

Um Etiquetador Morfo-Sintático Baseado em Cadeias de Markov de Tamanho Variável

Defesa de Mestrado

Candidato: Fábio Natanael Kepler
Orientador: Prof. Dr. Marcelo Finger

Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

São Paulo, 12 de abril de 2005.

Roteiro

Introdução

Objetivo

Noções Teóricas

- Cadeias de Markov

- Cadeias de Markov de Tamanho Variável (VLMCs)

Construção dos Etiquetadores

Testes Realizados

Resultados Obtidos

Pós-Resultados

Conclusões

Trabalhos Futuros

Introdução

A Lingüística Computacional

- ▶ A Lingüística Computacional procura caracterizar e explicar os fenômenos lingüísticos através do computador
- ▶ Na impossibilidade de criação de um conjunto fixo de regras, buscamos padrões comuns no uso da linguagem
- ▶ Esta abordagem sugere que podemos aprender a estrutura da linguagem aplicando vários métodos a uma grande quantidade de usos da linguagem

Introdução

Nosso trabalho

- ▶ Vamos nos concentrar na *análise morfo-sintática*
 - ▶ É a tarefa intermediária de classificar as palavras de um texto em categorias gramaticais
 - ▶ De acordo com o contexto de cada palavra
- ▶ Poderemos alavancar o desenvolvimento das análises dos níveis mais altos da lingüística
 - ▶ Sintática
 - ▶ Semântica
 - ▶ Pragmática
- ▶ Útil para lidar com textos
 - ▶ Tradução
 - ▶ Sumarização
 - ▶ Extração de informação

Introdução

Nosso trabalho

- ▶ Vamos nos concentrar na *análise morfo-sintática*
 - ▶ É a tarefa intermediária de classificar as palavras de um texto em categorias gramaticais
 - ▶ De acordo com o contexto de cada palavra
- ▶ Poderemos alavancar o desenvolvimento das análises dos níveis mais altos da lingüística
 - ▶ Sintática
 - ▶ Semântica
 - ▶ Pragmática
- ▶ Útil para lidar com textos
 - ▶ Tradução
 - ▶ Sumarização
 - ▶ Extração de informação

Introdução

Nosso trabalho

- ▶ Vamos nos concentrar na *análise morfo-sintática*
 - ▶ É a tarefa intermediária de classificar as palavras de um texto em categorias gramaticais
 - ▶ De acordo com o contexto de cada palavra
- ▶ Poderemos alavancar o desenvolvimento das análises dos níveis mais altos da lingüística
 - ▶ Sintática
 - ▶ Semântica
 - ▶ Pragmática
- ▶ Útil para lidar com textos
 - ▶ Tradução
 - ▶ Sumarização
 - ▶ Extração de informação

Análise Morfo-Sintática

Problemas

- ▶ Realizar esta tarefa não é fácil
 - ▶ Muitas palavras possuem vários significados
- ▶ Precisamos tirar a ambigüidade
 - ▶ Determinando qual dos sentidos de uma palavra é invocado em um uso particular
- ▶ Temos que olhar para o contexto de cada palavra

Ambigüidade

Por exemplo, a palavra **como**

- ▶ Sabei de certo que já **como** a muito boas horas ;
- ▶ (...) além de nos explicar **como** se ensinava a Gramática Latina , (...)
- ▶ Mas , seja **como** for , estas não são razões (...)
- ▶ (...) e o **como** se pode ordenar uma Gramática (...)

Ambigüidade

Por exemplo, a palavra **como**

- ▶ Sabei/VB-I de/P certo/ADJ que/C já/ADV **como** a muito boas horas ;
- ▶ (...) além/ADV de/P nos/CL explicar/VB **como** se ensinava a Gramática Latina ,
(...)
- ▶ Mas/CONJ ,/, seja/SR-SP **como** for , estas não são razões (...)
- ▶ (...) e/CONJ o/D **como** se pode ordenar uma Gramática (...)

Ambigüidade

Por exemplo, a palavra **como**

- ▶ Sabei/VB-I de/P certo/ADJ que/C já/ADV **como**/VB-P a muito boas horas ;
- ▶ (...) além/ADV de/P nos/CL explicar/VB **como**/WADV se ensinava a Gramática Latina , (...)
- ▶ Mas/CONJ ,/, seja/SR-SP **como**/CONJS for , estas não são razões (...)
- ▶ (...) e/CONJ o/D **como**/ADJ se pode ordenar uma Gramática (...)

