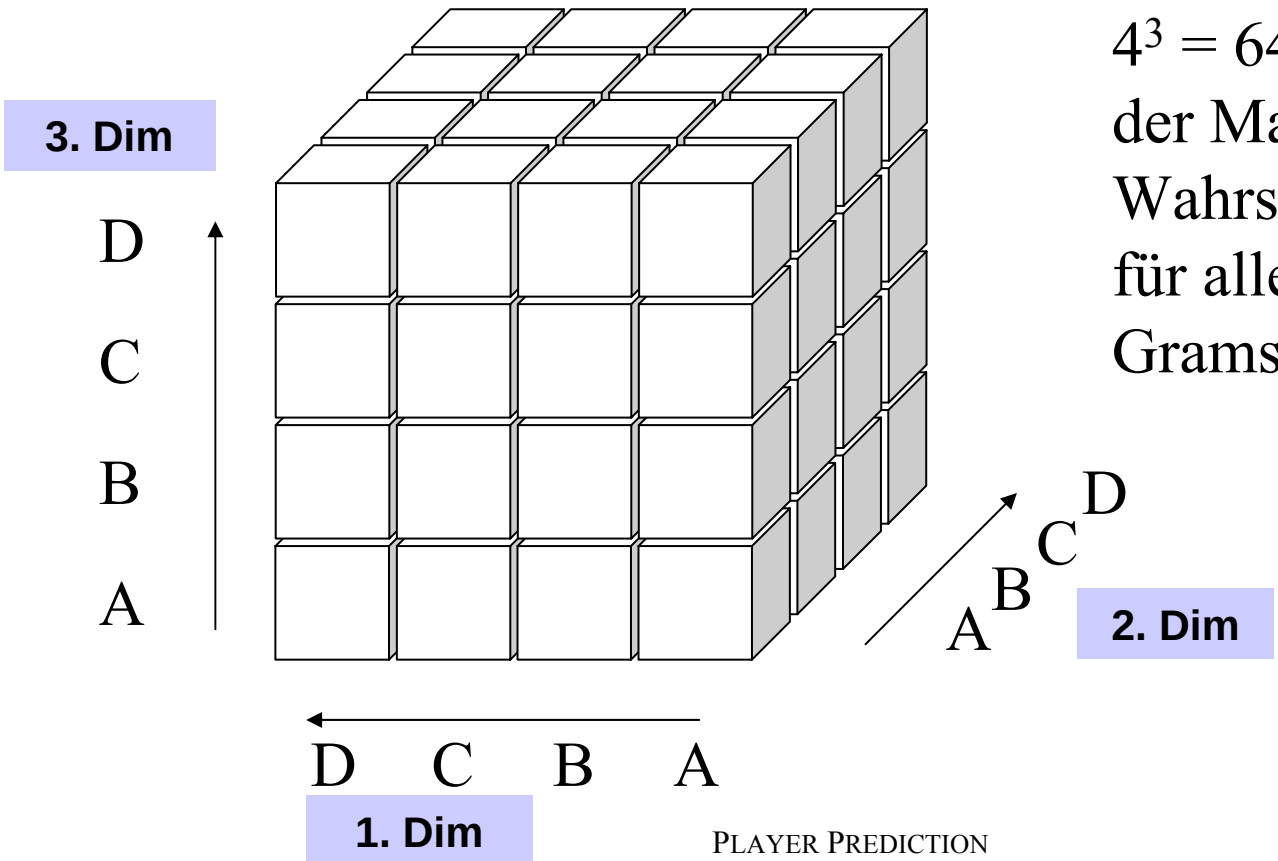
Motivation für N-Grams in Spielen

- N-Grams können diese unregelmäßigen Verteilungen analysieren
  - Vorhersage des Player Verhaltens möglich
  - Gegenangriff / Verteidigung planbar
- Datenstruktur: N-dimensionales Array
  - Speicherbedarf effizient für 3-Gram
- Bsp.: Kampfspiel mit 20 mögl. Moves
  - 3-Gram Array enthält 8000 (20^3) Einträge für die Wahrscheinlichkeiten aller möglichen Moves
  - Mit 32-bit Worten passt die Datenstruktur in 32K!

