SuSAna - Módulo multifuncional de análise sintáctica de superfície

Fernando M. Batista, Nuno J. Mamede

L^{2}F /INESC-ID
Laboratório de Sistemas de Língua Falada
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{Fernando.Batista,Nuno.Mamede}@inesc-id.pt

Abstract

This paper describes a surface syntactic analysis module for unrestricted text. The algorithm goes beyond usual bracketing approaches, since its capable of recognising the boundaries, the internal structure and syntactic category of sentence constituents. It is also robust, since it is capable of treating unkown lexical units and limits the number of hypothesis for each analysis. It can be used as a standalone application, in a client/server architecture, or integrated in a natural language processing system. It allows different types of analysis and several kinds of results can be obtained. This paper also presents performance tests concerning processing time and number of hypothesis for each analysis.

Keywords: Syntactic analysis, Robust Analysis, Surface Analysis, Natural Language, Computational Language Processing

1 Introdução

O aumento significativo da quantidade de informação em suporte digital que se tem verificado durante os últimos anos, em especial devido à expansão da Internet e ao desenvolvimento dos meios de comunicação social, tem permitido a criação de repositórios de textos cada vez mais abrangentes, que constituem recursos de extrema importância para o processamento da língua. Estes repositórios abrangem uma grande quantidade de domínios e muitas vezes o tipo de construções aí presentes não são contemplados pelas gramáticas comuns. Surge assim a necessidade de desenvolver sistemas de processamento e manipulação de texto que tenham a capacidade de lidar com grandes quantidades de texto real. Os sistemas de tratamento de informação ao nível morfossintáctico encontram-se em avançado estado de desenvolvimento. Por um lado, os analisadores morfológicos de larga cobertura, conseguem taxas de erro reduzidas e implementam algoritmos bastante rápidos, por outro lado, os sistemas de desambiguação encontram-se em franco desenvolvimento, oferendo taxas de erro cada vez mais baixas [Ribeiro et al., 2002]. Assim, estão a surgir as condições necessárias para o processamento da língua aos níveis sintáctico e semântico.

A análise sintáctica de um corpus torna visível uma grande quantidade de informação, que possibilita o desenvolvimento de aplicações mais complexas e poderosas. O processamento sintáctico tem aplicação nas mais variadas áreas como é o caso da síntese e reconhecimento de fala [Fach, 1999, Abney, 1992], pesquisa e extracção de informação e tradução automática. A análise sintáctica é também o ponto de partida para os sistemas de processamento semântico.

O módulo de análise sintáctica de superfície aqui apresentado foi desenvolvido com base no estudo do funcionamento do protótipo de análise de superfície AF[1], descrito em [Hagège, 2000], que permite reconhecer fronteiras, estrutura interna e categoria sintáctica de sintagmas, tanto simples como mais complexos, como é o caso dos sintagmas nucleares, apresentados na secção 5.1. O algoritmo de SuSAna[2] foi concebido de forma a conseguir uma análise eficiente, permitir diferentes configurações para a análise e obter diferentes tipos de resultados. A estrutura da gramática que utiliza provém da estrutura da gramática utilizada pelo AF, contudo adopta-se agora o formato XML para a especificação das suas regras, que por sua vez foram também adaptadas e melhoradas. Resultante da forma como foi concebido, a utilização de SuSAna pode ser feita por uma aplicação isolada, numa plataforma cliente/servidor ou integrado num sistema de processamento de Língua Natural.

2 Descrição linguística

O conjunto de regras utilizadas por SuSAna constituem o que se designam por *descrição linguística* e por vezes se chama de *gramática*. A estrutura desta gramática provém da gramática utilizada pelo AF: as regras para uma dada língua, são derivadas a partir de um conjunto de propriedades, que correspondem à formalização das características linguísticas dessa língua. Durante a fase inicial de desenvolvimento de SuSAna, foi utilizada a descrição linguística do AF tal como estava, de forma a facilitar a comparação de resultados. A sua utilização foi realizada com o auxílio das ferramentas FLEX (Gerador de analisadores lexicais) e BISON (gerador de analisadores sintácticos) que além de oferecerem grande flexibilidade, permitem também um desenvolvimento rápido. O formato da gramática mostrou-se, contudo, pouco robusto e com algumas limitações.

