Automatic Learning of Language Model Structure

Kevin Duh and Katrin Kirchhoff
{duh,katrin}@ee.washington.edu
University of Washington, USA

Introduction

- **Motivation:**
  - Factored Language Models have been applied to various tasks with good results, but specifying model parameters is tedious.
  - Desire an automatic method for finding model parameters.
- **Approach:**
  - View the problem as a structure learning / model selection problem
  - Develop a Genetic Algorithm solution
- **Result:**
  - A structure learning algorithm that finds good model parameters in a data-driven fashion
  - Perplexity reductions in Arabic and Turkish

Introduction

- **Motivation:**
  - Factored Language Models have been applied to various tasks with good results, but specifying model parameters is tedious.
  - Desire an automatic method for finding model parameters.
- **Approach:**
  - View the problem as a structure learning / model selection problem
  - Develop a Genetic Algorithm solution
- **Result:**
  - A structure learning algorithm that finds good model parameters in a data-driven fashion
  - Perplexity reductions in Arabic and Turkish
- **Our Goal in this Talk:**
  - Show the effectiveness and usability of Factored Language Model combined with Structure Learning
  - Encourage researchers to try it for their own tasks

Outline

- Factored Language Models
  - Why Use Factors?
  - Factored Word Representation
  - Backoff Graph
- Structure Learning for Factored Language Models
- Experiments and Results

Word-based Language Models

- Standard word-based language models

$$p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_1, \ldots, w_{t-1})$$

$$\approx \prod_{t=1}^{T} p(w_t \mid w_{t-1})$$

- How to get robust n-gram estimates (e.g. $p(w_t \mid w_{t-1})$)?
  - Smoothing
    - E.g. Kneser-Ney, Good-Turing
  - Class-based language models

$$p(w_t \mid w_{t-1}) \approx p(w_t \mid C(w_t))\, p(C(w_t) \mid C(w_{t-1}))$$

Limitation of Word-based Language Models

Limitation of Word-based Language Models

- **<u>Words are inseparable whole units</u>**.
  - E.g. “book” and “books” are distinct vocabulary units

Limitation of Word-based Language Models

- **Words are inseparable whole units**.
  - E.g. “book” and “books” are distinct vocabulary units
- Especially problematic in **morphologically-rich languages**:
  - Arabic, Finnish, Russian, Turkish
  - Many unseen word contexts and high perplexity

Limitation of Word-based Language Models

- **<u>Words are inseparable whole units</u>**.
  - E.g. “book” and “books” are distinct vocabulary units
- Especially problematic in **<u>morphologically-rich languages</u>**:
  - Arabic, Finnish, Russian, Turkish
  - Many unseen word contexts and high perplexity

Arabic k-t-b	
Kitaab	A book
Kitaab-iy	My book
Kitaabu-hum	Their book
Kutub	Books
Kataaba	To write

Solution: Word as Factors

- Decompose words into “factors” (e.g. morphemes)
- Build language model over factors: P(w|factors)

Solution: Word as Factors

- Decompose words into “factors” (e.g. morphemes)
- Build language model over factors: P(w|factors)
- Previous approach:
  - Linear sequence of morphemes [e.g. Geutner, 1995]
  - Models relations between affixes/stems
  - What we really want is a model that predicts words, but uses affixes/stems for robust estimation

Solution: Word as Factors

- Decompose words into “factors” (e.g. morphemes)
- Build language model over factors: P(w|factors)
- Previous approach:
  - Linear sequence of morphemes [e.g. Geutner, 1995]
  - Models relations between affixes/stems
  - What we really want is a model that predicts words, but uses affixes/stems for robust estimation
- Our approach: Factored Language Models
  - [Kirchhoff et. al., 2002], [Bilmes & Kirchhoff, 2003]
  - Parallel sequence of “factors”
  - Novel backoff procedure

Factored Word Representations

- $w \equiv \{f^1, f^2, \ldots, f^K\} \equiv f^{1:K}$
- $p(w_1, w_2, \ldots, w_T) \equiv p(f_1^{1:K}, f_2^{1:K}, \ldots, f_T^{1:K})$

$$\approx \prod_{t=1}^{T} p(f_t^{1:K} \mid f_{t-1}^{1:K}, f_{t-2}^{1:K})$$

Factored Word Representations

- $w \equiv \{f^1, f^2, \ldots, f^K\} \equiv f^{1:K}$
- $p(w_1, w_2, \ldots, w_T) \equiv p(f_1^{1:K}, f_2^{1:K}, \ldots, f_T^{1:K})$

$$\approx \prod_{t=1}^{T} p(f_t^{1:K} \mid f_{t-1}^{1:K}, f_{t-2}^{1:K})$$

- Advantageous in backoff
  - Words may not be observed, but factors are
  - Simultaneous class assignment

Factored Word Representations

- $w \equiv \{f^1, f^2, \dots, f^K\} \equiv f^{1:K}$
- $p(w_1, w_2, \dots, w_T) \equiv p(f_1^{1:K}, f_2^{1:K}, \dots, f_T^{1:K})$

$$\approx \prod_{t=1}^{T} p(f_t^{1:K} \mid f_{t-1}^{1:K}, f_{t-2}^{1:K})$$

- Advantageous in backoff
  - Words may not be observed, but factors are
  - Simultaneous class assignment

Word
↓

$$\begin{bmatrix} \text{word} \\ \text{stem} \\ \text{root} \\ \text{tag} \end{bmatrix}$$

Factored Word Representations

- $w \equiv \{f^1, f^2, \ldots, f^K\} \equiv f^{1:K}$
- $p(w_1, w_2, \ldots, w_T) \equiv p(f_1^{1:K}, f_2^{1:K}, \ldots, f_T^{1:K})$

$$\approx \prod_{t=1}^{T} p(f_t^{1:K} \mid f_{t-1}^{1:K}, f_{t-2}^{1:K})$$

- Advantageous in backoff
  - Words may not be observed, but factors are
  - Simultaneous class assignment

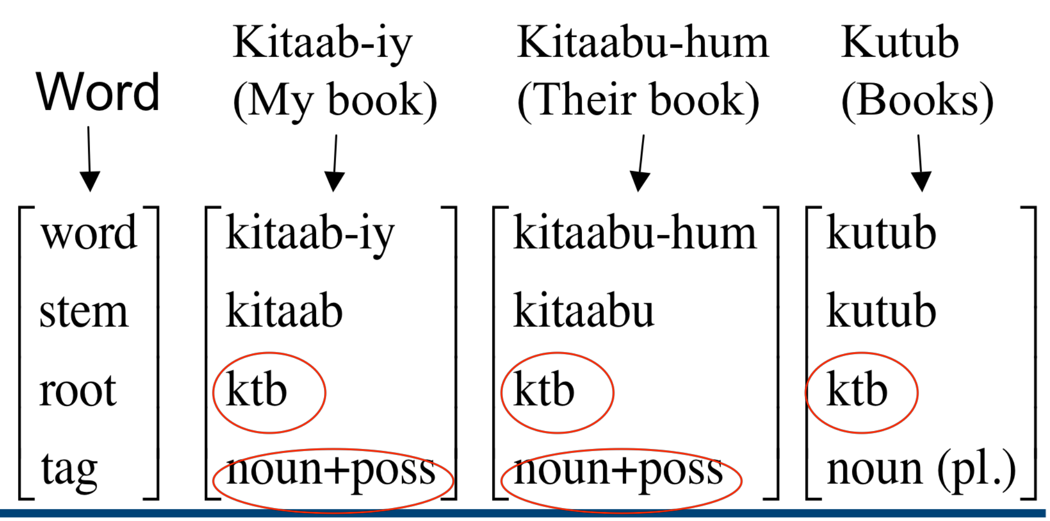
Word	Kitaab-iy (My book)	Kitaabu-hum (Their book)	Kutub (Books)
word	kitaab-iy	kitaabu-hum	kutub
stem	kitaab	kitaabu	kutub
root	ktb	ktb	ktb
tag	noun+poss	noun+poss	noun (pl.)