Ambigüidade

Por exemplo, a palavra **como**

- ▶ Sabei/VB-I de/P certo/ADJ que/C já/ADV **como**/VB-P a/P muito/Q boas/ADJ-F-P horas/N-P ;/.
- ▶ (...) além/ADV de/P nos/CL explicar/VB **como**/WADV se/SE ensinava/VB-D a/D-F Gramática/NPR Latina/ADJ-F ,/, (...)
- ▶ Mas/CONJ ,/, seja/SR-SP **como**/CONJS for/SR-SR ,/, estas/D-F-P não/NEG são/SR-P razões/N-P (...)
- ▶ (...) e/CONJ o/D **como**/ADJ se/SE pode/VB-P ordenar/VB uma/D-UM-F Gramática/NPR (...)

Etiquetador Morfo-Sintático

- ▶ Um programa de computador que procura tirar esta ambigüidade é chamado de **etiquetador morfo-sintático**
- ▶ Ele tenta associar a cada palavra de um texto uma única etiqueta que dê sua classificação gramatical dentro do contexto das outras palavras
- ▶ Mas esta tarefa é complexa
 - ▶ Não se conhece nenhum método analítico para resolvê-la de forma exata
- ▶ Várias abordagens teóricas e práticas têm sido pesquisadas pela comunidade de PLN

Métodos e Modelos

- ▶ Podemos usar diferentes abordagens para criar um etiquetador
- ▶ Primeiro, escolhemos entre algoritmos de aprendizado
 - ▶ Supervisionado
 - ▶ Não-supervisionado
- ▶ Depois decidimos qual modelo teórico usar
 - ▶ Simbólico
 - ▶ Neural
 - ▶ Estatístico
 - ▶ Híbrido

Métodos e Modelos

- ▶ Podemos usar diferentes abordagens para criar um etiquetador
- ▶ Primeiro, escolhemos entre algoritmos de aprendizado
 - ▶ Supervisionado
 - ▶ Não-supervisionado
- ▶ Depois decidimos qual modelo teórico usar
 - ▶ Simbólico
 - ▶ Neural
 - ▶ Estatístico
 - ▶ Híbrido

Nosso Objetivo

Comparar lado a lado a eficiência dos modelos estatísticos de cadeias de Markov de ordem dois e de tamanho variável aplicados ao problema da análise morfo-sintática

Cadeias de Markov

- ▶ Suponha $X = (X_1, \dots, X_T)$ uma seqüência de variáveis aleatórias
- ▶ Com valores num conjunto finito $S = \{s_1, \dots, s_N\}$, o espaço de estados
- ▶ Se X tiver a propriedade

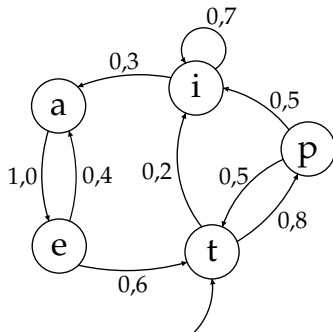
Horizonte Limitado

$$P(X_{t+1} = s_k | X_1, \dots, X_t) = P(X_{t+1} = s_k | X_t)$$

- ▶ Então X é dita ser uma cadeia de Markov

Representação

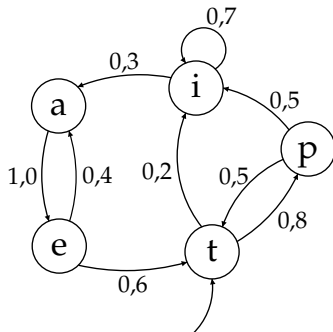
- ▶ Podemos representar uma cadeia de Markov por um diagrama de estados



- ▶ Apesar da propriedade do Horizonte Limitado, podemos considerar que um estado é composto por duas etiquetas

Representação

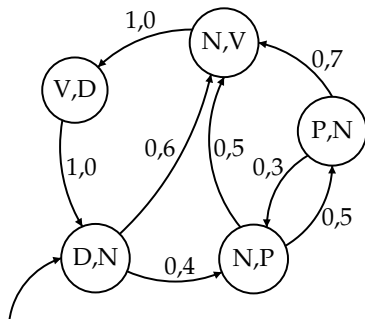
- Podemos representar uma cadeia de Markov por um diagrama de estados



- Apesar da propriedade do Horizonte Limitado, podemos considerar que um estado é composto por duas etiquetas

Representação

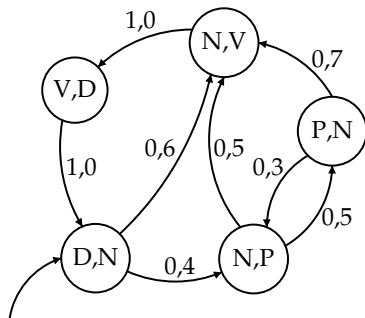
- Podemos representar uma cadeia de Markov por um diagrama de estados



