Uma proposta de autômato adaptativo para reconhecimento de anáforas pronominais segundo algoritmo de Mitkov

Djalma Padovani

***Resumo*— A Extração de Informações é uma área do processamento da linguagem natural que apresenta diversos desafios e um deles é a identificação de termos anafóricos ou co-referentes. Existem diversas propostas para resolução do problema e uma delas, chamada algoritmo de Mitkov, poderia ser implementada de uma forma mais eficiente por meio de tecnologias adaptativas, trazendo uma contribuição importante para esta área, pois existem poucos trabalhos propondo o uso de tecnologias adaptativas para o processamento da linguagem natural. Este trabalho propõe um autômato adaptativo que implementa o algoritmo de reconhecimento de anáforas pronominais de Mitkov.**

Palavras Chave*— *Autômatos Adaptativos, Processamento de Linguagem Natural, Reconhecedores Sintáticos, Gramáticas Livre de Contexto

I. INTRODUÇÃO

A Extração de Informações é uma área do processamento da linguagem natural que apresenta diversos desafios e um deles é a identificação de termos anafóricos ou co-referentes. Existem diversas propostas para resolução do problema e uma delas, chamada algoritmo de Mitkov, poderia ser implementada de uma forma mais eficiente por meio de tecnologias adaptativas, trazendo uma contribuição importante para esta área, pois existem poucos trabalhos propondo o uso de tecnologias adaptativas para o processamento da linguagem natural. Este trabalho propõe um autômato adaptativo que implementa o algoritmo de reconhecimento de anáforas pronominais de Mitkov.

VI. O PROBLEMA DA ANÁFORA

Anáfora é um fenômeno lingüístico no qual uma informação anteriormente introduzida é referenciada posteriormente em outra frase [1]. Por exemplo na frase "O João ama a Maria, mas ela não o ama", o sintagma nominal "O João" da primeira frase é referenciado na segunda frase através do pronome "o" (mesmo número, gênero e pessoa). O mesmo acontece com o sintagma "a Maria" e o pronome pessoal "ela" (mesmo gênero e número).

Formalmente, a expressão lingüística que introduz a anáfora na frase é denominada expressão anafórica. A informação previamente introduzida, que deve ser ligada à expressão anafórica, é denominada antecedente e o processo pelo qual é identificado o antecedente de uma expressão anafórica é denominado resolução anafórica ou resolução de anáforas.

As expressões anafóricas podem ser de diferentes tipos, tais como [2]:

- Pronominais: são aquelas nas quais o termo anafórico é um pronome. Por exemplo, na frase "O menino leu o livro, mas ele não gostou", o pronome "ele" é uma anáfora pronominal .
- Definidas: são compostas por um substantivo antecedido por um artigo definido. Por exemplo, na frase "Comprei uma casa. A casa fica longe daqui", o artigo "a" e o substantivo "casa" constituem uma anáfora definida.
- Demonstrativas: são compostas por um substantivo antecedido de um pronome demonstrativo. Por exemplo na frase "Comprei uma casa. Esta casa será sempre minha", o pronome "esta" e o substantivo "casa" constituem uma anáfora demonstrativa.

Em [3] são apresentadas diferentes técnicas de resolução de anáforas pronominais: algoritmo de Hobbs, algoritmo de Dagan e Itai, algoritmo de Lapin e Leass, o algoritmo de Palomar e o algoritmo de Mitkov.

O algoritmo de Mitkov tem por objetivo resolver anáforas pronominais cujos antecedentes são sintagmas nominais. A abordagem usada por Mitkov evita análises semânticas e sintáticas complexas e utiliza como método fundamental de resolução uma lista de heurísticas denominadas indicadores de antecedentes.

A abordagem proposta por Mitkov realiza os seguintes passos [3]:

- Examina a sentença corrente e as duas sentenças precedentes (se existirem) à anáfora em busca de sintagmas nominais (SN).
- Dentre os SNs encontrados seleciona somente aqueles que concordam em gênero e número com a anáfora e os agrupa em um conjunto de candidatos a antecedentes potenciais.
- Os SNs deste conjunto são pontuados por indicadores de antecedentes e posteriormente é realizada a soma dos pontos.

