Pontifícia Universidade Católica do
Rio Grande do Sul
Faculdade de Informática
Curso de Bacharelado em
Ciência da Computação

Parsing Probabilístico para a Língua Portuguesa

Trabalho de Conclusão II

Autores:

Rodrigo R M Kochenburger

Marlon Gomes Lopes

Orientador:

Carlos Augusto Prolo

Porto Alegre, Dezembro de 2009

Resumo

O objetivo desse trabalho é desenvolver uma ferramenta de *parsing* para o português baseado em modelos probabilísticos. Como *treebank* alimentador do processo utilizamos o Floresta Sintática. Os modelos probabilísticos estudados são os desenvolvidos por Michael collins, e usaremos como base o parser implementado por Dan Bikel, avaliando as possibilidades de parametrização e extenção possíveis.

Também será necessário o desenvolvimento de módulos de pré-processamento para adaptação do corpus utilizado.

Exploramos várias configurações em termos de parametrização possíveis do parser e melhorias no corpus utilizado para treino do parser.

Nossa melhor configuração atingiu bla bla bla.

1 Resultados Obtidos "REFAZER ....."

1.1 Metodologia

Este trabalho possui um forte componente experimental e exploratório. Assim, em termos metodológicos, a cada experiência realizada, os resultados obtidos devem ser analisados quantitativa e qualitativamente, para orientar as correções nos parâmetros do *parser* ou indicar a necessidade de alterações como: a) de pré-processamento ou pós-processamento dos casos; ou b) no código do *parser*. Nesse sentido, a avaliação quantitativa é um componente importante e será feita de forma rigorosa. Pretende-se utilizar os métodos de avaliação tradicionais de *precision/recall* [BLA93] e possivelmente outras a definir.

Será usado um sistema de controle de versão que permite que se trabalhe com diversas versões dos arquivos de trabalho e versões do software durante nossos testes e implementações.

Inicialmente trabalhamos sobre a influêcia da escolha do núcleo *(Head)* dos constituíntes que é o principal fator na implementação dos modelos propostos por Collins [COL99].

O corpus do Bosque ja tem anotados muitos dos *heads* das sentenças disponibilizadas, porém o parser de bikel precisa de todos os *heads*, e verificou-se que nem todas as sentenças possuem essa informação, tendo que ser analisada e construída de forma empírica.

A ferramenta de Bikel tem um módulo para especificação de regras para determinar o *head* de um constituinte e optou-se por se usar esse módulo e construir regras baseadas nas regras de formação da lingua portuguesa.

Primeiro usou-se as regras .....

Utilizamos nos primeiros experimentos as regras definidas para o ingles nao atingindo

resultados satisfatórios

Nos experimentos seguintes reduzimos o número de regras apenas para as regras básicas de encontro de heads de sentenças, testou-se primeiro selecionando como head de um constituinte o a palavra mais a direita e logo a palavra mais a esquerda.

Os resultados deste teste mostrou que o conjunto de regras ....

è interessante mensionar que a construção das regras foi incremental ... a título de curiosidade figura tal ...

O primeiro conjunto de regras usadas é o que estamos usando atualmente bem melhor que o original, conforme resultados apresentados, esses resultados foram ... na arvore de experimentos

Formado o conjunto de regras de head-find . A segunda sequencia de experimentos tera a ver com a granilaridade das POS tags. ....

...

A ideia basica é que tags devem ser distintos quando a categorias tem distribuições sintáticas diferentes, por outro lado se duas classes tem mesma distribuição ..., ...

Verificou-se conforme esperado que para verbos é bastante relevante as sub-categorias pois a distribuição sintática das diferentes categorias verbais é bem distinta, da mesma forma para pronomes porque os possessivos tem diferente distribuição que os pessoais; os .. tem posição de pré-modificadores nominal e os outros ...

O último caso relativo a nomes refere-se

Detalhar mais o metodo .....

1.2 Método de Avaliação

As medidas de avaliação do *parser* seguirão a proposta de GEIC/Parseval [BLA91], possivelmente adaptado conforme [COL97] para ignorar pontuação e não considerar a marcação de POS na avaliação.