- Uma cadeia de Markov dessa forma é dita ser de ordem dois

Representação

- Podemos representar uma cadeia de Markov por um diagrama de estados



- Uma cadeia de Markov dessa forma é dita ser de ordem dois

Expansão

- ▶ Em um modelo de Markov visível (VMM) sabemos por quais estados o modelo está passando
 - ▶ A seqüência de estados pode ser considerada como a saída
 - ▶ Sua probabilidade pode ser facilmente calculada
- ▶ Mas é um modelo restritivo demais para que seja aplicável a vários problemas de interesse
- ▶ Os modelos de Markov ocultos (HMMs) estendem o conceito de modelos de Markov
 - ▶ Incluem o caso onde a saída não corresponde à seqüência de estados

Modelos de Markov Ocultos (HMMs)

- ▶ Não sabemos qual é a seqüência de estados que o modelo percorre
 - ▶ Apenas alguma função probabilística dela
- ▶ Cada vez que o HMM deixa um estado, um símbolo é emitido
- ▶ No modelo usado no nosso trabalho
 - ▶ Os estados são as etiquetas
 - ▶ E os símbolos emitidos são as palavras

[▶ Continuar](#)

Modelos de Markov Ocultos (HMMs)

As Três Questões Fundamentais

1. Dado um modelo μ , como calculamos eficientemente a probabilidade de uma certa observação O , i.e., $P(O|\mu)$?
2. Dada a seqüência de observações O e um modelo μ , como escolhemos uma seqüência de estados (X_1, \dots, X_{T+1}) que melhor explica O ?
3. Dada uma seqüência de observações O , e um espaço de modelos possíveis encontrados pela variação dos parâmetros de μ , como achamos o modelo que melhor explica O ?

Modelos de Markov Ocultos (HMMs)

As Três Questões Fundamentais

1. Dado um modelo μ , como calculamos eficientemente a probabilidade de uma certa observação O , i.e., $P(O|\mu)$?
2. Dada a seqüência de observações O e um modelo μ , como escolhemos uma seqüência de estados (X_1, \dots, X_{T+1}) que melhor explica O ?
3. Dada uma seqüência de observações O , e um espaço de modelos possíveis encontrados pela variação dos parâmetros de μ , como achamos o modelo que melhor explica O ?

Modelos de Markov Ocultos (HMMs)

As Três Questões Fundamentais

1. Dado um modelo μ , como calculamos eficientemente a probabilidade de uma certa observação O , i.e., $P(O|\mu)$?
2. Dada a seqüência de observações O e um modelo μ , como escolhemos uma seqüência de estados (X_1, \dots, X_{T+1}) que melhor explica O ?
3. Dada uma seqüência de observações O , e um espaço de modelos possíveis encontrados pela variação dos parâmetros de μ , como achamos o modelo que melhor explica O ?

Modelos de Markov Ocultos (HMMs)

As Três Questões Fundamentais

1. Dado um modelo μ , como calculamos eficientemente a probabilidade de uma certa observação O , i.e., $P(O|\mu)$?

Como calculamos $P(w_{1,n}|\mu)$, ou seja, qual a probabilidade da seqüência de palavras $w_{1,n}$ ter sido gerada por μ ?

2. Dada a seqüência de observações O e um modelo μ , como escolhemos uma seqüência de estados (X_1, \dots, X_{T+1}) que melhor explica O ?
3. Dada uma seqüência de observações O , e um espaço de modelos possíveis encontrados pela variação dos parâmetros de μ , como achamos o modelo que melhor explica O ?

Modelos de Markov Ocultos (HMMs)

As Três Questões Fundamentais

1. Como calculamos $P(w_{1,n}|\mu)$, ou seja, qual a probabilidade da seqüência de palavras $w_{1,n}$ ter sido gerada por μ ?
2. Dada a seqüência de observações O e um modelo μ , como escolhemos uma seqüência de estados (X_1, \dots, X_{T+1}) que melhor explica O ?

Como achar a seqüência de etiquetas $l_{1,n}$ tal que $P(l_{1,n}|w_{1,n})$ é a melhor possível, isto é, quais são as melhores etiquetas para as palavras dadas?

3. Dada uma seqüência de observações O , e um espaço de modelos possíveis encontrados pela variação dos parâmetros de μ , como achamos o modelo que melhor explica O ?

Modelos de Markov Ocultos (HMMs)

As Três Questões Fundamentais

1. Como calculamos $P(w_{1,n}|\mu)$, ou seja, qual a probabilidade da seqüência de palavras $w_{1,n}$ ter sido gerada por μ ?
2. Como achar a seqüência de etiquetas $l_{1,n}$ tal que $P(l_{1,n}|w_{1,n})$ é a melhor possível, isto é, quais são as melhores etiquetas para as palavras dadas?
3. Dada uma seqüência de observações O , e um espaço de modelos possíveis encontrados pela variação dos parâmetros de μ , como achamos o modelo que melhor explica O ?

Como treinar μ de modo que $P(w_{1,n}|\mu)$ seja maximizada?