Beispiel: 3-Gram „Training“

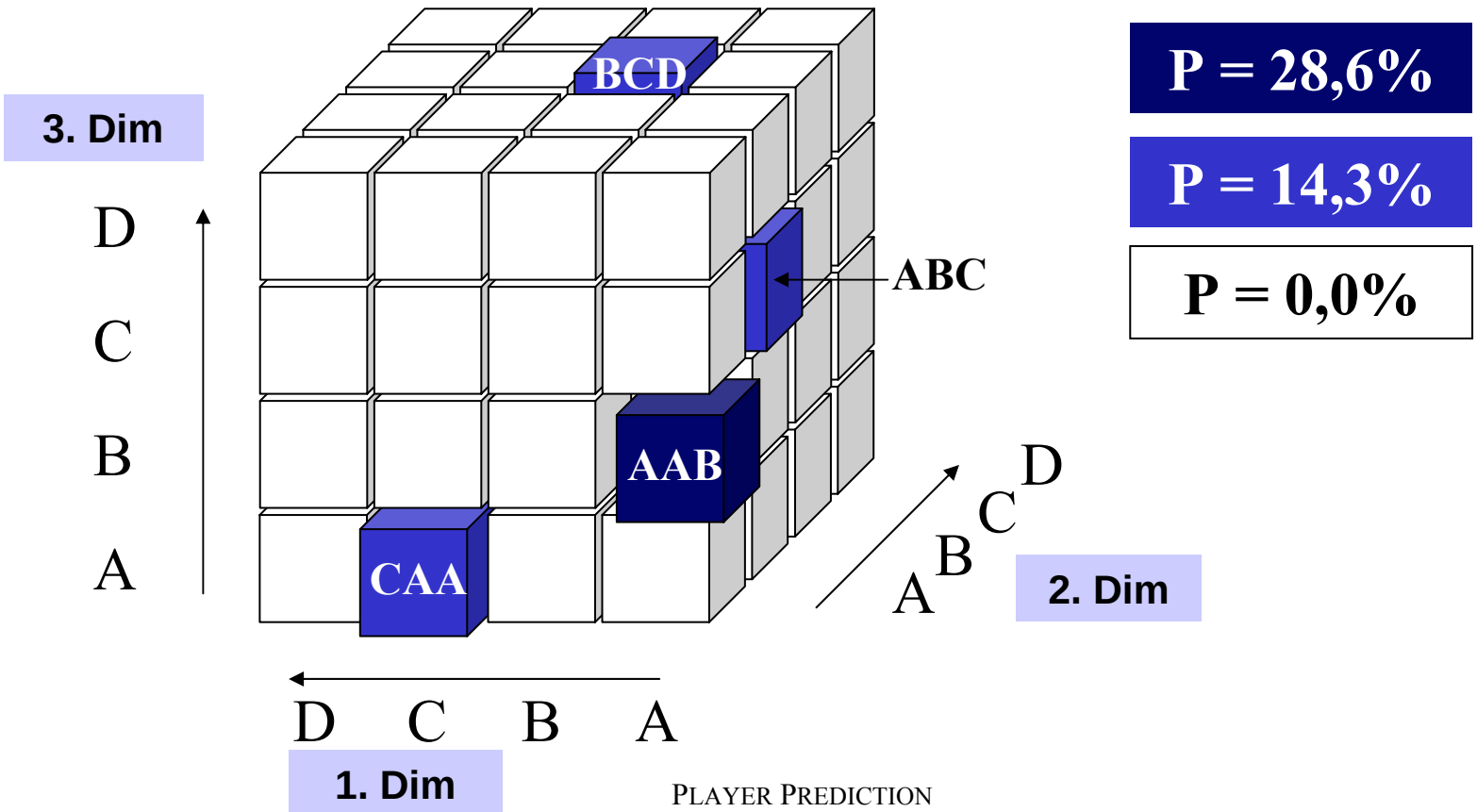
- Geg.: Menge möglicher *Moves*
  $M = \{A, B, C, D\}$
- Beobachtete Player-Sequenz:
  Σ_{Player} = „**AABCDCAAB**“
- 3-Gram Inkrementierung von Σ_{Player} :
  AAB, ABC, BCD, CDC, DCA, CAA, AAB
- Wahrscheinlichkeiten: 2/7 = 28,6% (AAB), 1/7 =14,3% (ABC, BCD,...) und 0 für Rest.