O SN escolhido como antecedente da anáfora será aquele com maior soma resultante das pontuações destes indicadores.

Os indicadores de antecedentes podem ser:

- PSN – Primeiro Sintagma Nominal. O primeiro sintagma nominal de cada sentença recebe 1 ponto.
- VI – Verbos Indicativos. Os sintagmas nominais seguidos de um verbo pertencente a um conjunto pré-definido (de maior saliência) recebem 1 ponto.
- RL – Reiteração Lexical. Sintagmas nominais que se repetem 2 ou mais vezes na mesma sentença na qual aparece o pronome recebem 2 pontos e os que se repetem apenas 1 vez recebem 1 ponto.
- PSTS – Preferência por SN em Títulos de Seção. Se

o sintagma nominal aparecer no título da seção onde o pronome aparece, então ele recebe 1 ponto.

- PC – Padrões de Colocação. Sintagmas nominais que seguem os mesmos padrões dos pronomes anafóricos podem ser candidatos a antecedentes, recebendo 2 pontos. Por exemplo, <Verbo>+<SN>... <Verbo><Pronome>.
- RI – Referência Imediata. Sintagmas nominais do tipo <V1>+<SN>...<V2>+<SN>...<Vn>+<SN>, onde V1, V2 e Vn são verbos, recebem 2 pontos.
- IS - Instruções Seqüenciais. O sintagmas nominais SN1 que aparecem no padrão <Para>+ <V1>+SN1>...<V2>+<SN2>...<V3>+<SN3>, onde V1, V2 e Vn são verbos, recebem 2 pontos.
- TP - Termo Preferencial. Os sintagmas nominais classificados como representativos do gênero textual recebem 1 ponto.
- SNI - Sintagma Nominal Indefinido. Os sintagmas nominais compostos por artigos ou pronomes indefinidos são punidos e recebem 1 ponto negativo.
- SNP - Sintagma Nominal Preposicionado. Os sintagmas nominais compostos por preposições também são punidos e recebem 1 ponto negativo.
- DR - Distância Referencial. Sintagmas nominais podem ser promovidos de acordo com a distância entre eles e a anáfora. Sintagmas nominais presentes na mesma sentença da anáfora recebem 2 pontos; sintagmas nominais presentes na sentença anterior à da anáfora recebem 1 ponto e sintagmas nominais presentes em 2 sentenças anteriores à anáfora recebem 0 pontos.

De maneira geral, as técnicas de Extração de Informações usam heurísticas e expressões regulares para a seleção de elementos semânticos do texto tais como expressões anafóricas[4][5]. Autômatos adaptativos poderiam ser usados de uma forma mais eficiente pois dispõem de recursos para a implementação de regras semânticas durante o processo de reconhecimento sintático.

II. AUTÔMATOS ADAPTATIVOS

Um autômato adaptativo é uma máquina de estados, inicialmente fixa, à qual são impostas sucessivas alterações durante a sua operação, resultantes da aplicação de ações adaptativas associadas às regras de transições realizadas pelo autômato. Desta maneira, estados e transições podem ser eliminados ou incorporados ao autômato em decorrência de cada um dos passos executados durante a análise da entrada [6].

O autômato adaptativo tem sua estrutura baseada no autômato de pilha estruturado, acrescido de ações adaptativas associadas às regras de transição. As ações adaptativas correspondem às chamadas de funções adaptativas e estas, por sua vez, são compostas de ações adaptativas elementares, declarações de variáveis e de geradores, e eventuais chamadas de funções adaptativas anteriores e posteriores à execução das ações adaptativas elementares [7].

O reconhecimento de uma cadeia ω pelo autômato adaptativo é caracterizado pela seqüência de reconhecimentos parciais de sucessivas sub-cadeias αi da cadeia ω pelas correspondentes máquinas de estados Ei ($0 \leq i \leq n$). Para reconhecer a cadeia ω o autômato terá percorrido, uma trajetória de reconhecimento correspondente à seqüência (E0,α0), (E1,α1), ... , (En,αn) , com $\omega = \alpha 0\ \alpha 1 \ldots\ \alpha n$, onde (Ei,αi) representa o consumo da sub-cadeia αi pela máquina de estados Ei [6].