Para adaptar a descrição linguística às exigências impostas a todos os módulos desenvolvidos no L^{2}F[3], utiliza-se o formato XML para a representação das regras. A figura 1 mostra o DTD[4] da gramática resultante, utilizada pela SuSAna.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!ELEMENT LangSpec
     (topmodel|remplace|block|preference)*>
<!ELEMENT topmodel EMPTY>
<!ATTLIST topmodel name CDATA #REQUIRED>
<!ELEMENT remplace (submodel)*>
<!ATTLIST remplace name CDATA #REQUIRED>
<!ELEMENT submodel EMPTY>
<!ATTLIST submodel name CDATA #REQUIRED>
<!ELEMENT block (nextmod)*>
<!ATTLIST block
        name    CDATA #REQUIRED
        sup     CDATA #REQUIRED
        start   (0|1|2) #REQUIRED
        end     (0|1|2) #REQUIRED>
<!ELEMENT nextmod EMPTY>
<!ATTLIST nextmod name CDATA #REQUIRED>
<!ELEMENT preference EMPTY>
<!ATTLIST preference
        prefmod CDATA #REQUIRED
        discmod CDATA #REQUIRED
        supmod  CDATA #IMPLIED
        prevmod CDATA #IMPLIED
        word    CDATA #IMPLIED
        cat     CDATA #IMPLIED>
```

Figura 1: DTD da gramática da SuSAna

2.1 Definição de modelo

Designam-se por *modelos*, as estruturas que a SuSAna identifica, definidas sempre a partir de um conjunto de propriedades. Um modelo para um conjunto de propriedades é uma sequência de símbolos que satisfaz o conjunto das propriedades consideradas [Hagège, 2000]. Ao considerar as características morfossintácticas das palavras como símbolos, um modelo para um conjunto de propriedades linguísticas, no contexto da análise sintáctica, é uma sequência de categorias morfossintácticas que satisfazem essas propriedades.

De forma a simplificar a notação, estende-se aqui a noção de modelo, de forma a abranger também as categorias morfossintácticas, que constituem a base da análise. Assim, utilizam-se os termos,

[1] AF é acrónimo de *Analisador por Folhas*
[2] SuSAna é acrónimo de *Surface Syntactic Analyser*.
[3] acrónimo de Laboratório de sistemas de Língua Falada. O L^{2}F faz parte da unidade de Investigação e Desenvolvimento do Instituto de Engenharia de Sistemas e Computadores (INESC-ID).
[4] Definição do tipo de documento (*Document Type Definiton*)

modelo terminal e *modelo não terminal,* para distinguir as categorias dos modelos.

2.2 Modelo de topo

Na figura 1 verifica-se que, dos quatro tipos de elementos básicos, o elemento *topmodel* tem apenas um atributo. Este elemento é apenas declarado uma vez e define o elemento de topo por omissão. O elemento de topo indica o modelo pelo qual se inicia a análise e corresponde à estrutura linguística que se pretende analisar. Por exemplo, o identificador *sentence* poderia ser utilizado para fazer a análise de frases, o identificador *address* seria usado para identificar endereços e o identificador *par* poderia ser usado para processar parágrafos. Este elemento pode ser alterado em tempo de execução.

2.3 Comportamento dos modelos

O comportamento de modelos dentro de outros modelos é descrito pelo elemento *block*. Os atributos *name* e *sup* são obrigatórios e indicam que o modelo *name* pode ocorrer dentro do modelo *sup*. Os atributos *start* e *end* podem tomar os valores 0 (nunca), 1 (sempre) ou 2 (por vezes) e indicam o modo como pode ser feita essa ocorrência. Por exemplo, o valor 2 no atributo *start* indica que o modelo pode ocorrer no início, mas essa ocorrência não é obrigatória.

O exemplo seguinte indica que a categoria, ou modelo terminal, *arti_s* (artigo indefinido singular) pode ocorrer num modelo *mpp_n,* embora nunca o possa começar nem acabar. Também indica que os modelos que podem ocorrer após um *arti_s,* dentro de um *mpp_n* (modelo proposicional nuclear), podem ser apenas: *nc* (nome comum), *madj* (modelo adjectival) ou *inconnu* (palavra desconhecida).