Modelos de Markov Ocultos (HMMs)

As Três Questões Fundamentais

1. Como calculamos $P(w_{1,n}|\mu)$, ou seja, qual a probabilidade da seqüência de palavras $w_{1,n}$ ter sido gerada por μ ?
2. Como achar a seqüência de etiquetas $l_{1,n}$ tal que $P(l_{1,n}|w_{1,n})$ é a melhor possível, isto é, quais são as melhores etiquetas para as palavras dadas?
3. Como treinar μ de modo que $P(w_{1,n}|\mu)$ seja maximizada?

Questão 1

Encontrar a probabilidade de uma observação

- ▶ Avaliar a probabilidade de cada seqüência possível de estados obviamente é inviável
- ▶ Utilizamos o **Algoritmo Progressivo**
 - ▶ Utiliza programação dinâmica
 - ▶ Ou seja, lembra de resultados parciais ao invés de recalculá-los
- ▶ Muito mais eficiente

Questão 2

Encontrar a melhor seqüência de estados

- ▶ Aqui também, testar todas seqüências de estados é inviável
- ▶ Usamos outro algoritmo pra isso, o **Algoritmo de Viterbi**
 - ▶ Também usa programação dinâmica
- ▶ Calcula o caminho mais provável

Questão 3

Estimando os Parâmetros

- ▶ Também é conhecida como o problema do treinamento
 - ▶ Que é um método de aprendizado supervisionado
- ▶ Não existem métodos analíticos conhecidos para resolver o problema de maneira exata
 - ▶ Por isto geralmente usa-se métodos iterativos
 - ▶ Garantem apenas encontrar máximos locais
- ▶ O mais conhecido é o **Algoritmo de Baum-Welch**
 - ▶ Também chamado **Algoritmo Progressivo-Regressivo**

Questão 3

Algoritmo de Baum-Welch

1. Começamos com um modelo inicial
2. Vemos as transições mais usadas
3. E aumentamos suas probabilidades
4. Até convergirmos
 - ▶ Este processo de maximização é geralmente chamado de *treinamento* do modelo
 - ▶ É feito usando-se *dados de treinamento*
 - ▶ Por isto é essencial a existência de um *cópus pré-etiquetado* (de tamanho razoável)

Modelos de Markov Ocultos (HMMs)

Problemas

- ▶ O número de parâmetros de uma cadeia cresce exponencialmente com sua ordem
- ▶ Considerar cadeias de ordem maior que 3 é computacionalmente inviável
- ▶ Muitas palavras podem depender de um contexto maior para serem etiquetadas corretamente
- ▶ Gostaríamos de condicionar o tamanho do contexto à palavra sendo analisada

Cadeias de Markov de Tamanho Variável (VLMCs)

- ▶ Queremos permitir que o tamanho da cadeia de Markov utilizada dependa dos valores passados observados
- ▶ Considere uma cadeia de Markov de ordem finita k

$$P(l_i | l_{i-\infty, i-1}) = P(l_i | l_{i-k, i-1}), \forall l_{i-\infty, i}$$

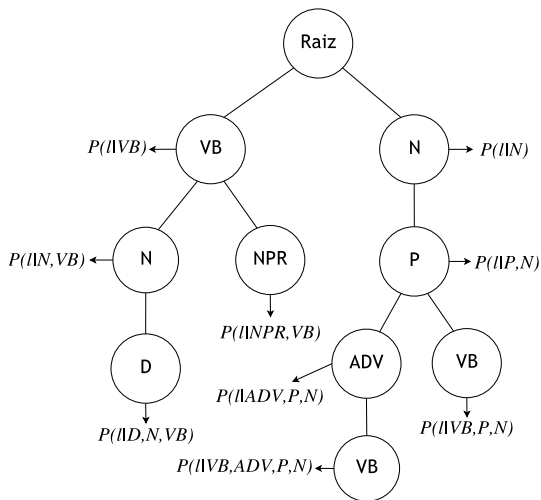
- ▶ A idéia de uma memória de tamanho variável pode ser vista como um corte de estados irrelevantes do histórico $l_{i-k, i-1}$
 - ▶ O histórico de l_i possui apenas alguns estados que precisam ser considerados
 - ▶ Ao conjunto destes estados damos o nome de *contexto* de l_i

Representação

- ▶ Os estados que determinam as probabilidades de transição da VLMC são dados pelos valores da função contexto $c(\cdot)$
- ▶ Representamos estes estados — o espaço de estados minimal da VLMC — como uma árvore

Representação

Árvore de Contexto



Algoritmo de Contexto

- ▶ Construimos uma Árvore de Contexto utilizando o **Algoritmo de Contexto**
- ▶ Primeiro construímos uma árvore de contexto maximal
 - ▶ Utilizando para isto um cópuz de treinamento etiquetado