Em particular, serão usadas as medidas de *Labeled Precision* (LP) e *Labeled Recall* (LR) e sua média harmônica ($F_{\beta=1}$) ou *F-Score*, descritas abaixo:

$$LP = \frac{número\ de\ constituintes\ corretas\ na\ análise\ proposta}{número\ de\ constituintes\ da\ análise\ proposta}$$

$$LR = \frac{número\ de\ constituintes\ corretas\ na\ análise\ proposta}{número\ de\ constituintes\ do\ treebank\ analisado}$$

$$F_{\beta=1} = \frac{2 * LP * LR}{LP + LR}$$

O termo *Labeled* se refere ao fato de que uma constituinte, para contar como corretamente recuperado, deve acertar a extensão correta do texto bem como o rótulo do constituinte.

O procedimento de avaliação compara a saída do parser com as análises anotadas no *treebank*; usa a informação de parentização da representação do *treebank* de uma sentença e a análise produzida pra computar tres medidas: *crossing braqkets, precision e recall*, neste trabalho não utilizaremos a medida de *crossing braquets*.

Estas métricas são chamadas métricas estruturais, e são baseadas na avaliação dos limites dos sintágmas. Os algoritmos de parsing tem por objetivo otimizar uma métrica em comum, que é a probabilidade de se ter uma árvore corretamente rotulada, ou seja, com uma marcação correta dos limites dos constituíntes. Assim dado um nó em uma árvore sintática, a sequencia de palavras dominadas por esse nó forma um sintágma, sendo o limite do sintágma representado por um interválo inteiro *[i,j]*, em que i representa o índice da primeira palavra e j o da última palavra do sintágma.

Black [BLA91] propõe três medidas estruturais para avaliar sistemas de parsing: Labeled Precision, Labeled Recall e Crossing-Brackets. Segundo Lin (1995), esse esquema de avaliação pode ser classificado como em nível de sintágma, ou nível de sentença. As medidas de Labeled Precision e Labeled Recall são computadas da seguinte forma:

Os limites dos sintágmas na resposta (análise produzida pelo parser) e no gold (análise do treebank) são tratados como dois conjuntos (A e K), em que A é a análise obtida do parser proposto e K, o gold do treebank a ser usado na avaliação. O Labeled Recall é definido como a percentagem no gold que também é encontrada na resposta ((A K)/K). A Labeled Precision é definica como a percentagem de limites no sintagma da resposta que também é encontrada no gold ((A K)/A).

As medidas propostas no PARSEVAL partem de um pressuposto de que um constituinte esta correto se corresponde ao mesmo conjunto de palavras (ignorando qualquer caractere de pontuação) e tem o mesmo rótulo que um constituínte no treebank.

Exemplo: Considere (1) gold e (2) análise do parser:

1. (Eles ((chegaram) ontem))
2. ((Eles (chegaram)) (ontem))

Temos:

limite para os sintagmas em (1): (0,2),(1,1) e (1,2).
limite para os sintagmas em (2): (0,2),(1,1),(0,1) e (2,2).

Pontuações em (2): Labeled Precision = 2/4 = 50%, Labeled Recall = 2/3 = 66.7%

Essas pontuações tem que ser consiteradas juntas para ter significado. Por exemplo, tratando a sentença como uma lista de palavras (Eles chegaram ontem) teriamos 100% de Labeled Precision. Entretanto, Labeled Recall seria baixa (1/3 = 33%).

1.3 Resultados

Nos primeiros experimentos foram feitas apenas alterações nas regras de *head-find*. O Corpus de treino, desenvolvimento e teste utilizado é o Bosque da Floresta Sintática que contém um total de 5200 sentenças separadas em 4160 para treino 520 para desenvolvimento e 520 para teste, nas primeiras baterias de teste utilizamos pouco menos de 100 sentenças para ajustar parâmetros mais rapidamente, na última foi utilizado 520 sentenças na fase de desenvolvimento e teste.