```
<block name="arti_s" sup="mpp_n" start="0" end="0">
<nextmod name="nc"/>
     <nextmod name="nadj"/>
     <nextmod name="inconnu"/>
</block>
```

Podem também ser utilizadas variáveis para designar um conjunto de modelos. No próximo exemplo, indica-se que a categoria *coord* (coordenação) pode ocorrer em qualquer modelo, embora não o possa começar nem terminar e que pode ser seguida, dentro desse modelo por qualquer outro.

```
<block name="coord" sup="VARM" start="0" end="0">
     <nextmod name="X"/>
</block>
```

2.4 Hierarquia de modelos

A gramática permite definir hierarquias para os modelos, possibilitando a elaboração de regras mais simples e abrangentes. Neste ponto a gramática sofreu uma evolução em relação à gramática utilizada pelo AF, que apenas permite definir relações hierárquicas para os modelos terminais. As relações hierárquicas entre os modelos são definidas pelo uso do elemento *remplace.*

O exemplo seguinte indica que *n* (nome) pode ser: *nc* (nome comum), *npr* (nome próprio) *nadj* (ambiguidade nome/ adjectivo). Por sua vez *nc* pode ser *nc_s* (singular), *nc_p* (plural), *nc1* (nomes comuns contáveis), *nc2* (nomes comuns massivos).

```
<remplace name="n">
       <submodel name="nc"/>
       <submodel name="npr"/>
       <submodel name="nadj"/>
</remplace>
<remplace name="nc">
       <submodel name="nc_s"/>
       <submodel name="nc_p"/>
</remplace>
<remplace name="nc">
       <submodel name="nc1"/>
       <submodel name="nc2"/>
</remplace>
<remplace name="nc_s">
       <submodel name="nc1_s"/>
       <submodel name="nc2_s"/>
</remplace>
<remplace name="nc1">
       <submodel name="nc1_s"/>
       <submodel name="nc1_p"/>
</remplace>
```

Note-se que o resultado não pode ser representado como uma árvore, mas sim como um grafo dirigido, tal como mostra a figura 2.

2.5 Preferências

Um dos mecanismos que permite a limitação do número de análises, mantendo o resultado "usável" por outras aplicações são as restrições impostas pelas preferências. Este tipo de regra permite escolher um de dois caminhos possíveis na análise

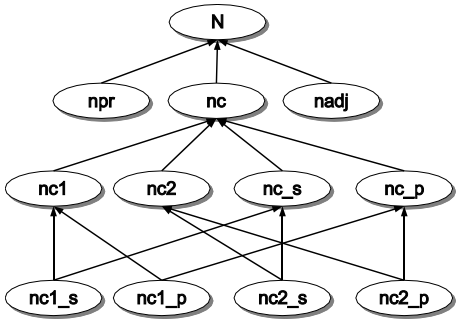

Figura 2: Hierarquia de modelos

e é descrita pelo elemento *preference*. Os atributos *prefmod* e *discmod* que indicam o modelo que se prefere e o modelo que se despreza são obrigatórios (ver figura 1). Os restantes atributos são opcionais e limitam o contexto em que se aplica a restrição. *Supmod* indica o modelo superior, *prevmod* refere-se ao modelo anterior, *word* é a palavra e *cat* é a categoria que se está a analisar. Esta combinação de atributos opcionais aumenta a flexibilidade e constitui, por isso, uma grande evolução em relação à gramática utilizada pelo AF.

Nos dois exemplos seguintes, verifica-se a presença obrigatória do modelo que se prefere e o modelo que se despreza. No primeiro exemplo, indica-se que se prefere o modelo *nc1_p* (nome comum singular) ao modelo *adj3_p* (adjectivo do tipo 3 no plural)[5] dentro do modelo *m_nn* (modelo nominal nuclear). No segundo exemplo a preferência será aplicada em qualquer contexto, desde que seja depois do modelo *copv_n* (modelo de verbo copulativo nuclear) e a categoria que se está a analisar seja *todo_p* (pronome indefinido, plural).