Algoritmo de Contexto

- ▶ Utilizamos uma função reversa de poda de árvore
 - ▶ Seja vu um galho da árvore, u a folha e v o sub-galho
 - ▶ Então, cortamos a folha fora se

$$\Delta_{vu} = \sum_{l \in \mathcal{L}} P(l|vu) \log \left(\frac{P(l|vu)}{P(l|v)} \right) C(vu) < K$$

- ▶ K é o valor de corte, definido como
$$K = K_n \sim \mathcal{C} \log(n), \mathcal{C} > 2|\mathcal{L}| + 4$$
- ▶ Este processo leva possivelmente a uma árvore menor
- ▶ Repetimos até que mais nenhuma poda seja possível

Algoritmo de Contexto

- ▶ Utilizamos uma função reversa de poda de árvore
 - ▶ Seja vu um galho da árvore, u a folha e v o sub-galho
 - ▶ Então, cortamos a folha fora se

$$\Delta_{vu} = \sum_{l \in \mathcal{L}} P(l|vu) \log \left(\frac{P(l|vu)}{P(l|v)} \right) C(vu) < K$$

- ▶ K é o valor de corte, definido como
$$K = K_n \sim \mathcal{C} \log(n), \mathcal{C} > 2|\mathcal{L}| + 4$$
- ▶ Este processo leva possivelmente a uma árvore menor
- ▶ Repetimos até que mais nenhuma poda seja possível

Algoritmo de Contexto

- ▶ Utilizamos uma função reversa de poda de árvore
 - ▶ Seja vu um galho da árvore, u a folha e v o sub-galho
 - ▶ Então, cortamos a folha fora se

$$\Delta_{vu} = \sum_{l \in \mathcal{L}} P(l|vu) \log \left(\frac{P(l|vu)}{P(l|v)} \right) C(vu) < K$$

- ▶ K é o valor de corte, definido como
$$K = K_n \sim \mathcal{C} \log(n), \mathcal{C} > 2|\mathcal{L}| + 4$$
- ▶ Este processo leva possivelmente a uma árvore menor
- ▶ Repetimos até que mais nenhuma poda seja possível

Construção dos Etiquetadores

Etiquetador de Ordem Dois

- ▶ Problema da esparsidade dos dados
- ▶ Utilizamos interpolação linear
 - ▶ $P(l_i|l_{1,i-1}, w_i) = \lambda_1 P_1(w_i|l_i) + \lambda_2 P_2(l_i|l_{i-1}) + \lambda_3 P_3(l_i|l_{i-1}, l_{i-2})$
 - ▶ Como obter os lambdas?
- ▶ Colocamos os lambdas em cada estado
- ▶ E então treinamos os valores dos lambdas ao invés da probabilidade das etiquetas

[▶ Continuar](#)

Construção dos Etiquetadores

Etiquetador de Ordem Dois

- ▶ Calculamos as probabilidades relativas a partir de um **cópus etiquetado**
 - ▶ $P(w_3|l_3) = \frac{C(w_3, l_3)}{C(l_3)}$
 - ▶ $P(l_3|l_2) = \frac{C(l_2, l_3)}{C(l_2)}$
 - ▶ $P(l_3|l_1, l_2) = \frac{C(l_1, l_2, l_3)}{C(l_1, l_2)}$

Construção dos Etiquetadores

Etiquetador de Ordem Dois

- ▶ Calculamos as probabilidades relativas a partir de um cópús etiquetado
 - ▶ $P(w_3|l_3) = \frac{C(w_3, l_3)}{C(l_3)}$
 - ▶ $P(l_3|l_2) = \frac{C(l_2, l_3)}{C(l_2)}$
 - ▶ $P(l_3|l_1, l_2) = \frac{C(l_1, l_2, l_3)}{C(l_1, l_2)}$
- ▶ Por que usamos o teórico $P(w_3|l_3)$ ao invés do natural $P(l_3|w_3)$?

Construção dos Etiquetadores

Etiquetador de Ordem Dois

- ▶ Calculamos as probabilidades relativas a partir de um córpus etiquetado
 - ▶ $P(w_3|l_3) = \frac{C(w_3, l_3)}{C(l_3)}$
 - ▶ $P(l_3|l_2) = \frac{C(l_2, l_3)}{C(l_2)}$
 - ▶ $P(l_3|l_1, l_2) = \frac{C(l_1, l_2, l_3)}{C(l_1, l_2)}$
- ▶ Por que usamos o teórico $P(w_3|l_3)$ ao invés do natural $P(l_3|w_3)$?
 - ▶ Porque o tamanho do córpus influencia

Construção dos Etiquetadores

Etiquetador de Tamanho Variável

- ▶ Para etiquetar uma palavra consideramos duas probabilidades
 1. A probabilidade de uma etiqueta gerar a palavra: $P(w|l)$
 2. E a probabilidade do contexto: $P(l_j|c(l_{1,i-1}))$
- ▶ Para a probabilidade da palavra usamos a probabilidade relativa (contagem)
- ▶ Para a do contexto construímos a Árvore de Contexto
- ▶ Tudo isto a partir do cópús de treino
- ▶ **Por isto é importante a existência de um cópús pré-etiquetado**