Primeiro teste

Regras de *head-find* usadas:

```
(
(NP (l N PROP PRON-PERS PRON-INDP N-ADJ NP))
(VP (l V-FIN V-INF V-PCP V-GER) (l VP))
(ADJP (l ADJ ADJP) (l PRON-DET))
(ADVP (r ADV ADVP))
(CU (r CONJ-C CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Resultados obtidos

```
Bracketing Recall         =  62.38
Bracketing Precision      =  68.80
Tagging accuracy          =  58.84
```

Achamos que os resultados estavam baixos devido as TAGS de anotação dos verbos possuírem um hífen, e o *parser* não reconhece o hífen como parte da TAG.

Segundo teste

Alteramos as *tags* que possuíam o caractere hífen ("-"), pois o *parser* desconsidera a parte a direita do mesmo, e o programa de avaliação não. Por isso os resultados da avaliação são prejudicados.

As regras de *head-find* foram alteradas para:

```
(
(NP (l N PROP PRONPERS PRONINDP NADJ NP))
(VP (l VFIN VINF VPCP VGER) (l VP))
(ADJP (l ADJ ADJP) (l PRONDET))
(ADVP (r ADV ADVP))
(CU (r CONJC CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Resultados obtidos

```
Bracketing Recall         =  61.47
Bracketing Precision      =  68.20
Tagging accuracy          =  79.83
```

Apenas a precisão das tags melhorou, porém não houve melhora nas métricas *Bracketing Recall* e *Bracketing Precision*.

Terceiro teste

No próximo teste pretendemos verificar o quanto o *parser* consegue melhorar seu desempenho quando não tem que especificar se o verbo esta em uma das quatro categorias, tendo apenas a anotação genérica de verbo.

As regras de *head-find* foram alteradas para:

```
(
(NP (l N PROP PRONPERS PRONINDP NADJ NP))
(VP (l V) (l VP))
(ADJP (l ADJ NUM) (l ADJP) (l PRONDET))
(ADVP (r ADV ADVP))
(CU (r CONJC CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Resultados obtidos

```
Bracketing Recall         =  60.42
Bracketing Precision      =  67.42
Tagging accuracy          =  80.73
```

Novamente, a precisão das tags aumentou, porém o *Precision* e *Recall* diminuíram. A generalização da tag V fez com que categorias sintáticas externas como ICL, ACL e FCL fossem influenciadas.

Quarto teste

Como já verificado com as tags, o *parser* também exibe problemas com o caractere hífen ("-") no texto. Após substituir todos os hífens por *underscore* ("_"), os resultados obtidos foram melhores.

As regras de *head-find* são as mesmas.

```
(
(NP (l N PROP PRONPERS PRONINDP NADJ NP))
(VP (l V) (l VP))
(ADJP (l ADJ NUM) (l ADJP) (l PRONDET))
(ADVP (r ADV ADVP))
(CU (r CONJC CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Resultados obtidos

```
Bracketing Recall         =  67.42
Bracketing Precision      =  68.09
Tagging accuracy          =  90.36
```

Percebemos uma melhora significativa nos valores de *Bracketing Recall* e *Tagging accuracy*, a partir de agora sempre substituiremos os caracteres "-" por "_".

Quinto teste

Na próxima experiência vamos utilizar novamente a especialização das categorias de verbo e com um corpus de desenvolvimento e teste maior.

```
(
(NP (l N PROP PRONPERS PRONINDP NADJ NP))
(VP (l VFIN VINF VPCP VGER) (l VP))
(ADJP (l ADJ ADJP) (l PRONDET))
(ADVP (r ADV ADVP))
(CU (r CONJC CU , ;))
(X (l VP))
(PP (l PRP PP))
(FCL (l VP) (l NP))
(ICL (l VP) (l NP))
(ACL (l VP) (l NP))
(* (l))
)
```

Resultados obtidos

```
Bracketing Recall         =  72.50
Bracketing Precision      =  72.73
Tagging accuracy          =  90.46
```

Verificamos uma melhora expressiva em relação aos primeiros testes realizados após a retirada dos hífens das *tags* de categoria de verbos e após a substituição dos caracteres "-" por "_".

Uma avaliação mais aprofundada será feita posteriormente.