Processamento Estatístico da Linguagem Natural

Aula 5
Professora Bianca
(Sala 302 – Bloco E)
bianca@ic.uff.br
http://www.ic.uff.br/~bianca/peln/

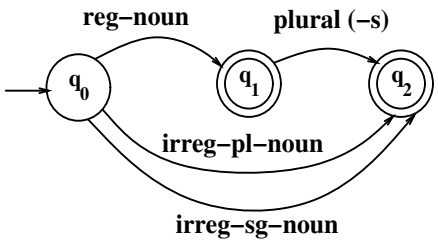
O que queremos

- Algo que faça automaticamente um mapeamento do tipo:
  - Cats `cat +N +PL`
  - Cat `cat +N +SG`
  - Cities `city +N +PL`
  - Merging `merge +V +Present-participle`
  - Caught `catch +V +past-participle`

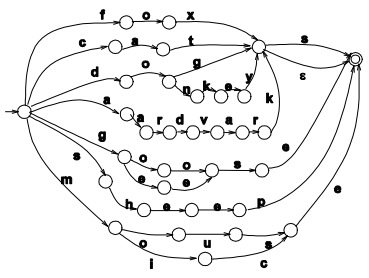
Parsing Morfológico

- Para construir um parser morfológico precisamos de:
  - **Léxico**: lista de raízes e afixos, mais informação básica sobre eles (se uma raiz é substantivo ou verbo, por exemplo).
  - **Morfotática**: o modelo de ordenação de morfemas que explica que classes de morfemas podem seguir outras classes de morfemas dentro da mesma palavra.
    - Por exemplo, o fato de o morfema de plural vem depois do morfema substantivo em inglês.
  - **Regras ortográficas**: usadas para modelar as mudanças que ocorrem em uma palavra quando dois morfemas se combinam.

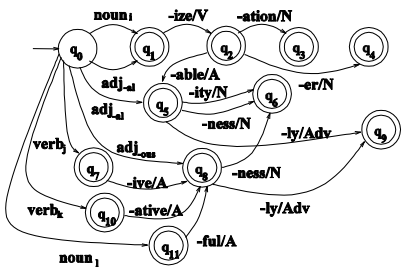
Regras Simples



Adicionando palavras



Regras Derivacionais



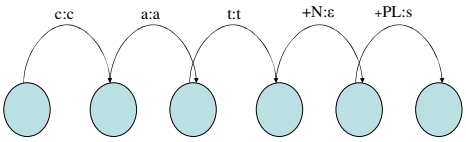
Reconhecimento vs. Parsing/Geração

- FSAs podem ser usados para reconhecimento.
- Porém reconhecimento não é suficiente.
  - Além de saber que a string pertence à linguagem queremos saber sua estrutura (**parsing**)
  - A partir da estrutura podemos querer gerar a string (**geração**)
- Exemplo
  - De "`cats`" para "`cat +N +PL`"
  - De "`cat +N +PL`" para "`cats`"

Transdutores de Estado Finito (FSTs)

- Um FST é um tipo de autômato finito que relaciona dois conjuntos de símbolos.
  - Tem duas fitas.
  - Lê uma string numa fita e escreve uma string na outra fita.
  - Cada arco do autômato tem dois símbolos (um para cada fita), separados por dois pontos.
  - Exemplo: em uma fita lê "`cats`", na outra escreve "`cat +N +PL`"

Exemplo



- c:c significa ler c numa fita e escrever c na outra.
- +N:ε significa ler o símbolo +N numa fita e não escrever nada na outra.
- +PL:s significa ler +PL e escrever um s

FSTs – Definição Formal

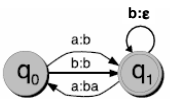
- Um conjunto de estados: Q
- Um alfabeto finito: Σ
- Um estado inicial $q_0 \in Q$
- Um conjunto F de estados finais $F \subseteq Q$
- Uma função de transição $\delta(q,w)$ que mapeia de $Q \times \Sigma^*$ para um conjunto de estados $\in Q$.
- Uma função de saída $\sigma(q,w)$ que dá o conjunto de possíveis saídas para cada estado e entrada.

Propriedades dos FSTs

- Enquanto FSAs são equivalentes às linguagens regulares, FSTs são equivalentes às relações regulares.
- FSTs são "fechados" em relação às operações de
  - União
  - Inversão (trocar as strings de entrada e saída)
    - Útil para converter de parsing para geração
  - Composição (Se T_1 é um transdutor de A para B e T_2 é um transdutor de B para C então $T_1 \circ T_2$ é um transdutor de A para C).

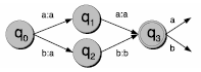
Transdutores Sequenciais

- Nem todo transdutor não-determinístico pode ser transformado em um transdutor determinístico.
- Transdutores sequenciais são um tipo de transdutor determinístico.
  - A saída pode ser ambígua, mas a entrada não.
  - Podem ter símbolos ε na saída, mas não na entrada.

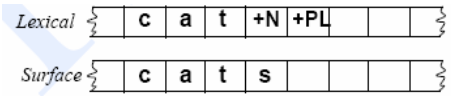


Transdutores Subsequenciais

- Geram uma saída adicional nos estados finais, que é concatenada à saída normal.
- Assim como os transdutores sequenciais são eficientes.
  - O tempo de processamento é proporcional ao tamanho da entrada.
- A generalização chamada de **transdutor p-subsequencial** permite p strings de saída para cada estado final.
  - Permite uma quantidade finita de ambiguidade mantendo a eficiência.
  - Serve para muitas tarefas de NLP, incluindo parsing morfológico.