III. MODELO PROPOSTO

O autômato adaptativo pode ser usado para implementar o algoritmo de Litkov de uma maneira eficiente pois, à exceção das chamadas e dos retornos das funções adaptativas, ele se comporta como um autômato finito.

O projeto de tal autômato deve levar em conta as características próprias do algoritmo de Mitkov. Em primeiro lugar, o algoritmo identifica o sintagma candidato por meio de pontuações obtidas com a aplicação de heurísticas, além de estados de aceitação. O autômato precisa analisar a cadeia, identificar os sintagmas nominais e pontuá-los de acordo com as regras do algoritmo, repetindo o processo até encontrar um pronome ou a cadeia terminar. Quando o autômato encontrar um pronome, ele precisa recuperar o sintagma nominal de maior pontuação e identificá-lo como candidato a antecedente.

Outra característica importante do projeto é a necessidade de ordenação dos sintagmas de acordo com a pontuação obtida na classificação. O autômato deve incorporar mecanismos de ordenação que facilitem a identificação do sintagma candidato ao final da análise da entrada, evitando, com isso, perda de desempenho com novas buscas. Uma lista encadeada ponderada pelos pontos acumulados, por exemplo, pode ser pensada como uma maneira mais rápida de identificar o sintagma candidato, sem que haja necessidade de retornar o processamento.

Por outro lado, é necessário preparar o léxico para que possa ser usado pelo autômato, identificando os *tokens* que compõem o texto e associando-os aos componentes morfológicos usados pelo algoritmo de Litkov: sintagmas nominais, sintagmas nominais indeterminados, sintagmas nominais preposicionados e sintagmas verbais. Se o *token* analisado é um substantivo, é necessário também identificar se ele está presente no título, se é um termo preferencial, a sentença em que aparece, e seu gênero (masculino ou feminino) e número (singular ou plural). Já se o *token* analisado é um verbo, é necessário identificar se ele faz parte do conjunto de verbos indicativos. Por fim, é necessário criar uma gramática que possa ser usada pelo autômato para reconhecer as sentenças da linguagem analisada. As seções seguintes descrevem os elementos usados no projeto do autômato.

A. Gramática e Léxico

A Fig. 1 apresenta uma gramática simplificada proposta para identificar os elementos usados pelo algoritmo de Litkov. (Uma gramática completa da Língua Portuguesa do Brasil pode ser encontrada em [8]).

A gramática é composta do conjunto de não-terminais

(S, SN, SV, Sind, Sprep, ind, prep, pro, N , V, D), onde:

S – Raiz
SN – Sintagma Nominal
SV – Sintagma Verbal
Sind – Sintagma indefinido
Sprep – Sintagma preposicionado
ind – elemento indefinido do SN
prep – elemento preposicional do SN
pro – elemento pronominal
N – núcleo do sintagma nominal
V – núcleo do sintagma verbal
D – demais elementos da Língua Portuguesa

S ➔ SN SV
SN ➔ Sind | Sprep | N | D
SV ➔ V SN | V
Sind ➔ ind N
Sprep ➔ prep N
ind ➔ ∀ α ϵ (artigos indefinidos)
prep ➔ ∀ β ϵ (preposições|contrações)
pro ➔ ∀ γ ϵ (pronomes)
N ➔ ∀ π ϵ (substantivos)
V➔ ∀ ξ ϵ (verbos)
D ➔ ∀ δ ℰ (artigos indefinidos | preposições |contrações | pronomes| substantivos| verbos)

Fig. 1. Gramática proposta.