Etiquetador de Tamanho Variável

Questão da interpolação

- ▶ Queremos a probabilidade de uma etiqueta dada a palavra e o contexto passado
- ▶ Como no etiquetador de ordem 2, usamos interpolação
$$P(l_i | w_i, c(l_{i-\infty, i-1})) = \lambda_1 P(w_i | l_i) + \lambda_2 P(l_i | c(l_{i-\infty, i-1}))$$
- ▶ Mas nos testes, não achávamos valores bons para os lambdas

Etiquetador de Tamanho Variável

Questão da interpolação

- ▶ Experimentamos multiplicar as probabilidades

$$P(l_i | w_i, c(l_{i-\infty, i-1})) = P(w_i | l_i) * P(l_i | c(l_{i-\infty, i-1}))$$

- ▶ Estranhamente, esta equação deu melhores resultados do que quaisquer combinações de lambdas na equação anterior
- ▶ Não encontramos explicações teóricas para isto

Etiquetador de Tamanho Variável

Questão da interpolação

- ▶ Experimentamos multiplicar as probabilidades

$$P(l_i | w_i, c(l_{-\infty, i-1})) = P(w_i | l_i) * P(l_i | c(l_{-\infty, i-1}))$$

- ▶ Estranhamente, esta equação deu melhores resultados do que quaisquer combinações de lambdas na equação anterior
- ▶ Não encontramos explicações teóricas para isto

Etiquetador de Tamanho Variável

Questão da interpolação

- ▶ Experimentamos multiplicar as probabilidades

$$P(l_i | w_i, c(l_{i-\infty, i-1})) = P(w_i | l_i) * P(l_i | c(l_{i-\infty, i-1}))$$

- ▶ Estranhamente, esta equação deu melhores resultados do que quaisquer combinações de lambdas na equação anterior
- ▶ Não encontramos explicações teóricas para isto
- ▶ Mas deixamos assim ;-)

Decisões de algoritmo

- ▶ Na construção da árvore de contexto, pela teoria, o valor da constante de corte é igual a
$$K = C \log(n), C > 2|\mathcal{L}| + 4$$
- ▶ Mas este valor de K não deu bons resultados
 - ▶ Com o cópuz de treino, era igual a 10350
 - ▶ **Muito grande!**

Decisões de algoritmo (valor de corte)

- ▶ Modificamos para

$$K = \frac{\log(n)}{\log(|\mathcal{L}|)} \cdot \frac{n}{|S|}$$

- ▶ Ficou valendo, com corpus inteiro de treino, 54,6615
- ▶ Pode ser definido na execução do etiquetador

▶ Continuar

Tratamento das Palavras Desconhecidas

- ▶ Construimos uma árvore de **sufixos**
- ▶ Na fase de treinamento:
 - ▶ O tamanho do sufixo depende do tamanho da palavra
 - ▶ Usamos metade da palavra
- ▶ Na fase de etiquetagem/teste, para uma palavra desconhecida:
 - ▶ Procuramos na árvore o maior sufixo possível

Implementações

- ▶ Utilizamos a linguagem de programação C++
- ▶ Fizemos bastante uso da STL (Standard Template Library)
- ▶ Estruturas genéricas
 - ▶ `map`
 - ▶ `set`
 - ▶ `vector`
- ▶ Algoritmos eficientes sobre estas estruturas

Testes Realizados

Recursos

- ▶ Utilizamos o *córpus Tycho Brahe* para o aprendizado
 - ▶ Pré-etiquetado no formato $\langle \text{palavra} \rangle / \langle \text{etiqueta} \rangle$
 - ▶ Selecionamos alguns textos
 - ▶ Num total de 1.035.593 palavras/etiquetas
- ▶ Dividimos o *córpus* em 3/4 e 1/4
- ▶ Criando um *córpus* de treino e um de teste
 - ▶ 775.602 palavras de treino
 - ▶ 259.991 palavras de teste

Testes Realizados

Conjuntos de testes

- ▶ Seleccionamos aleatoriamente partes de 5%, 10%, ..., 95% do córpus de treino
- ▶ Executamos 10 vezes o treinamento e a etiquetagem com cada uma destas partes
- ▶ Nos dois etiquetadores
 - ▶ **HMM Tagger**
 - ▶ **VLMC Tagger**

Precisão

Comparação entre os dois etiquetadores

- ▶ A precisão é dada por

$$\text{Precisão} = \text{Taxa de acerto} = \frac{\text{Número de etiquetas coincidentes}}{\text{Número total de etiquetas}}$$

- ▶ Com o **HMM Tagger** obtivemos

93,4875%

- ▶ Com o **VLMC Tagger** obtivemos

95,5129%!