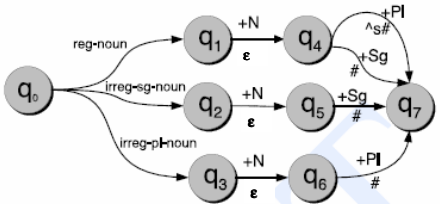


FSTs para Parsing Morfológico



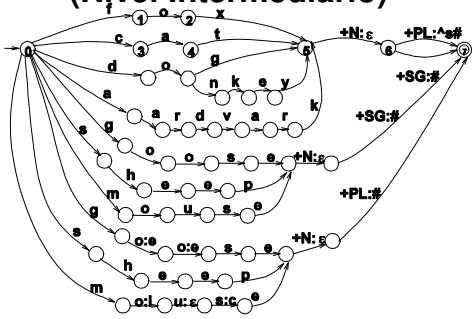
- Fita léxica: composta de símbolos do alfabeto Σ.
- Fita superficial: composta de símbolos do alfabeto Δ.
- Pares factíveis: pares de Σ×Δ que são aceitos pelo FST.
- Pares default: pares do tipo a:a (que podem ser escritos apenas como a)

FST para Inflexão de Número



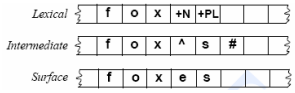
O símbolo ^ indica um **limite de morfema.**
O símbolo # indica um **limite de palavra**
Fitas com esses símbolos são chamadas de **fitas intermediárias.**

FST Expandido (Nível Intermediário)



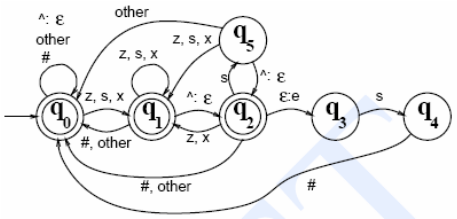
Regras Ortográficas

- Para passar do nível intermediário para o nível superficial usamos regras ortográficas.

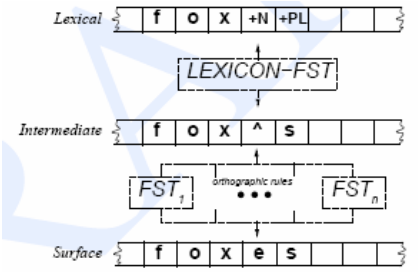


Name	Description of Rule	Example
Consonant doubling	1-letter consonant doubled before *-ing/-ed*	beg/begging
E deletion	Silent e dropped before *-ing* and *-ed*	make/making
E insertion	e added after *-s,-z,-x,-ch, -sh* before *-s*	watch/watches
Y replacement	*-y* changes to *-ie* before *-s*, *-i* before *-ed*	try/tries
K insertion	verbs ending with *vowel* + *-c* add *-k*	panic/panicked

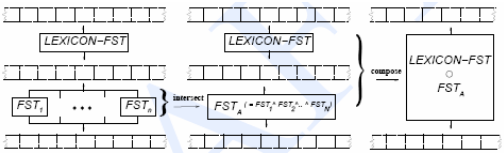
Transdutor para Regra Ortográfica de Inserção de E



FSTs em Cascata



Interseção e Composição



Stemming: O algoritmo de Porter

- Somente encontra a raiz (stem).
- Não utiliza um léxico.
- É baseado numa série de regras de re-escrita em cascata.
  - Exemplos: ATIONAL → ATE (e.g., relational → relate)
    ING → ϵ if stem contains vowel (e.g., motoring → motor)
  - Podem ser implementadas como um FST.
- É simples e eficiente, porém comete erros.

Errors of Commission		Errors of Omission	
organization	organ	European	Europe
doing	doe	analysis	analyzes
generalization	generic	matrices	matrix
numerical	numerous	noise	noisy
policy	police	sparse	sparsity

Correção Ortográfica

- Três tipos de problemas:
  - Detecção de palavras que não existem.
    - Detectar que a palavra *Graffe* não existe em inglês.
  - Correção de palavras isoladas.
    - Corrigir *graffe* para *giraffe*
  - Detecção e correção de erros dependentes de contexto.
    - Corrigir *three* pra *there* na frase "*I went three*".

Detecção de palavras que não existem

- Podemos usar um parser morfológico (FST) para reconhecer as palavras da língua.
  - Se o parser não reconhece ela não existe.
- Lida automaticamente com as inflexões e derivações morfológicas.
  - Não é necessário listar todas as palavras da língua explicitamente.

Correção de Erros

- Temos que encontrar a palavra que tem mais chance de ser a palavra original que foi escrita incorretamente.
- Precisamos de uma métrica para calcular a distância entre a palavra "superficial" (digitada) e cada possível palavra "fonte" (do dicionário).
  - Soluções atuais usam probabilidade e serão tratadas no capítulo 4.
    - Mas são baseadas no conceito de **distância mínima de edição** que será discutido nesse capítulo.

Distância Mínima de Edição

- É o número mínimo de edições (inserções, deleções e substituições) necessário para transformar uma string em outra.

```
INTE*NTION
||||||||||
*EXECUTION
d s s  i s
```

Alinhamento

Distância Mínima de Edição

- A distância mínima pode ser computada usando um algoritmo de **programação dinâmica**.
  - Classe de algoritmos introduzida por Bellman que resolve problemas combinando soluções de sub-problemas.