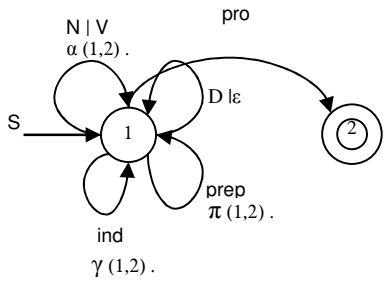
e dos terminais α ϵ (artigos indefinidos), β ϵ (preposições | contrações), π ϵ (pronomes), ξ ϵ (substantivos) , φ ϵ (verbos) e δ ℰ (artigos indefinidos | preposições | contrações | pronomes | substantivos | verbos). O espaçamento simples é usado como separador de *tokens* e o ponto é usado como separador de sentenças.

No caso de substantivos, o léxico é acrescido com a indicação do gênero, número, termo preferencial e presença no título da seção. Por exemplo, o substantivo "flúor" recebe a classificação <N\masc\sing\tp\psts>. No caso de verbos, o léxico é acrescido da identificação de verbo indicativo. Por exemplo, o verbo "analisar" recebe a classificação <V\VerboC>.

B. Autômato Adaptativo

A Fig. 2 apresenta o autômato adaptativo que implementa o algoritmo de Mitkov. São considerados os indicadores de antecedentes PSN, VI, RL, PSTS, TP, SNI, SNP e DR.

O autômato proposto lê cada átomo de entrada S até encontrar o primeiro espaçamento, montando um *token*. Em seguida, ele procura pelo não-terminal do *token* na lista de classificação léxica e, caso o encontre, realiza a transição para o estado correspondente. Caso não encontre o não terminal, ele realiza uma transição em vazio e continua a leitura da cadeia.



Produções Iniciais:

{(1, D |ε) :⟶ 1 }
{(1, N|V) , α (1,2) :⟶ 1}
{(1, ind), γ (1,2) :⟶ 1}
{(1, prep), π (1,2) :⟶ 1}
{(1, pro) :⟶ 2 }

Funções Adaptativas:

α (j,k): { o* :
- [(j, N|V), α (j,k) :⟶ j]
- [(j, ind), γ (j,k) :⟶ j]
- [(j, prep), π (j,k) :⟶ j]
+ [(o, '.') :⟶ j]
+ [(j, N) :⟶ o, □ (o,k)]
+ [(j, V) :⟶ o, □ (o,k)]}

□ (j,k): { t*u*:
+ [(j, ind) :⟶ t]
+ [(j, V|N|D| ε) :⟶ j]
+ [(j, pro) :⟶ k]
+ [(t, D| ε) :⟶ t]
+ [(t, N) :⟶ j]
+ [(j, prep) :⟶ u]
+ [(u, D| ε) :⟶ u]
+ [(u, N) :⟶ j]}

γ (j,l): { p*r* :
- [(j, N|V), α (j,l) :⟶ j]
- [(j, ind), γ (j,l) :⟶ j]
- [(j, prep), π (j,l) :⟶ j]
+ [(j, ind) :⟶ p]
+ [(p, D| ε) :⟶ p]
+ [(p, N) :⟶ r, δ (j,r,l,p)]}

δ (j, r,l,p): {v*
+
[(r,'.'): ⟶ j]
+ [(r, pro) :⟶ l]
+ [(r, ind) :⟶ p]
+ [(r, V|N|D| ε) :⟶ r]
+ [(r,prep) :⟶ v]
+ [(v, D| ε) :⟶ v]
+ [(v, N) :⟶ r]
+ [(j, V) :⟶ r]
+ [(j,prep): ⟶ v]}

π (j,l): { q*s* :
- [(j, N|V), α (j,l) :⟶ j]
- [(j, ind), γ (j,l) :⟶ j]
- [(j, prep), π (j,l) :⟶ j]
+ [(j, prep) :⟶ q]
+ [(q, D| ε) :⟶ q]
+ [(q, N) :⟶ s, ξ (j, s,l,q)]}

ξ (j, s,l,q): {x*
+ [(s, '.') :⟶ j]
+ [(s, pro) :⟶ l]
+ [(s, prep) :⟶ q]
+ [(s, V|N|D| ε) :⟶ s]
+ [(s, ind) :⟶ x]
+ [(x, D| ε) :⟶ x]
+ [(x, N) :⟶ s]
+ [(j, V) :⟶ s]
+ [(j,ind): ⟶ x]}