```
<preference prefmod="nc1_p" discmod="adj3_p"
     supmod="m_nn"/>
<preference prefmod="m_nn" discmod="m_an2q"
     prevmod="copv_n" cat="todo_p"/>
```

[5]Um adjectivo é do tipo 3, se puder ser núcleo de um sintágma nominal nuclear e não se puder encontrar à esquerda de um núcleo nominal que qualifique [Hagège, 2000]. Exemplo: *portuguesa*.

3 Algoritmo

O núcleo da análise de superfície é composto por duas funções, que são responsáveis por identificar os modelos presentes na entrada. O algoritmo que cada uma delas implementa é descrito nos algoritmos 1 e 2. A função FIND-CHUNKS é responsável por encontrar todos os possíveis modelos do tipo T, para a posição w do segmento em análise. GET-SIBLING-SEQUENCES encontra as sequências de modelos (ou *chunks*) que podem ocorrer a partir da posição w do segmento, dentro do modelo S e após a ocorrência do modelo L. Uma terceira função, cujo algoritmo se encontra descrito no algoritmo 3, é responsável por encontrar uma sequência de modelos para um conjunto de modelos de partida. Estas três funções funcionam recursivamente de forma cruzada, isto é, as duas primeiras chamam a última, que por sua vez chama as duas primeiras recursivamente.

Alg. 1 FIND-CHUNKS(T, w)

```
01. if Repository has chunks of type T for word w then
02.   return success
03. if some category of word w matches T then
04.   Repository.add( new Chunk(T,w) )
05.   return success
06. if T for w is a bad case according to repository then
07.   return not found
08. next_models ← ∅
09. for each category c of word w do
10.   next_models.add getNextChildModels(T, c)
11. Sequences ←CALC-SEQUENCES(T, w, next_models)
12. if Sequences is not empty then
13.   for each sequence s in Sequences do
14.     if LS permits s to close T then
15.       Repository.add( new chunk(T,w,s) )
16.   return success
17. return not found
```

Alg. 2 GET-SIBLING-SEQUENCES(L,S,w)

```
01. if Repository knows sibling sequences for w, L, S then
02.   return
03. next_models←Ls.getNextSiblingModels(L, S)
04. Sequences←CALC-SEQUENCES(S, w, next_models)
05. return Sequences
```

No que diz respeito ao algoritmo FIND-CHUNKS, as primeiras sete linhas verificam se o resultado pode ser imediato, o que poderá acontecer, se o cálculo já tiver sido anteriormente efectuado, ou se uma das categorias da palavra corresponde ao tipo de modelo pretendido (T). Se a resposta não for imediata, serão calculados os modelos candidatos ao seguimento da análise, com base na

Alg. 3 CALC-SEQUENCES(T, w, *next_models*)