Factored Word Representations

- $w \equiv \{f^1, f^2, \ldots, f^K\} \equiv f^{1:K}$
- $p(w_1, w_2, \ldots, w_T) \equiv p(f_1^{1:K}, f_2^{1:K}, \ldots, f_T^{1:K})$

$$\approx \prod_{t=1}^{T} p(f_t^{1:K} \mid f_{t-1}^{1:K}, f_{t-2}^{1:K})$$

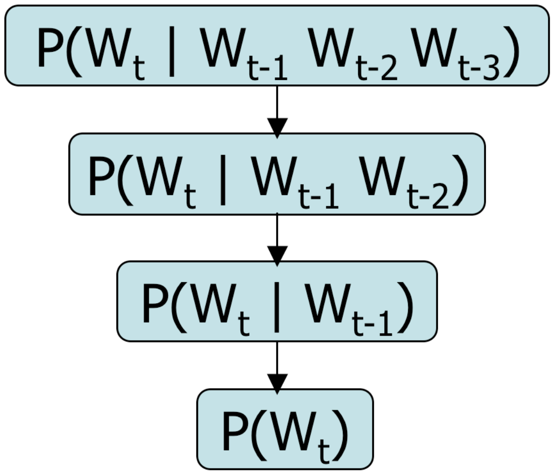
- Advantageous in backoff
  - Words may not be observed, but factors are
  - Simultaneous class assignment

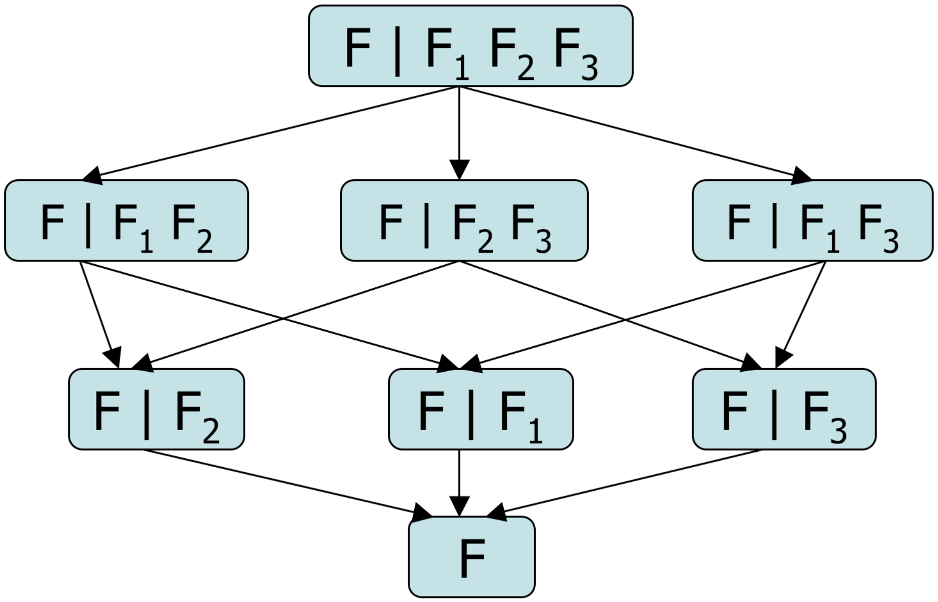


Language Model Backoff

- Idea: When n-gram count is low, use (n-1)-gram estimate

Word-based language model: Backoff most distant word	Factored language model: Backoff graph: multiple backoff paths possible





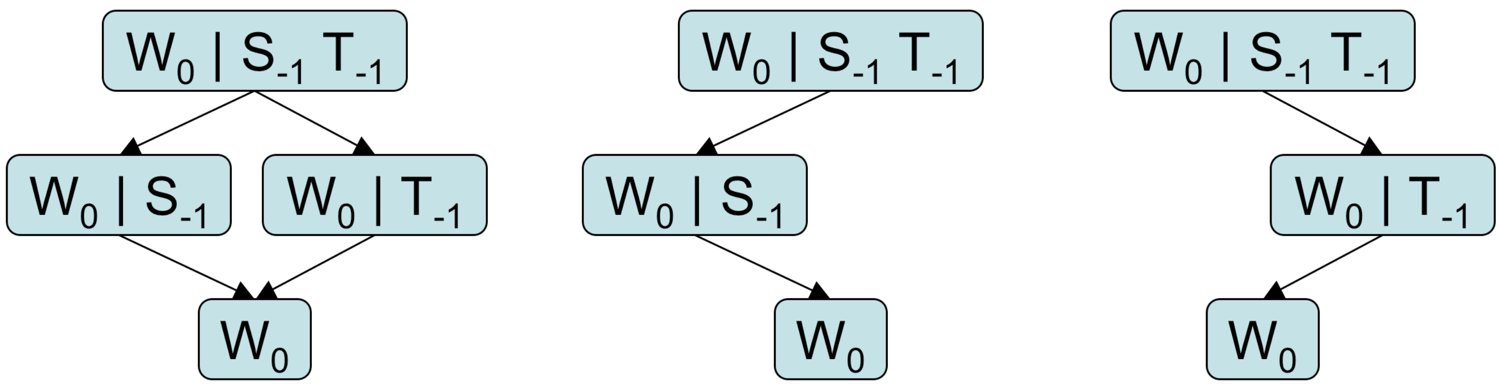
$$p_{bo}(F \mid F_1, F_2, F_3) = \begin{cases} \beta \cdot p_{ML}(F \mid F_1, F_2, F_3) & \text{if count} \geq \text{threshold} \\ \alpha \cdot g(F, F_1, F_2, F_3) & \text{else} \end{cases}$$

Outline

- Factored Language Models
- Structure Learning for Factored Language Models
  - What Parameters to Learn?
  - Genetic Algorithms Overview
  - Genetic Algorithms Applied to Structure Learning
- Experiments and Results

Parameters for Factored Language Models

1. Initial conditioning factors
   - E.g. If there are 4 available factors: $\{s_{-1}, s_{-2}, t_{-1}, t_{-2}\}$
     - Do we use all of them? $p(w_0 \mid s_{-1}, s_{-2}, t_{-1}, t_{-2})$
     - Or some subset of them? $p(w_0 \mid s_{-1}, t_{-1})$
2. Backoff graph
   - E.g. 3 backoff graphs are possible for 2 initial factors



The Need for Automatic Structure Learning

- Previously, parameters have been specified by hand.
- But search space is large:

1. For a total of *K* available factors….

   $\sum_{n=1}^{K} \binom{K}{n}$ possible subsets of factors

2. For a set of *M* factors (from 1)….

   $M\,!$ possible backoff graph configurations

The Need for Automatic Structure Learning

- Previously, parameters have been specified by hand.
- But search space is large:
  1. For a total of *K* available factors….

     $$\sum_{n=1}^{K} \binom{K}{n}$$ possible subsets of factors

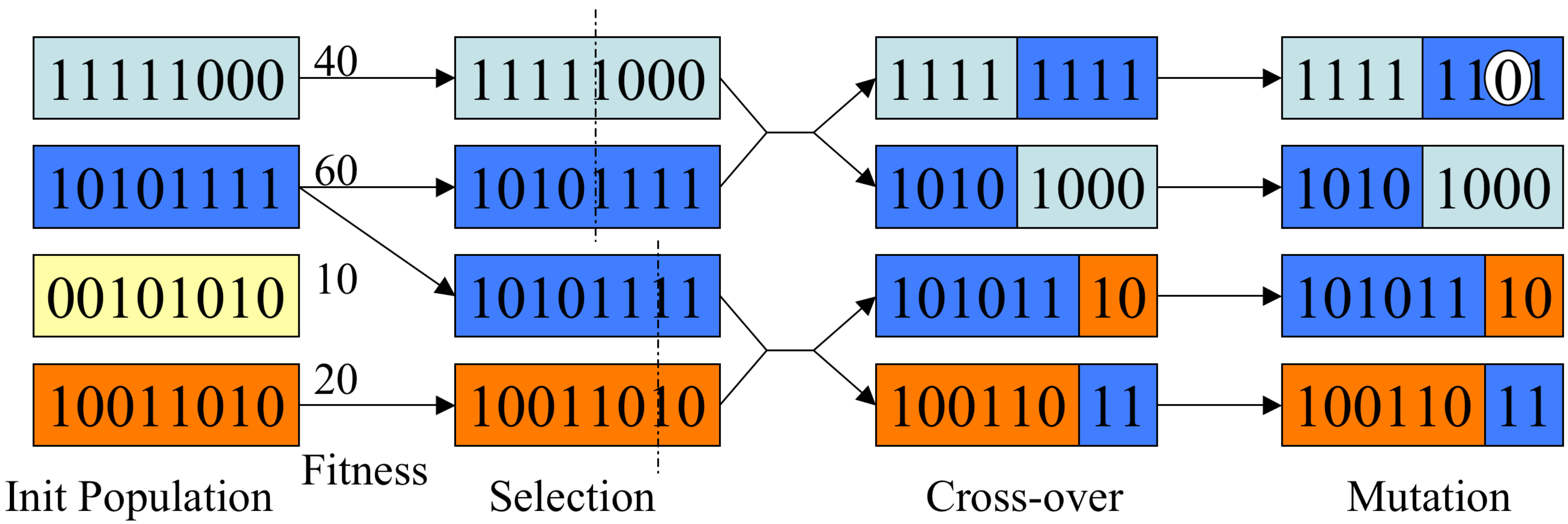
  2. For a set of *M* factors (from 1)….

     $$M\,!$$ possible backoff graph configurations

- Solution:
  - Genetic Algorithms!