Fig. 2. Autômato adaptativo para o algoritmo de Mitkov

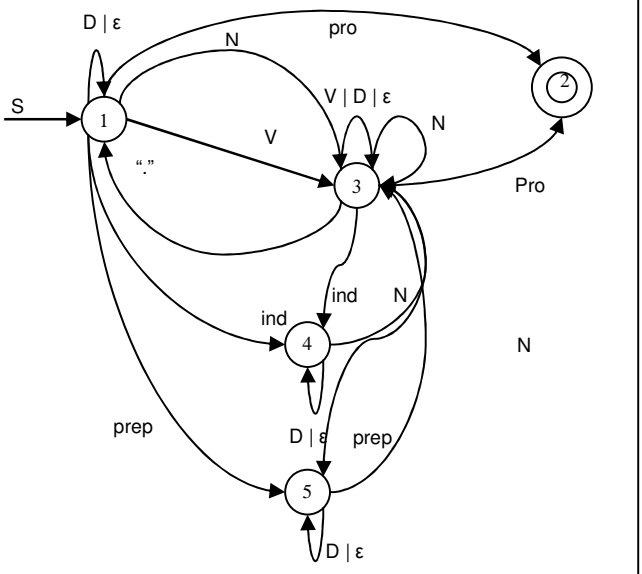
O processo se repete até que o autômato encontre um pronome ou finalize a leitura da cadeia. O autômato é construído dinamicamente, de acordo com o não-terminal identificado no estado 1.

A função adaptativa α (j,k) é chamada pelo autômato no estado 1 antes de processar um *token* N ou V, criando o estado 3 e as produções que levam o autômato do estado 1 ao novo estado e deste ao estado 1. Após a criação do

novo estado, o autômato chama a função (j,k), criando o estado 4 e o estado 5, e as produções que interligam o estado 3 aos estados 2, 3, 4 e 5.

A função adaptativa γ (j,l) é chamada pelo autômato no estado 1, antes de processar o *token* ind, criando o estado 4 e o estado 3, e as produções que levam o autômato do estado 1 ao estado 4. Em seguida, o autômato chama a função δ (j, r,l,p), criando o estado 5 e as produções que interligam o estado 3 aos estados 1, 2, 3, 4 e 5.

A função adaptativa π (j,l) é chamada pelo autômato no estado 1, antes de processar o *token* prep, criando o estado 5 e o estado 3, e as produções que levam o autômato do estado 1 ao estado 5. Em seguida, o autômato chama a função ξ (j,s,l,q), criando o estado 4 e as produções que interligam o estado 3 aos estados 1, 2, 3, 4 e 5.



Produções Finais:

(1, D |ε) :⟶ 1
(1, V) :⟶ 3
(1, N) :⟶ 3
(1, ind) :⟶ 4
(1, prep) :⟶ 5
(1, pro) :⟶ 2
(3, pro) :⟶ 2
(3, V| D |N | ε) :⟶ 3
(3, ".") :⟶ 1
(3, ind) :⟶ 4
(4, N) :⟶ 3
(4, D | ε) :⟶ 4
(5, N) :⟶ 3
(5, D | ε) :⟶ 5

Fig. 3. Situação final do autômato.

Se o autômato identifica um pronome nos estados 1 ou 3, ele realiza uma transição para o estado de aceitação 2. Nota-se que o autômato estabiliza-se após a criação dos estados 3, 4 e 5 e das correspondentes transições, sendo capaz, a partir daí, de processar qualquer elemento da gramática proposta. A Fig. 3 apresenta a situação final do autômato.

A pontuação dos candidatos a termo antecedente é feita sempre que o autômato encontra um substantivo. Uma lista encadeada ordenada pela pontuação é atualizada e, quando o autômato encontra um pronome, o substantivo mais bem pontuado da lista de mesmo gênero e número do pronome é apresentado como termo candidato. Um exemplo conceitual de implementação é apresentado a seguir.