Precisão

Comparação entre os dois etiquetadores

- ▶ A precisão é dada por

$$\textit{Precisão} = \textit{Taxa de acerto} = \frac{\textit{Número de etiquetas coincidentes}}{\textit{Número total de etiquetas}}$$

- ▶ Com o **HMM Tagger** obtivemos

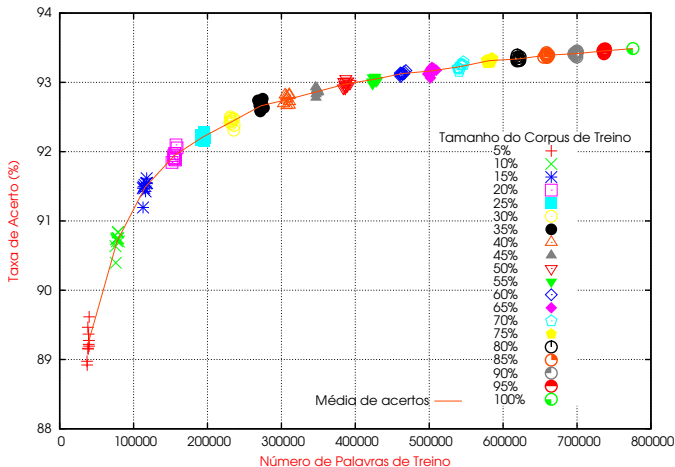
93,4875%

- ▶ Com o **VLMC Tagger** obtivemos

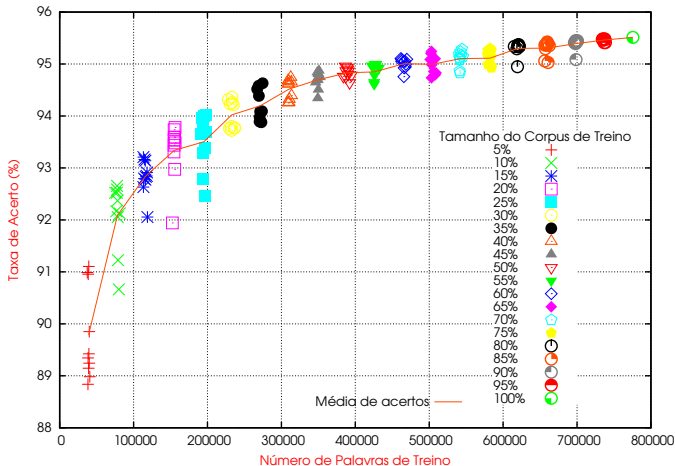
95,5129%!