```
01. Founded ← ∅
02. for each model m in next_models do
03.   FIND-CHUNKS(m,w)
04.   Founded ← Founded ∪ getChunks(m,w)
05. SortedChunks ←sortByPreferences(T, w, Founded)
06. Sequences← ∅
07. for each chunk c in SortedChunks do
08.   if Sequences is Empty or c is not discardable then
09.     GET-SIBLING-SEQUENCES(c, T, w + c.length() )
10.     if founded some then
11.       for each founded sequence s of sibling chunks do
12.         s.push_back(c)
13.         Sequences.add(s)
14.     if consider short models or not found a sequence then
15.       Sequences.add( sequence(c) )
16. return Sequences
```

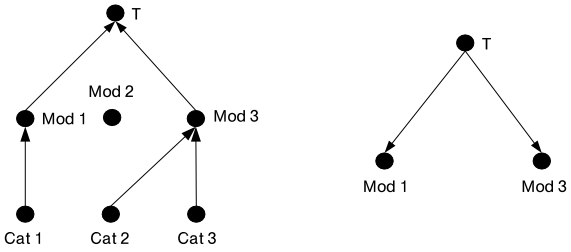

Figura 3: Identificação caminhos descendentes entre um modelo e as categorias de uma palavra.

procura de todos os caminhos descendentes entre o modelo actual e as classificações morfossintácticas da palavra em análise (linhas 8 a 10). A obtenção de caminhos é realizada pela função *getNextChildModels,* que fazendo uma procura do tipo bottom-up, é ilustrada pela figura *3.* Ao entrar na linha 11 o algoritmo já encontrou todos os caminhos possíveis, desde o modelo T até à palavra, que se podem seguir na análise. Assim, o resultado armazenado em *next_models* corresponderá a todos os modelos que se podem seguir dado o actual estado da análise. No caso ilustrado pela figura *3* o conjunto de modelos obtidos é composto apenas pelos modelos *mod1* e *mod*3. As sequências de modelos que podem ocorrer dentro do modelo T são calculadas na linha 11. Note-se cada uma dessas sequências terá como modelo inicial um dos modelos presentes no conjunto *next_models*, o que se traduz numa pesquisa orientada. Finalmente, o algoritmo percorre cada uma das sequências de modelos encontradas, verifica se a sequência pode fechar o modelo actual (T), e no caso de o poder fazer, constrói um novo modelo do tipo T composto por essa sequência.

O algoritmo GET-SIBLING-SEQUENCES, tal como o anterior, começa por verificar se o cálculo pretendido já foi anteriormente efectuado. Se é a primeira vez que se realiza o cálculo, seleccionam-se todos os modelos candidatos ao seguimento da análise, em função do seu estado actual. Finalmente, calculam-se e devolvem-se as sequências de modelos que podem ocorrer dentro do modelo T e que são iniciadas por algum dos modelos presentes em *next_models.*

A função CALC-SEQUENCES encontra todas as sequências de modelos que podem ocorrer dentro do modelo T e que são iniciadas por algum dos elementos de *next_models.* A primeira fase consiste em refinar o conjunto de modelos possíveis, isto é, em obter todas as estruturas possíveis (chunks) para cada um dos elementos presentes em *next_models,* recorrendo à função FIND-CHUNKS (linhas 1 a 4). Os resultados obtidos são ordenados, em função das preferências existentes na gramática, de forma a que numa fase posterior se possam restringir (linha 5). Calculado o primeiro elemento de cada uma das sequências possíveis, a segunda fase consiste em encontrar os modelos que podem seguir esse elemento, tarefa que é feita recorrendo à função GET-SIBLING-SEQUENCES (linhas 6 a 15). Note-se que o algoritmo permite aplicar o princípio dos modelos mais longos à análise (ver secção 3.1).

A obtenção do resultado de uma análise é feita em duas fases, cada uma delas obedecendo a um conjunto de opções. A primeira fase, consiste em produzir um conjunto de resultados intermédios, e corresponde à análise propriamente dita. A segunda corresponde a extrair os resultados pretendidos em função dos resultados provisórios, previamente calculados. A figura 4 mostra o modo de funcionamento.

3.1 Opções de análise

A utilização de resultados intermédios, entre a fase de análise e a fase de extracção de resultados, resulta da estrutura linguística a analisar poder ser independente da estrutura linguística a extrair. Esta arquitectura permite, por exemplo, identificar frases num texto, mas extrair ape-

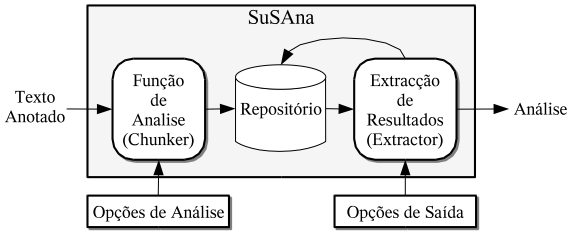

Figura 4: SuSAna - arquitectura interna

nas os sintagmas nominais dessas frases. Seguidamente serão descritas algumas das opções que se podem utilizar na fase de análise.

Elemento de topo O elemento de topo corresponde à estrutura linguística que se pretende identificar. O seu valor por omissão é definido na gramática, mas é possível alterar o seu valor em tempo de execução.

Salto de palavras Por omissão, cada segmento que se analisa deve corresponder exactamente à estrutura linguística que se pretende analisar, contudo o módulo de análise permite também desprezar as unidades lexicais que não consegue tratar, no início ou no fim desse segmento. Com esta opção, pretende-se obter a estrutura linguística de maior tamanho, presente no segmento. Esta opção é aplicável quando a demarcação de segmentos não é fiável.

Múltiplas estruturas linguísticas Por omissão, cada segmento deve conter uma só estrutura linguística, contudo é possível analisar segmentos com múltiplas estruturas. Esta funcionalidade permite, por exemplo, identificar as frases de um parágrafo. Em conjunto com a opção anterior, permite extrair múltiplas estruturas que podem estar ou não seguidas. A utilização desta opção aumenta consideravelmente o tempo de execução.

O algoritmo 4 considera as opções anteriores, para realizar a análise.

Estruturas incompletas Uma das tarefas que se pode realizar com a SuSAna, consiste em analisar uma estrutura linguística incompleta, e prever o conjunto de modelos admissíveis para a palavra seguinte. Esta funcionalidade está a ser utiliza para limitar a procura de palavras, num sistema de auxílio à escrita de poemas.

Alg. 4 DO-ANALYSE(*S*, *top*, *skips*, *multiple*)