Genetic Algorithms Overview

- General search/optimization technique inspired by evolution and genetics
  - Potential solutions are encoded as “genes”
  - “Evolve” genetic population using “fitness function”, etc.



Genetic Algorithms for Factored Language Models

- Each gene represents a particular model structure, in particular:
  - Initial Factors
  - Backoff Graph

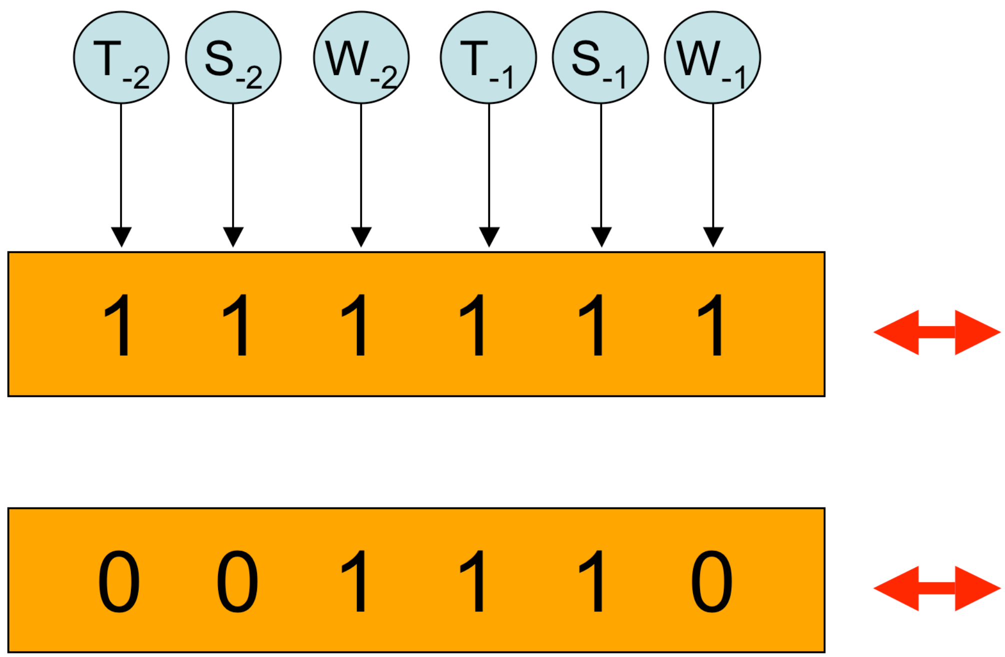
Initial Factors → 0 1 1 0 1 | 1 0 1 1 0 1 0 1 0 ← Backoff Graph

- Fitness function: Dev Set perplexity
- The hope: Genetic algorithm creates successive populations of genes with better model, lower perplexity

Gene for Initial Factors

- 0 or 1 indicates whether to use a factor

E.g. 6 available factors.
Which ones to use?



$p(w_0 \mid w_{-1}, w_{-2}, s_{-1}, s_{-2}, t_{-1}, t_{-2})$

$p(w_0 \mid w_{-2}, s_{-1}, t_{-1})$

Gene for Backoff Graph

- 0 or 1 indicates whether to activate a **<u>Graph-Grammar Production Rule</u>**
  - Rule indicates which factor to backoff

Rule1: $\{X1,X2,X3\} \rightarrow \{X1,X2\}$

Rule2: $\{X1,X2,X3\} \rightarrow \{X1,X3\}$

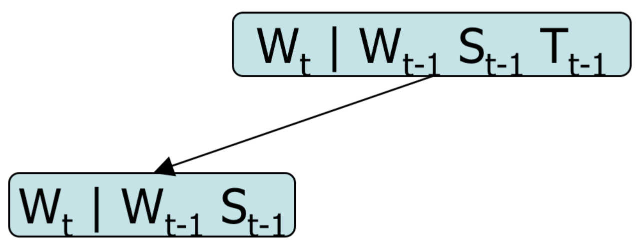
Rule3: $\{X1,X2,X3\} \rightarrow \{X2,X3\}$

$W_t \mid W_{t-1}\ S_{t-1}\ T_{t-1}$

Gene for Backoff Graph

- 0 or 1 indicates whether to activate a **Graph-Grammar Production Rule**
  - Rule indicates which factor to backoff

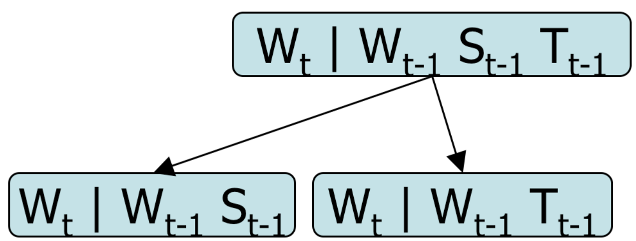
✔Rule1: {X1,X2,X3} → {X1,X2}
Rule2: {X1,X2,X3} → {X1,X3}
Rule3: {X1,X2,X3} → {X2,X3}



Gene for Backoff Graph

- 0 or 1 indicates whether to activate a **<u>Graph-Grammar Production Rule</u>**
  - Rule indicates which factor to backoff

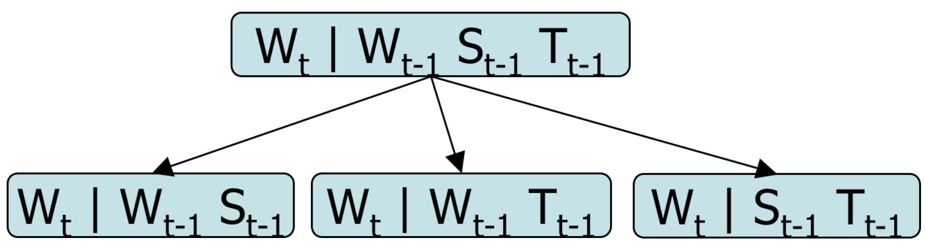
✔Rule1: {X1,X2,X3} → {X1,X2}
✔Rule2: {X1,X2,X3} → {X1,X3}
Rule3: {X1,X2,X3} → {X2,X3}



Gene for Backoff Graph

- 0 or 1 indicates whether to activate a **Graph-Grammar Production Rule**
  - Rule indicates which factor to backoff

✔Rule1: {X1,X2,X3} → {X1,X2}
✔Rule2: {X1,X2,X3} → {X1,X3}
✔Rule3: {X1,X2,X3} → {X2,X3}



Gene for Backoff Graph: Example

Production Rules:

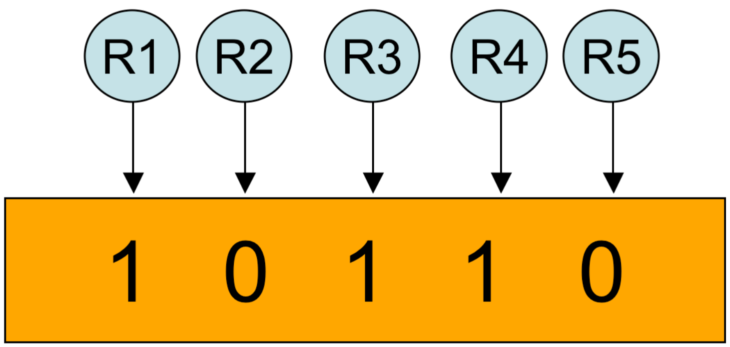
R1: {X1 X2 X3} → {X1 X2}
R2: {X1 X2 X3} → {X1 X3}
R3: {X1 X2 X3} → {X2 X3}
R4: {X1 X2} → {X1}
R5: {X1 X2} → {X2}

Backoff Graph

Gene for Backoff Graph: Example

<u>Production Rules:</u>

R1: {X1 X2 X3} → {X1 X2}
R2: {X1 X2 X3} → {X1 X3}
R3: {X1 X2 X3} → {X2 X3}
R4: {X1 X2} → {X1}
R5: {X1 X2} → {X2}



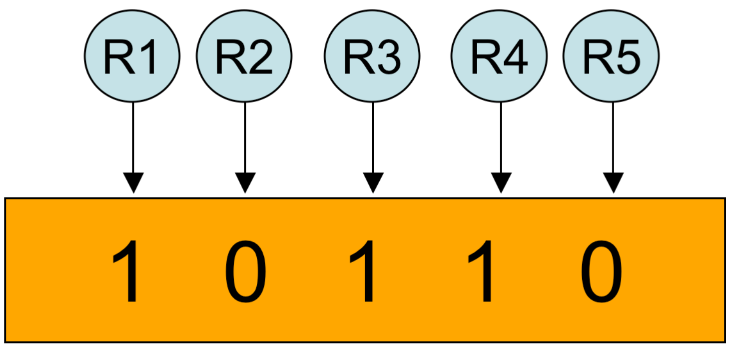
Gene

Backoff Graph

Gene for Backoff Graph: Example

Production Rules:

R1: {X1 X2 X3} → {X1 X2}
R2: {X1 X2 X3} → {X1 X3}
R3: {X1 X2 X3} → {X2 X3}
R4: {X1 X2} → {X1}
R5: {X1 X2} → {X2}



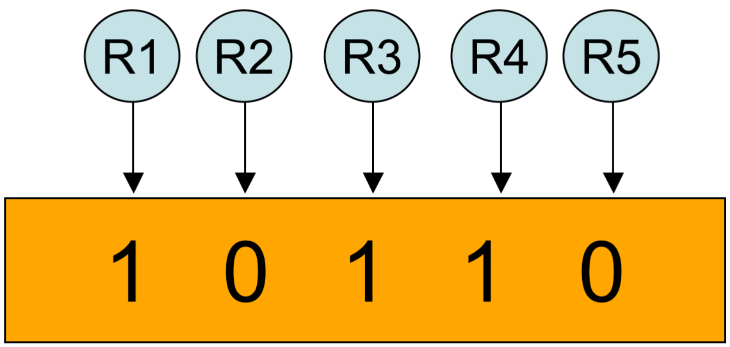
Gene

Backoff Graph

Gene for Backoff Graph: Example

Production Rules:

R1: {X1 X2 X3} → {X1 X2}
R2: {X1 X2 X3} → {X1 X3}
R3: {X1 X2 X3} → {X2 X3}
R4: {X1 X2} → {X1}
R5: {X1 X2} → {X2}



Gene

Backoff Graph

Gene for Backoff Graph: Example

Production Rules:

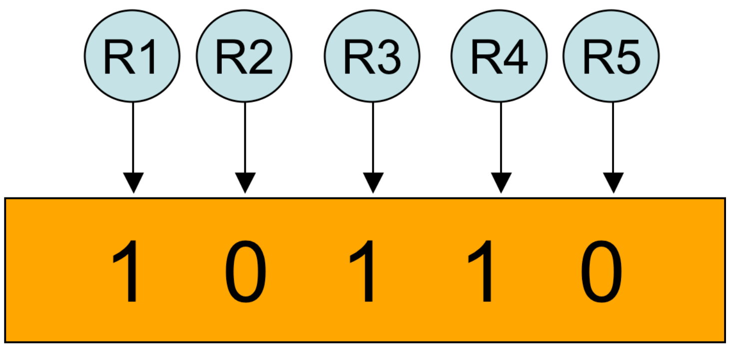
R1: {X1 X2 X3} → {X1 X2}

R2: {X1 X2 X3} → {X1 X3}

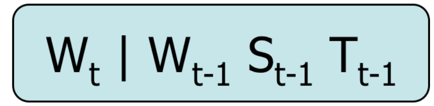
R3: {X1 X2 X3} → {X2 X3}

R4: {X1 X2} → {X1}

R5: {X1 X2} → {X2}



Gene



Backoff Graph

Gene for Backoff Graph: Example

<u>Production Rules:</u>

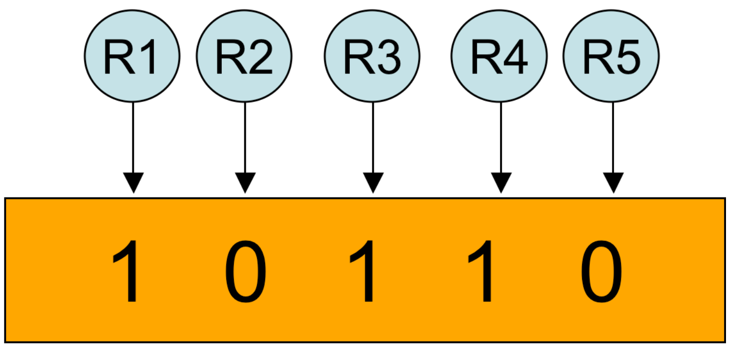
✔ R1: {X1 X2 X3} → {X1 X2}

R2: {X1 X2 X3} → {X1 X3}

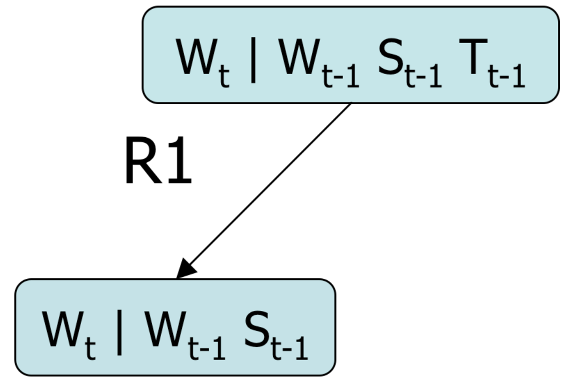
R3: {X1 X2 X3} → {X2 X3}

R4: {X1 X2} → {X1}

R5: {X1 X2} → {X2}



Gene

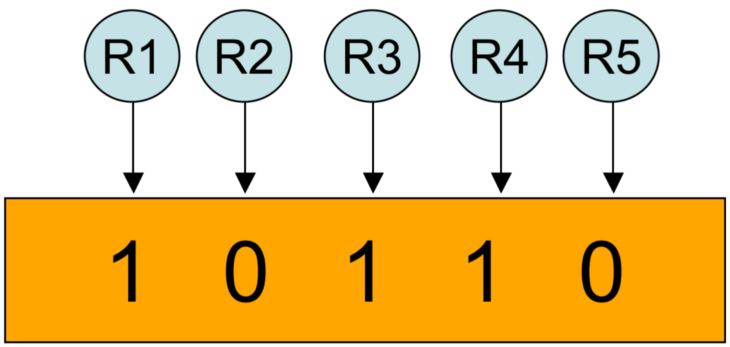


Backoff Graph

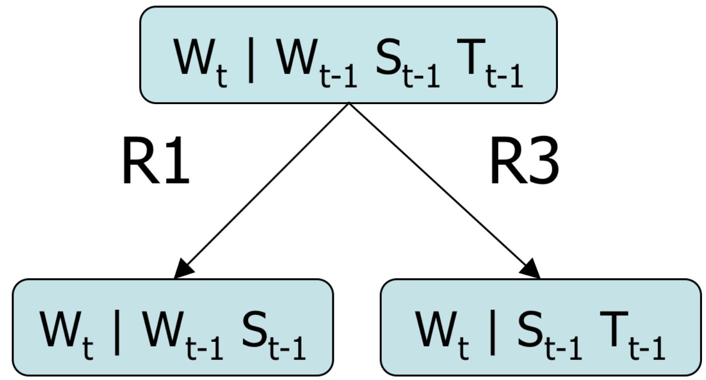
Gene for Backoff Graph: Example

<u>Production Rules:</u>

✔ R1: {X1 X2 X3} → {X1 X2}
R2: {X1 X2 X3} → {X1 X3}
✔ R3: {X1 X2 X3} → {X2 X3}
R4: {X1 X2} → {X1}
R5: {X1 X2} → {X2}



Gene



Backoff Graph

Gene for Backoff Graph: Example

Production Rules:

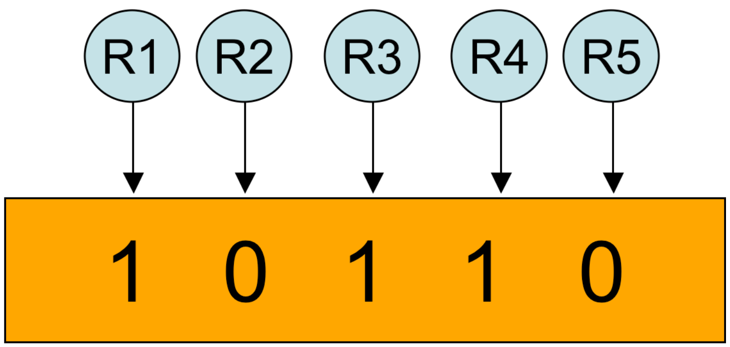
✔ R1: {X1 X2 X3} → {X1 X2}

R2: {X1 X2 X3} → {X1 X3}

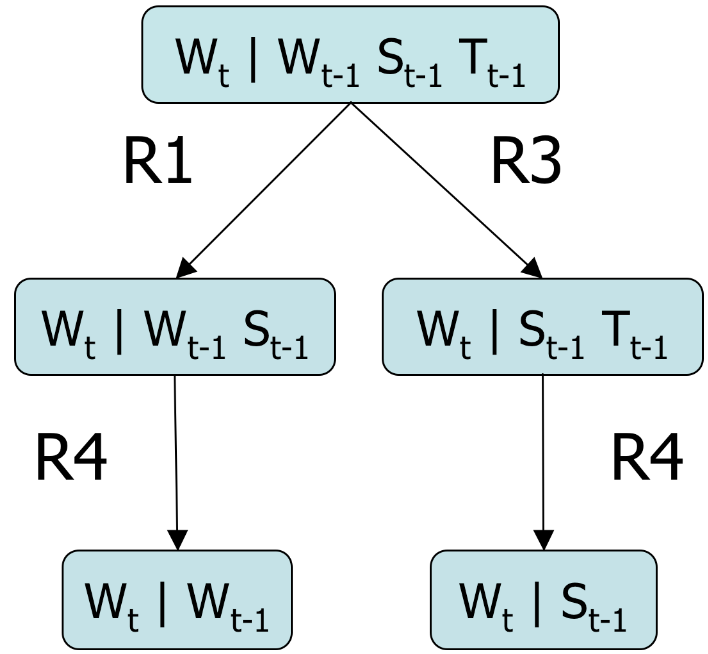
✔ R3: {X1 X2 X3} → {X2 X3}

✔ R4: {X1 X2} → {X1}

R5: {X1 X2} → {X2}



Gene

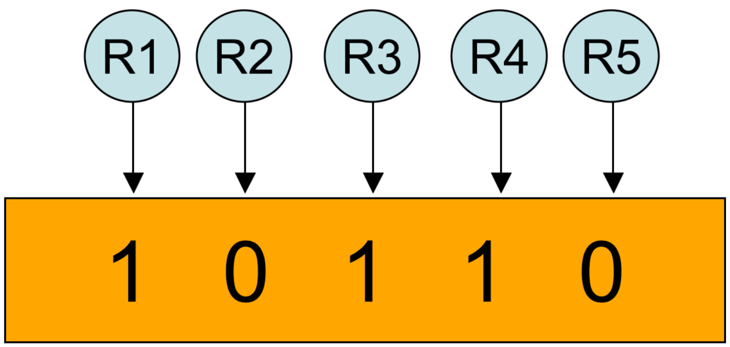


Backoff Graph

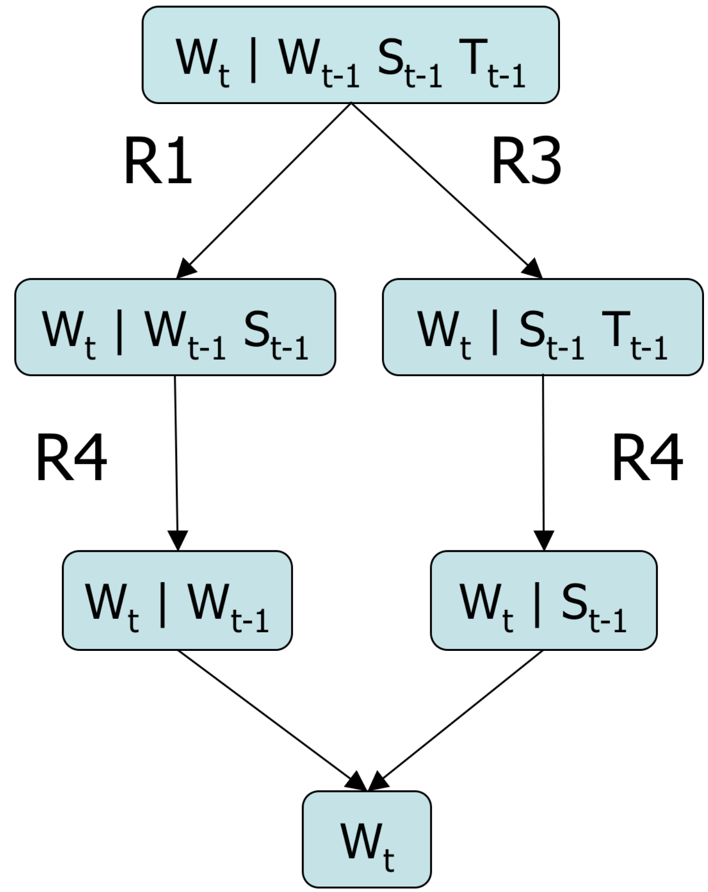
Gene for Backoff Graph: Example

<u>Production Rules:</u>

- ✔ R1: {X1 X2 X3} → {X1 X2}
- R2: {X1 X2 X3} → {X1 X3}
- ✔ R3: {X1 X2 X3} → {X2 X3}
- ✔ R4: {X1 X2} → {X1}
- R5: {X1 X2} → {X2}



Gene



Backoff Graph

Outline

- Factored Language Models
- Structure Learning for Factored Language Models
- Experiments and Results
  - Experimental Setup
  - Turkish Language Models
  - Arabic Language Models
  - Conclusion

Experimental Setup

- Main Question:
  - Can we find good factored language model structures automatically?
- 3 methods for getting factored language models:
  - Genetic Algorithms (50 genes/iteration, 10-50 iterations)
  - Search by hand
  - Random Search (500-2500 samples)
- Compare perplexities of different models

Turkish Language Models

- Data:
  - Newspaper text from web [Hakkani-Tür, 2000]
  - Train: 800K tokens / Dev: 100K / Test: 90K
  - Factors from morphological analyzer
    - Word, Root, Number, Case, POS for inflection groups

Eval Set perplexities

Ngram	Word LM	Hand FLM	Random FLM	Genetic FLM	Δppl(%)
2	609.8	558.7	525.5	**487.8**	**-7.2**
3	545.4	583.5	509.8	**452.7**	**-11.2**
4	543.9	559.8	574.6	**527.6**	**-5.8**

The best models used Word, POS, Case, Root factors, and various parallel backoff

Arabic Language Models

- Data:
  - Conversational Egyptian Arabic speech transcripts (LDC)
  - Train: 170K words / Dev: 23K / Test: 18K
  - Factors from morphological analyzer [Darwish, 2002]
    - Word, Morphological tag, Stem, Root, Pattern

Eval Set perplexities

Ngram	Word LM	Hand FLM	Random FLM	Genetic FLM	Δppl(%)
2	249.9	230.1	239.2	**223.6**	**-2.8**
3	285.4	217.1	224.3	**206.2**	**-5.0**

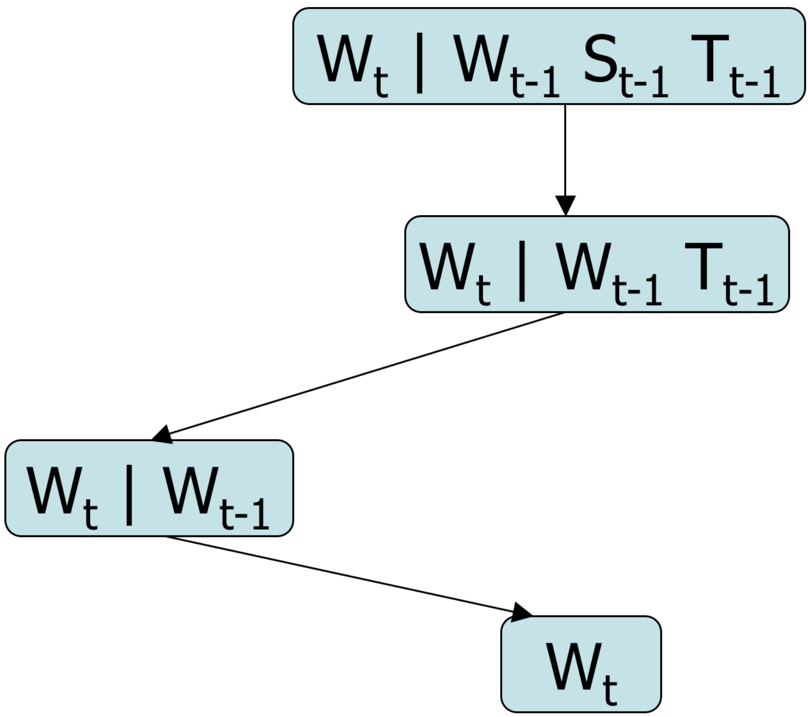
The best models used all available factors (Word, Stem, Root, Pattern, Morph), and various parallel backoffs

Conclusions

- Genetic algorithm finds superior models than hand-derived factored and word-based models.
- Improves perplexity by 5% (Arabic), 11% (Turkish)
- Enables fast development of factored language models in other tasks
  - Researchers can concentrate on developing good factors. Genetic algorithm automatically finds good structure.
- Promising Arabic speech recognition results
  - [Vergyri et. al., ICSLP 2004]