Precisão

HMM Tagger

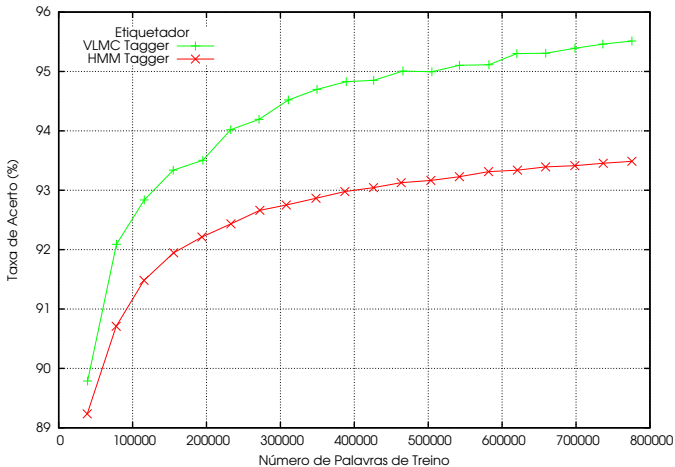


Precisão VLMC Tagger



Precisão

HMM Tagger e VLMC Tagger



Tempo

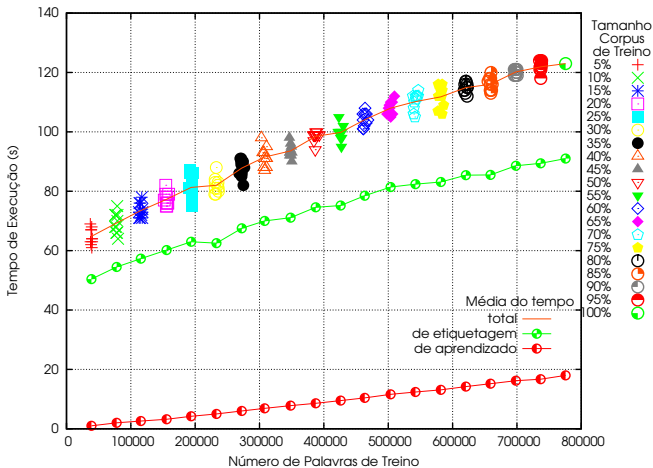
Comparação entre os dois etiquetadores

- ▶ Calculamos o tempo total e os tomados no aprendizado e na etiquetagem

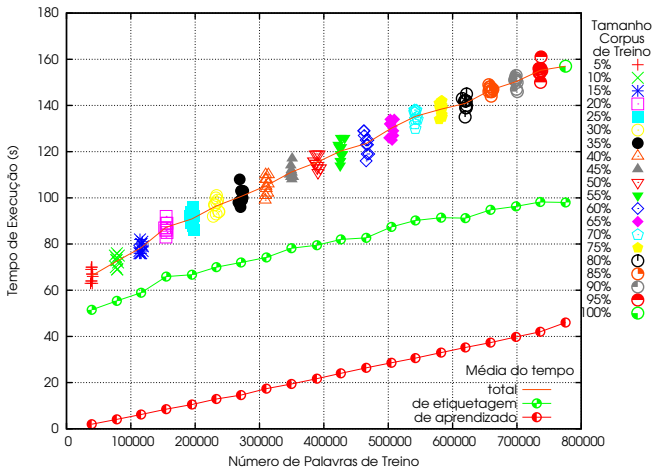
Tempos (s)	HMM Tagger	VLMC Tagger
Aprendizagem	18	46
Etiquetagem	91	98
Total	123	157

Tempo

HMM Tagger

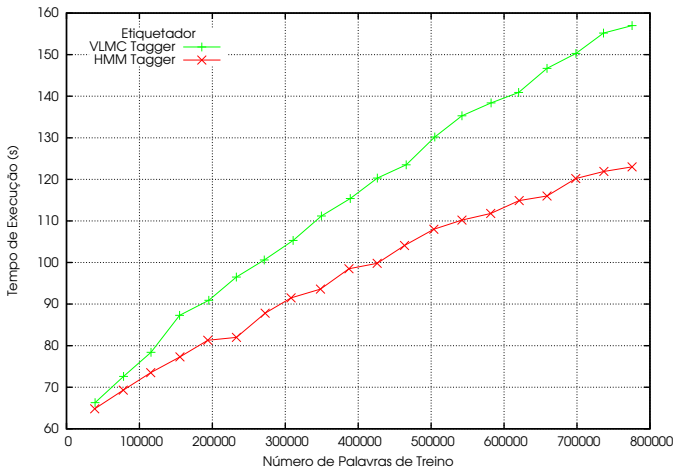


Tempo VLMC Tagger



Tempo

HMM Tagger e VLMLC Tagger



Pós-Correções

No **VLMC Tagger**

- ▶ Achamos um erro lógico na implementação do aprendizado no **VLMC Tagger**
- ▶ Algumas probabilidades estavam erradas
- ▶ Também demos maior probabilidade de uma palavra com a inicial maiúscula ser um nome próprio
- ▶ Com estas correções, obtivemos uma precisão de

95.7745%!

Outros testes

Outros Testes Realizados

Com outro córpus

- ▶ Um córpus em inglês, mal estruturado STOP! [citar qual e fonte]
- ▶ Dividimos aleatoriamente o córpus em 3/4 e 1/4
- ▶ Tivemos uma precisão média de 96,xx% STOP! [conferir]

Outros Testes Realizados

Com outro cópús

- ▶ Testamos com o cópús MAC-MORPHO, do NILC-ICMC-USP
- ▶ 1.221.331 palavras
- ▶ Dividindo aleatoriamente o cópús em 3/4 e 1/4, obtivemos em média

93,20%

Conclusões

- ▶ Comparamos e avaliamos de forma transparente e imparcial dois modelos de Markov aplicados à análise morfo-sintática
- ▶ Usamos pouca informação lingüística na construção dos etiquetadores, deixando-os apenas probabilísticos
 - ▶ Não dependemos do conjunto de etiquetas do córpus
- ▶ Finalmente, vimos que VLMCs são melhores que HMMs na aplicação à análise morfo-sintática
- ▶ E possuem potencial para uma eficiência ainda maior

Trabalhos Futuros

- ▶ Árvores de contexto menores deram melhor resultado
 - ▶ Nossa suposição de que teríamos fortes dependências de contexto não se confirmou
 - ▶ Talvez um passado mais antigo seja mais importante que um recente
 - ▶ Para testar isto poderia-se tentar criar uma árvore mais flexível, com nós “curingas”
- ▶ Com base nos testes, a precisão parece depender bastante do corpus utilizado para aprendizagem
 - ▶ São necessários mais testes comparativos
- ▶ O etiquetador de tamanho variável (**VLMC Tagger**) estará disponível sob a licença GPL
 - ▶ Comunidade poderá melhorar o desempenho

Este é o fim.

Este/D é/SR-P o/D fim/N ./.