```
01. Repository.Clear()
02. for each segment hypothesis h in S do
03.   if not skips and multiple then
04.     while pos < h.nwords() do
05.       FIND-CHUNKS(top, pos)
06.       pos ←pos+max( getMaxChunkLen(sword, top ), 1)
07.   if skips and not multiple then
08.     repeat
09.       FIND-CHUNKS(top, pos)
10.     until found chunks or pos = end of hypothesis
11.   if skips and multiple then
12.     for each position pos of h do
13.       FIND-CHUNKS(top, pos)
14.   if not skips and not multiple then
15.       FIND-CHUNKS(top, 0)
```

Restrição de análises A restrição do número de hipóteses para uma dada análise, torna mais fácil o seu tratamento e utilização. Uma das formas de o fazer consiste na utilização das preferências, especificadas pela gramática. A SuSAna permite também ter em consideração o princípio dos modelos mais longos [Abney, 1996, Hagège, 2000], que consiste em desprezar os modelos mais curtos na presença de outros mais compridos, para restringir as suas análises. A utilização de cada uma destas formas de restrição pode ser activada ou desactivada por uma opção. De uma forma geral, a limitação dos resultados é desejável. Nos algoritmos apresentados, a aplicação das preferências é feita, imediatamente após a escolha preliminar dos modelos, com a função *sortByPreferences*. Como os modelos ficam ordenados, a análise prossegue com os modelos mais interessantes sob o ponto de vista de preferências.

3.2 Opções para as saídas

Os resultados intermédios, produzidos na fase de análise, permitem extrair informação diversa e em diferentes formatos.

Estrutura linguística a extrair Por omissão, o resultado da identificação de uma estrutura linguística, é a própria estrutura. Contudo, a SuSAna permite extrair apenas partes dessa estrutura. Esta opção pode ser aplicada por exemplo,

```
<segment>
 <!-- A ainda mais bela rapariga -->
 <hypothesis length="5">
  <analysis weight="0" start="0" length="5">
   <model name="ph" start="0" length="5">
    <model name="m_nn" start="0" length="5">
     <model name="artd_s" start="0" length="1">A</model>
     <model name="m_an1" start="1" length="3">
      <model name="m_advn1" start="1" length="2">
       <model name="adv1" start="1"
        length="1">ainda</model>
       <model name="advcomp" start="2"
        length="1">mais</model>
      </model>
      <model name="adj1_s" start="3"
       length="1">bela</model>
     </model>
     <model name="nc1_s" start="4"
      length="1">rapariga</model>
    </model>
   </model>
  </analysis>
 </hypothesis>
</segment>
```

Figura 5: Resultado (em XML) da análise da frase *"A ainda mais bela rapariga"*.

na extracção de sintagmas nominais de uma frase. Note-se que, no caso de se terem encontrado várias hipóteses para a análise, com a mesma sub-estrutura, essa sub-estrutura só deve ser apresentada uma vez.

Formatos de saída O resultado pode apresentado de diversas formas: toda a análise no formato XML, simples demarcações ou simples contagens. O exemplo seguinte apresenta o resultado da demarcação da frase "A ainda mais bela rapariga", contudo, a figura 5 apresenta a análise desta mesma frase, no formato XML.