Choosing Backoff Paths: A Priori

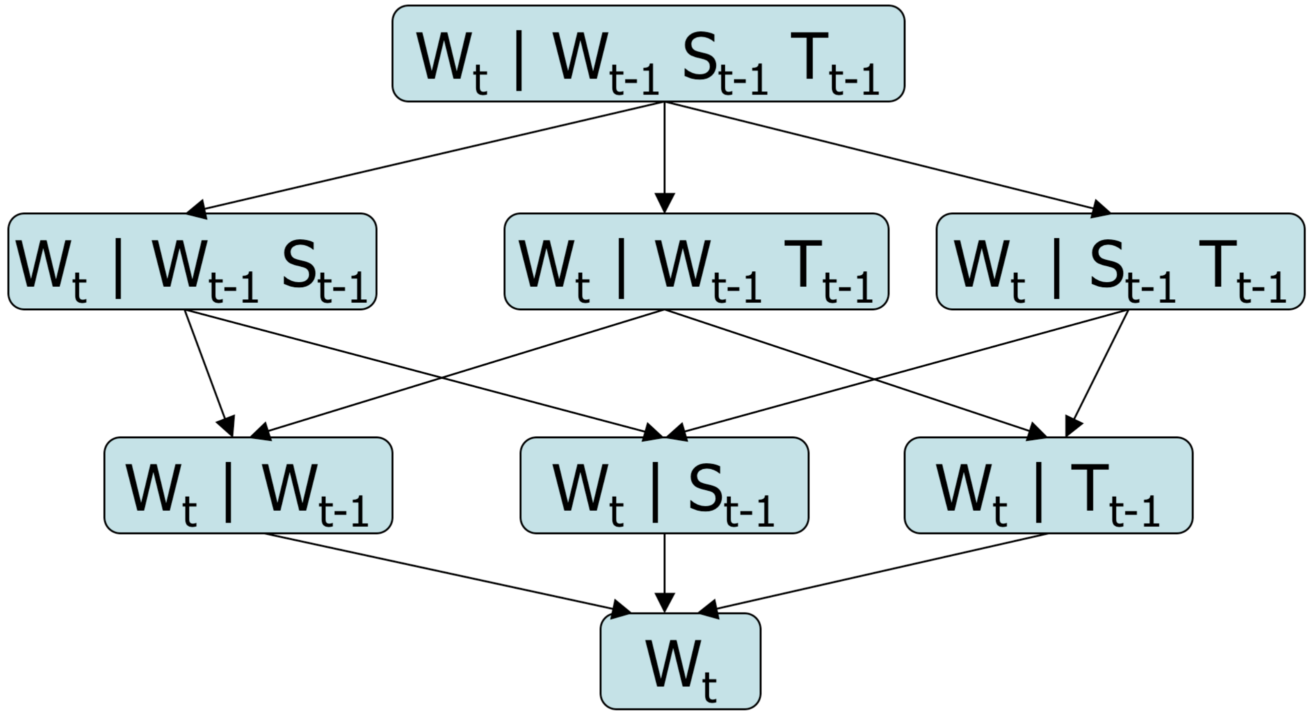
- Determine fixed backoff order *a priori* based on linguistic knowledge



<u>In following examples:</u>
- W = Word
- S = Stem
- T = Tag (POS)