Illustration: 3-Gram Würfel



$4^3 = 64$ Einträge in der Matrix enthalten Wahrscheinlichkeiten für alle möglichen 3-Grams

Illustration: 3-Gram Würfel



P = 28,6%

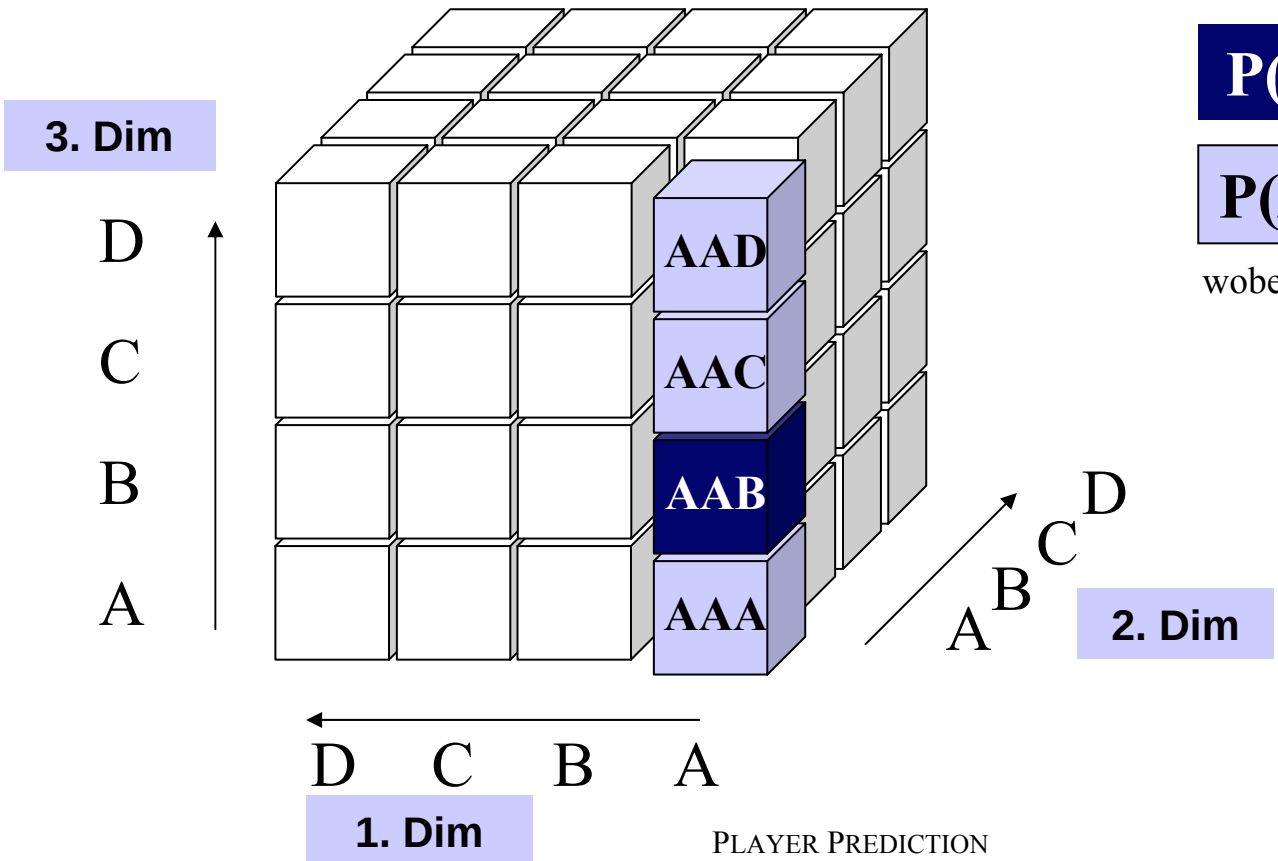
P = 14,3%

P = 0,0%

Beispiel: 3-Gram „Training“

- Relevantes Problem in Spielen:
  Wahrscheinlichkeit von Move Z, nachdem die Moves X und Y ausgeführt wurden
- Lösung: Speichern und Analysieren von bedingten Wahrscheinlichkeiten
- Beobachtung: Im vorigen Beispiel folgte auf AA **immer** B
- => **P(B | AA) = 1,**
  P(A | AA) = 0, P(C | AA) = 0, P(D | AA) = 0
- Güte dieser Approximation wächst mit Anzahl der Spiele