```
ph(m_nn(A m_an1(m_advn1(ainda mais) bela) rapariga))
```

4 Implementação

SuSAna foi implementada como uma classe, na linguagem de programação C++, que possibilita o paradigma de programação por objectos. Assim, pode ser incluída e utilizada dentro de qualquer aplicação, através da sua API (*Application Programming Interface*), isto é, dos seus membros públicos. Com base na classe disponibilizada, qualquer aplicação pode fazer uso do processamento sintáctico de superfície.

Para permitir a utilização do analisador remotamente foi também implementado um módulo cliente de RPC, que permite utilizar a SuSAna a partir de qualquer máquina. A utilização de uma plataforma cliente/servidor permite realização de tarefas, em máquinas dedicadas para o efeito, bastando para isso utilizar o cliente adequado a partir de qualquer máquina. Dada a simplicidade do módulo cliente de RPC, a sua instalação e utilização em diferentes plataformas torna-se mais simples. A figura 6 mostra a interacção com o módulo da SuSAna numa plataforma cliente/servidor. O cliente de RPC pode ser uma aplicação *standalone* ou ser um sistema mais complexo de processamento da língua que faça uso do processamento sintáctico.

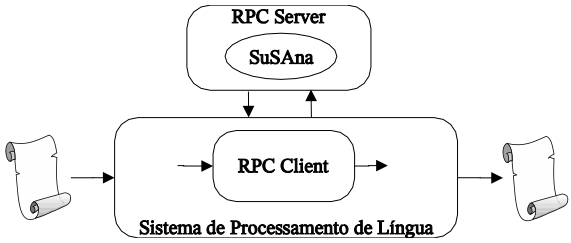

Figura 6: SuSAna - Funcionamento numa arquitectura cliente/servidor

5 Resultados experimentais

Os resultados apresentados de seguida, consistem na identificação da estrutura de sintagmas nucleares presentes num *corpus* com texto não restrito[6]. Começa-se por introduzir a noção de sintagma nuclear, indicaremos as características principais da gramática e do *corpus*, finalmente apresentaremos resultados estatísticos sobre a experiência realizada.

5.1 Sintagma Nuclear

Hagège [Hagège, 2000] define sintagmas nucleares como domínios linguísticos, onde se verificam de-

[6] A expressão *texto não restrito* vem do Inglês *unrestricted text* e corresponde à expressão *texto real*. Artigos de um jornal ou a transcrição de um *corpus* de fala, são exemplos de texto real.

terminadas propriedades e, que se podem definir entre a unidade lexical e o sintagma tradicional. Estes domínios linguísticos são de alguma forma um meio termo entre os conceitos do texto e os sintagmas tradicionais, que se passam a designar apenas por sintagmas. Normalmente os sintagmas nucleares incluem-se dentro dos sintagmas tradicionais e por vezes podem confundir-se com eles. O exemplo seguinte faz a divisão de uma frase, que neste caso corresponde a um sintagma nominal, em sintagmas nucleares.

(Esta bela amiga) (portuguesa) (do Pedro)

A primeira demarcação corresponde a um *sintagma nominal nuclear*, a segunda a um *sintagma adjectival nuclear* e a terceira a um *sintagma proposicional nuclear*. Cada um desses sintagmas nucleares organiza-se em torno de um núcleo ou cabeça lexical. No exemplo anterior os núcleos dos sintagmas nucleares serão respectivamente, o nome *amiga*, o adjectivo que se encontra sozinho e o nome próprio *Pedro*.

O motivo que leva à utilização dos sintagmas nucleares prende-se com a existência de propriedades particulares no interior destes domínios sintácticos que são muito mais fáceis de descrever e ligar do que os sintagmas. A análise de uma frase pode consistir, por um lado na delimitação de sintagmas nucleares aí presentes, e por outro em colocar em evidência as relações que existem entre esses sintagmas nucleares.

Os sintagmas nucleares utilizados são em tudo semelhantes aos sintagmas minimais utilizados por Giguet [Giguet, 1998] e também aos chunks de Abney [Abney, 1991]. Os sintagmas nucleares apresentam uma característica que os distinguem dos sintagmas minimais e dos *chunks*, que é o facto de que um sintagma nuclear além de ser constituído por categorias pode também conter outros sintagmas nucleares. Este é o caso por exemplo do sintagma nominal nuclear que pode conter um sintagma adjectival nuclear que por sua vez contém um sintagma adverbial nuclear, como é o caso de:

(A ((ainda mais) bela) rapariga)

Os parêntesis exteriores, do exemplo anterior, delimitam o sintagma nominal nuclear, os parêntesis no primeiro nível correspondem ao sintagma adjectival nuclear e no segundo nível ao sintagma adverbial nuclear. De sublinhar que Abney e Giguet utilizaram os seus equivalentes aos sintagmas nucleares para a identificação das unidades prosódicas da frase.