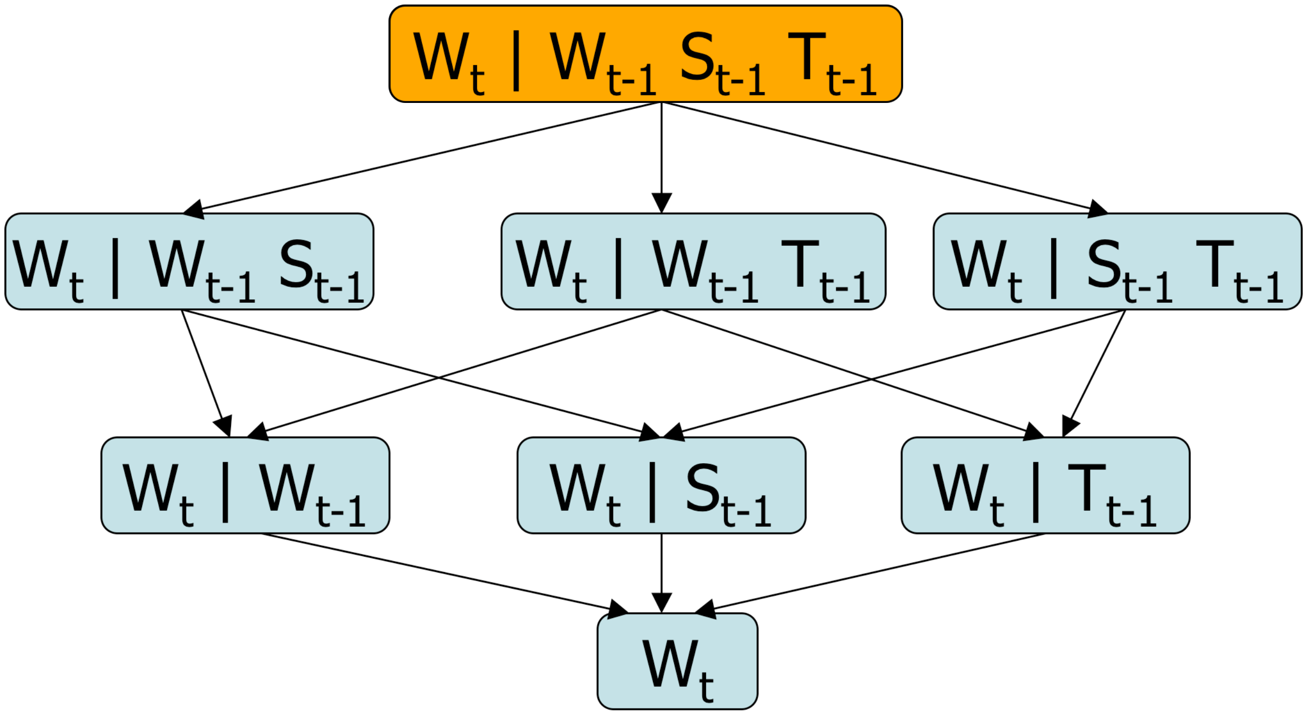
Choosing Backoff Paths: At Run-time

- Choose backoff path based based on statistical criteria during training



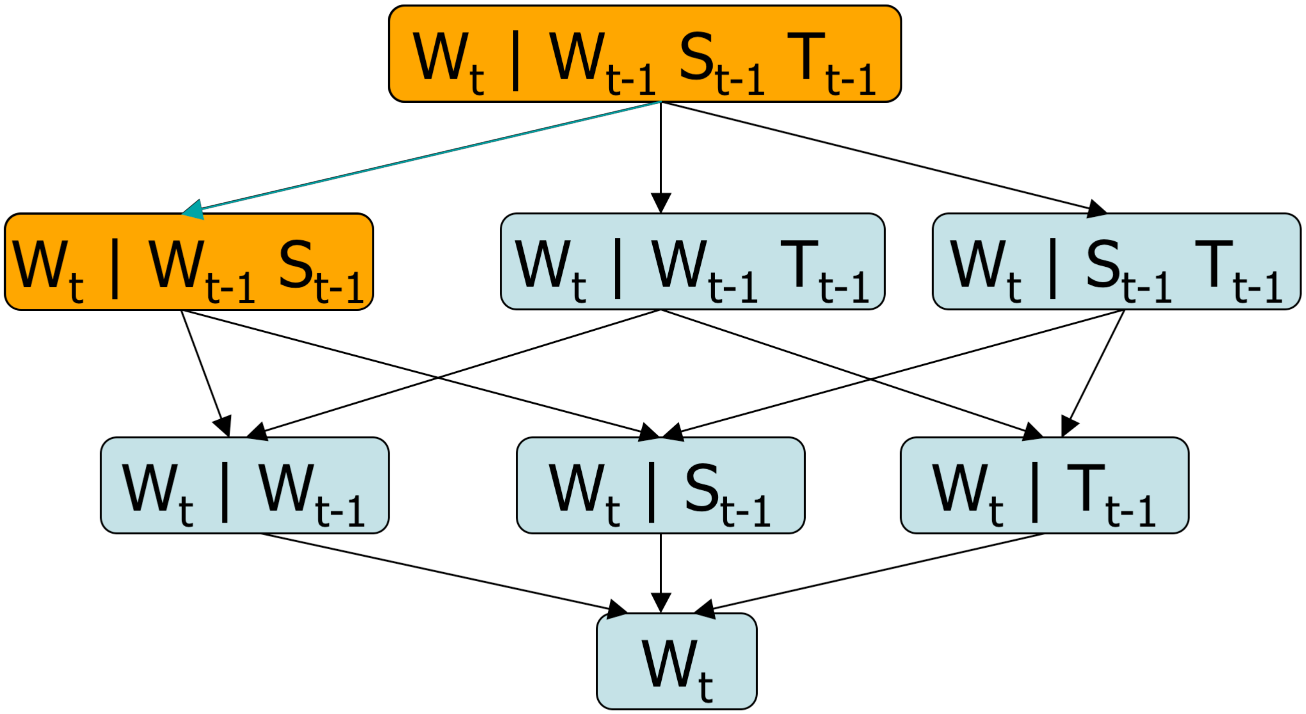
Choosing Backoff Paths: At Run-time

- Choose backoff path based based on statistical criteria during training



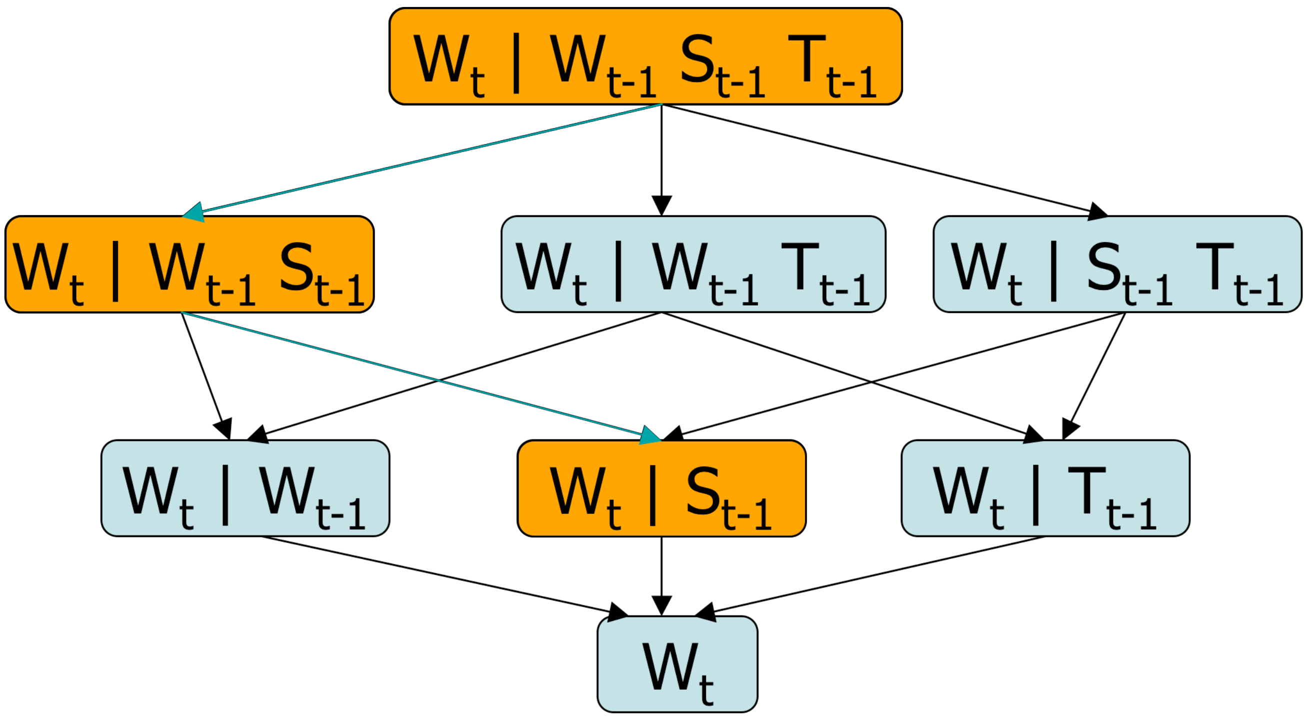
Choosing Backoff Paths: At Run-time

- Choose backoff path based based on statistical criteria during training



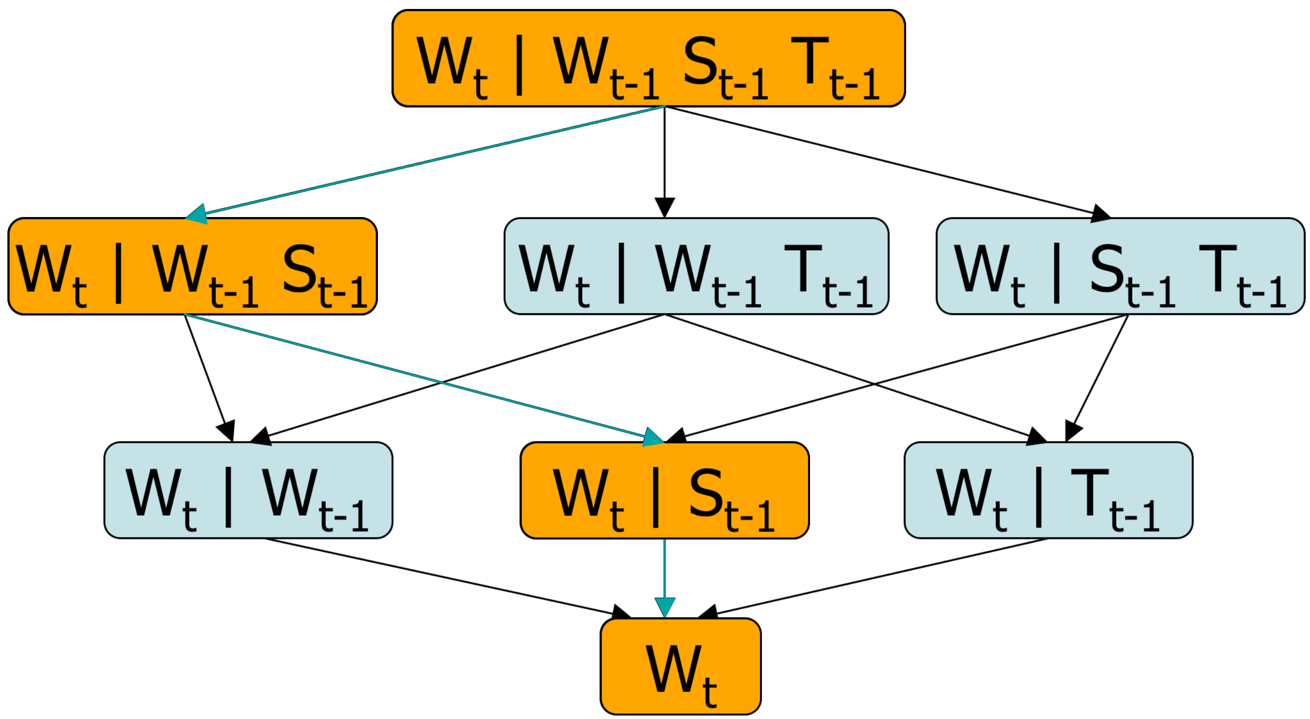
Choosing Backoff Paths: At Run-time

- Choose backoff path based based on statistical criteria during training



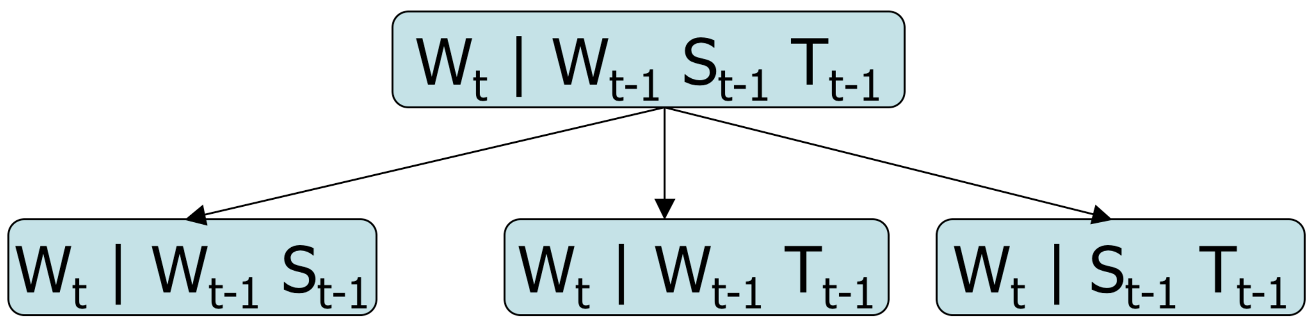
Choosing Backoff Paths: At Run-time

- Choose backoff path based based on statistical criteria during training



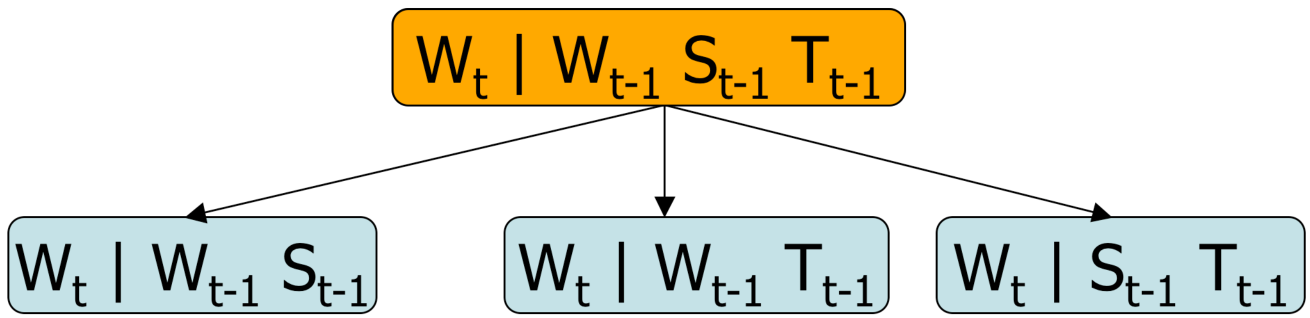
Choosing Backoff Paths: Generalized Parallel Backoff