Illustration: 3-Gram Würfel

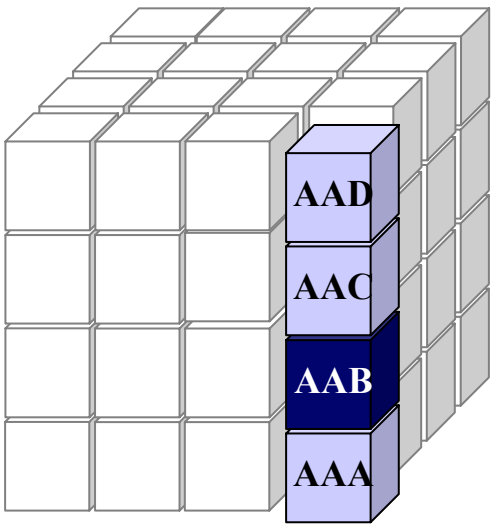


P(B|AA) = 100 %

P(*X*|AA) = 0%

wobei *X* aus {A,C,D}

Suche im 3-Gram: Indizierung

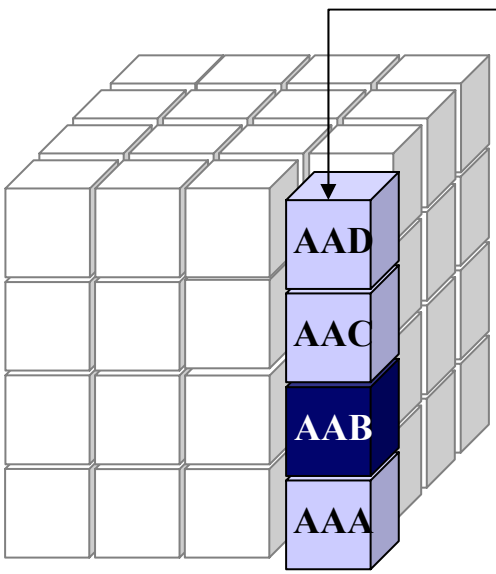


Selektionsmöglichkeiten:

(1) *deterministisch* – Auswahl des Eintrages mit der höchsten Wahrscheinlichkeit

(2) *probabilistisch* - Auswahl anhand einer Zufallszahl über die kumulierte Wahrscheinlichkeitsverteilung

Suche im 3-Gram: Indizierung



Pseudocode:

```
for (int i=0; i < numLegalMoves; i++)
{
     lookAt(Trigram[twoMovesAgo][lastMove][i]);
}
```

Selektionsmöglichkeiten:

(1) *deterministisch* – Auswahl des Eintrages mit der höchsten Wahrscheinlichkeit

(2) *probabilistisch* - Auswahl anhand einer Zufallszahl über die kumulierte Wahrscheinlichkeitsverteilung

Quellen

[Hut02] Hutchens, Jason; Barnes, Jonty: „Practical Natural Language Learning“, *AI Game Programming Wisdom, pp. 602-614,* [Charles River Media] Hingham, Mass., 2002.

[Jur00] Jurafsky, D. and Martin, J. H., „N-Grams,“ *Speech and Language Processing*, Prentice Hall, 2000.

[Lar02] Laramée, François Dominic: „Using N-Gram Statistical Models to Predict Player Behavior“, *AI Game Programming Wisdom, pp. 596-601,* [Charles River Media] Hingham, Mass., 2002.

[Mom02] Mommersteeg, Fri: „Pattern Recognition with Sequential Prediction“, *AI Game Programming Wisdom, pp. 586-595,* [Charles River Media] Hingham, Mass., 2002.

[Wit91] Witten, I. H. and Bell, T. C., „The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression,“ *IEEE Transactions on Information Theory*, 37 (4), pp. 1085-1094, 1991.

[W3C01] W3C: „Stochastic Language Models (N-Grams) Specifications“, *W3C Working Draft 3 January 2001*. http://www.w3.org/TR/ngram-spec

Fragen zu Player Prediction?

Vielen Dank für die Aufmerksamkeit.