Steven Abney indica que uma gramática livre de contextos é suficiente para definir a estrutura dos *chunks* [Abney, 1991]. As restrições impostas aos sintagmas nucleares utilizados em [Hagège, 2000] são bastante mais complexas do que as que são permitidas por uma gramática regular. Como resultado disto, é possível escrever uma descrição linguística, que permita identificar chunks com o módulo descrito neste artigo.

5.2 Características da gramática

Nos testes realizados, a gramática era composta pelo conjunto de regras apresentado na tabela seguinte:

Blocos	Remplace	Preferências	Total
269	28	199	496

As regras descrevem o comportamento de 151 modelos diferentes, 116 dos quais sendo terminais e os restantes não terminais.

5.3 Corpus

Os textos considerados neste teste são constituídos por excertos de livros com aproximadamente 29100 palavras. A segmentação dos textos em frases, revelou 2608 segmentos com uma média de 10.96 palavras por segmento. A classificação morfossintáctica produziu um total de 38574 etiquetas o que corresponde a uma ambiguidade média de 1.326.

5.4 Resultados da análise

Os testes foram realizados com e sem ambiguidade morfossintáctica. Informação com ambiguidade morfossintáctica traduz-se num tempo de processamento mais elevado e num conjunto maior de possibilidades de análise, por outro

lado a desambiguação morfossintáctica automática pode aumentar o número de erros. A tabela seguinte apresenta o tempo de processamento[7] dispendido em cada um dos testes.

Tempo (segs.)	Com amb.	Sem amb.
Total	333.99	212.73
por frase	0.13	0.08

A análise de uma frase pode ser constituída por mais do que uma possibilidade. Assim, se por um lado é desejável obter o menor conjunto de análises não vazias, por outro é importante que o número de hipóteses se mantenha baixo, eliminando as hipóteses menos desejáveis, de forma a permitir um tratamento mais fácil em fases posteriores. Nesse sentido, o número de hipóteses de cada análise foi considerado como uma das medidas de avaliação. A tabela seguinte mostra o número de hipóteses geradas em média, para o conjunto de frases. Apresentam-se resultados produzidos com base tanto em informação morfossintáctica ambígua, como desambiguada.

Hip.	0	1	2	3 - 4	> 4
Ambig.	14.9%	47.1%	23.1%	8.2%	6.7%
Desamb.	23.7%	71.7%	2.3%	2.1%	0.2%

Relativamente à informação com ambiguidade, os resultados mostram que a grande maioria das análises consiste numa ou duas possibilidades. A informação desambiguada, por seu lado, origina um maior número de análises vazias, devido aos erros introduzidos pelo desambiguador, e reduz bastante o número de possibilidades de análise.

6 Conclusões

Foi apresentado um analisador sintáctico de superfície que permite reconhecer tanto as fronteiras, como a estrutura interna e a categoria sintáctica de sintagmas, tanto simples como mais complexos, como é o caso dos sintagmas nucleares. A abordagem seguida no seu desenvolvimento e o algoritmo utilizado, permitem a sua integração em sistemas de processamento de linguagem natural e uma utilização eficiente.

[7]Testes realizados em linux, num processador Intel Pentium III 800 Mhz com 1599.07 bogomips, 512 Mbytes de RAM e uma cache de 256Kb.

Referências

[Abney, 1992] Abney, S. (1992). Prosodic structure, performance structure and phrase structure. In *Proceedings, Speech and Natural Language Workshop*, pages 425–428. Morgan Kaufmann Publishers, San Mateo, CA.

[Abney, 1991] Abney, S. P. (1991). Parsing by chunks. In Berwick, R. C., Abney, S. P., and Tenny, C., editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 257–278. Kluwer Academic Publishers, Dordrecht.

[Abney, 1996] Abney, S. P. (1996). Part-of-speech tagging and partial parsing. In Church, K., Young, S., and Bloothooft, G., editors, *Corpus-Based Methods in Language and Speech*, chapter Dordrecht. Kluwer Academic Publishers.

[Fach, 1999] Fach, M. (1999). A comparison between syntactic and prosodic phrasing. In *Proceedings of Eurospeech 1999*, volume 1, pages 527–530, Budapest.

[Giguet, 1998] Giguet, E. (1998). *Méthode pour l'analyse automatique de structures formelles sur documents multilingues.* PhD thesis, Université de Caen Basse-Normandie.

[Hagège, 2000] Hagège, C. (2000). *Analyse Syntaxique automatique du portugais.* PhD thesis, Laboratoire de Recherche sur le Language, Université Blaise Pascal - Clermont-Ferrand, GRIL.

[Ribeiro et al., 2002] Ribeiro, R., Oliveira, L., and Trancoso, I. (2002). Morphossyntactic disambiguation for tts systems. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1427–1431, Las Palmas, Spain. ELRA.