- Choose <u>multiple paths</u> at run-time and combine probability estimates



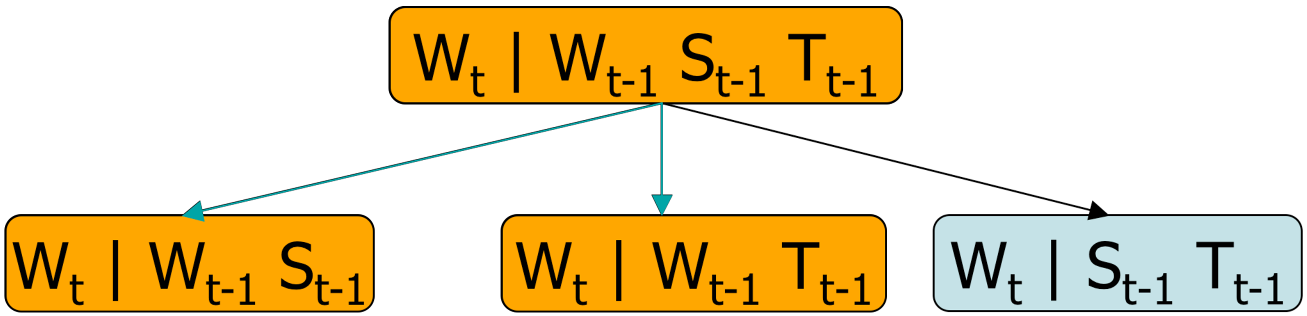
Choosing Backoff Paths: Generalized Parallel Backoff

- Choose <u>multiple paths</u> at run-time and combine probability estimates



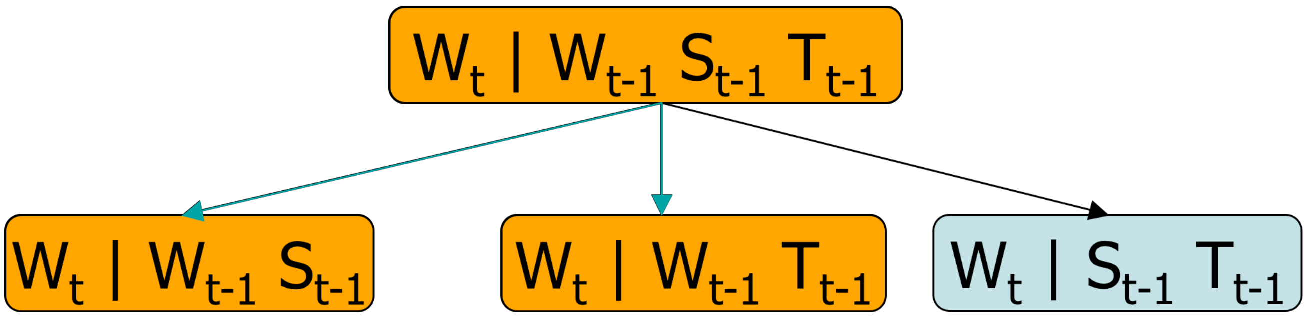
Choosing Backoff Paths: Generalized Parallel Backoff

- Choose <u>multiple paths</u> at run-time and combine probability estimates



Choosing Backoff Paths: Generalized Parallel Backoff

- Choose <u>multiple paths</u> at run-time and combine probability estimates



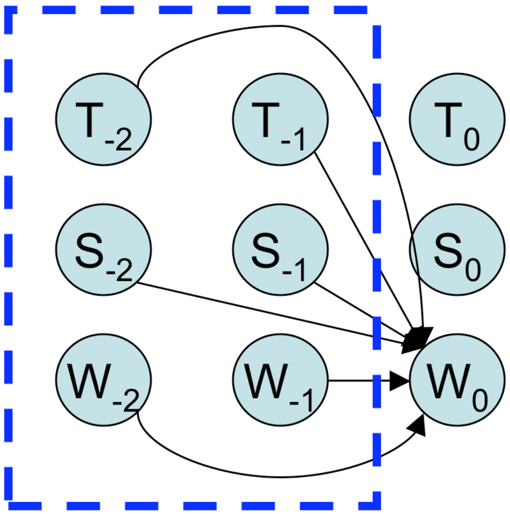
$$p_{bo}(w_t \mid w_{t-1}, s_{t-1}, t_{t-1}) = \begin{cases} d_c p_{ML}(w_t \mid w_{t-1}, s_{t-1}, t_{t-1}) & \text{if count} \geq \text{threshold} \\ \dfrac{\alpha}{2}[p_{bo}(w_t \mid w_{t-1}, s_{t-1}) + p_{bo}(w_t \mid w_{t-1}, t_{t-1})] & \text{else} \end{cases}$$

Joint Optimization of Initial Factors and Backoff Graph

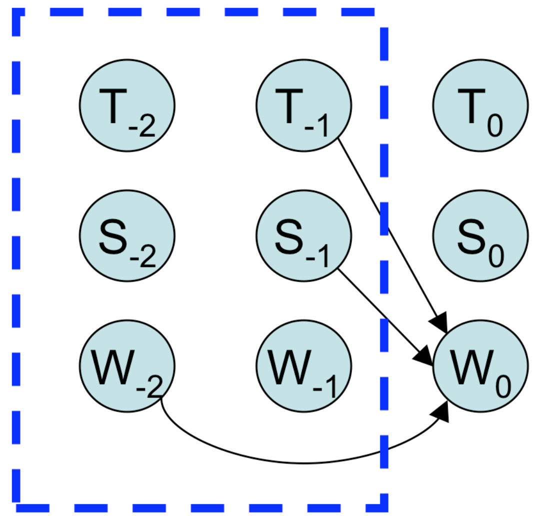
0 1 1 0 1	1 0 1 1 0 1 0 1 0 0 1 1 1 0
Initial Factors	Backoff Graph

- Genetic Algorithm Parameters:
  - Population: 30-50
  - Mutation probability: 0.01
  - Crossover probability: 0.90
  - Crossover method:1-point, 2-point, point-wise
  - Selection method: Roulette wheel, universal stochastic sampling
  - Elitist Strategy (best gene always survives)
- Fitness Function Convergence: $avg(1/ppl2) - avg(1/ppl1) \leq 10^{-5}$

Bayesian Networks Perspective



$$p(w_0 \mid w_{-1}, w_{-2}, s_{-1}, s_{-2}, t_{-1}, t_{-2})$$



$$p(w_0 \mid w_{-2}, s_{-1}, t_{-1})$$

Structure Learning = What dependencies (arrows) to add?

Turkish/Arabic Factored Words

- Turkish:
  - Yararlanmak (word)
  - Yarar (root)
  - NounInf-N:A3sg (part-of-speech),
  - Pnon (other)
  - Nom (case)
- Arabic
  - Il+dOr (word)
  - noun+masc-sg+article (morphological info)
  - dOr (stem)
  - dwr (root)
  - CCC (pattern)

Turkish: Comparison of Perplexities

Dev Set perplexities

Ngram	Word	Hand	Random	Genetic	Δppl(%)
2	593.8	555.0	556.4	**539.2**	**-2.9**
3	534.9	533.5	497.1	**444.5**	**-10.6**
4	534.8	549.7	566.5	**522.2**	**-5.0**

Eval Set perplexities

Ngram	Word	Hand	Random	Genetic	Δppl(%)
2	609.8	558.7	525.5	**487.8**	**-7.2**
3	545.4	583.5	509.8	**452.7**	**-11.2**
4	543.9	559.8	574.6	**527.6**	**-5.8**

The best models used Word, POS, Case, Root factors, and various parallel backoff

Arabic: Comparison of Perplexities

Dev Set perplexities

Ngram	Word	Hand	Random	Genetic	Δppl(%)
2	229.9	229.6	229.9	**222.9**	**-2.9**
3	229.3	226.1	230.3	**212.6**	**-6.0**

Eval Set perplexities

Ngram	Word	Hand	Random	Genetic	Δppl(%)
2	249.9	230.1	239.2	**223.6**	**-2.8**
3	285.4	217.1	224.3	**206.2**	**-5.0**

The best models used all available factors (Word, Stem, Root, Pattern, Morph), and various parallel backoffs