O algoritmo Identifica_Anáfora lê a cadeia de entrada e calcula os pontos do *token* por meio da rotina Calcula_Pontos. Se o *token* lido é um substantivo, Identifica_Anáfora chama a rotina Insere_Lista e atualiza uma lista ordenada ponderada pelos pontos do *token*. Identifica_Anáfora continua a leitura da cadeia e se em determinado momento o *token* lido é um pronome, Busca_Candidato_Lista é chamada, verificando os *tokens* que possuem o mesmo gênero e número do pronome e calculando a distância referencial entre estes elementos e o pronome. Busca_Candidato_Lista retorna o *token* candidato à instância de Identifica_Anáfora finalizando o processamento.

Identifica_Anáfora

1. [Lê e identifica *token* da cadeia de entrada]
   Termo = Lê_*Token* (cadeia)

   [Identifica termo do tipo D]
   Se Termo(tp) = D
   então Leia_Proximo

   [Identifica nova sentença]
   Se Termo = "."
   então sentença := sentença +1

   Termo(sa) := sentença

   [Identifica SNI e SNP]
   Se Termo(tp) = (ind | prep)
   então Redutor := True

   [Identifica SN]
   Se Termo(tp) = N
   então
   Chama Calcula_Pontos
   Insere_Lista (Termo(vl), Termo(gn), Termo(nm), Termo(sa), Termo(p))
   Se Redutor= True
   então Redutor :=False

   [Identifica Pronomes]
   Se Termo(tp) = pro
   então
   Candidato = Busca_Candidato_Lista ()
   Retorna Candidato

Calcula_Pontos:

1. [Verifica se SN é PSN]
   Se Lista = ε e Termo(tp) = N
   então Termo(p) := Termo(p) + 1

   [Verifica se SN é PSTS]
   Se Termo(tp) = N e N = PSTS
   então Termo(p) := Termo(p) +1

[Verifica se SN é TP]
Se Termo(tp) = N e N = TP
então Termo (p) := Termo (p) +1

[Verifica se SN é SNI]
Se Termo(tp) = N e Redutor = True
então Termo(p) := Termo(p) – 1
Redutor = False

[Verifica se SN é SNP]
Se Termo(tp) = N e Redutor = True
então Termo (p) := Termo (p) – 1
Redutor = False

2. [Verifica se LookAhead é VI]
Se LookAhead (SN) = V e V = VI
então Termo(p) := Termo(p) +1

Insere_Lista:

1. [Se a lista está vazia]
Lista(id) := primeiroID
Lista(vl) := Termo(vl)
Lista(gn) := Termo(gn)
Lista(nm) := Termo(nm)
Lista(sa) := Termo(sa)
Lista(p) := Termo(p)
Lista(lk) := null

2. [Se pontuação do termo for maior que primeiro registro da lista]
Se Termo(p) > Lista(p)
então
ListaI(id) := novo id
ListaI(vl) := Termo(vl)
ListaI(gn) := Termo(gn)
ListaI(nm) := Termo(nm)
ListaI(sa) := Termo(sa)
ListaI(p) := Termo(p)
ListaI(lk) := id do primeiro registro da lista

3. [Se pontuação do termo for menor que primeiro registro da lista]
Repita enquanto Termo(p) <= Lista(p)

Guarda(lk) := Lista(lk)
Lista(lk) := novo id
Termo(lk) := Guarda(lk)

ListaI(id) := novo id
ListaI(vl) := Termo(vl)
ListaI(gn) := Termo(gn)
ListaI(nm) := Termo(nm)
ListaI(sa) := Termo(sa)
ListaI(p) := Termo(p)
ListaI(lk) := Termo(lk)

Busca_Candidato_Lista:

1. [Varre lista em busca do candidato]
Repita enquanto Lista(p) > Candidato(p)]
[Verifica Gênero e Número]
Se Lista(gn) = Termo(gn) e Lista(nm) = Termo(nm)
então
[Verifica DR]
Se Lista(st) = Termo(st)
então Lista (p) := Lista (p) + 2
Senão
Se Lista(st) = Termo(st) – 1
então Lista (p) := Lista (p) + 1

[Identifica Candidato]
Candidato(vl) := Lista(vl)
Candidato(p) := Lista(p)
Retorna Candidato

Sendo:
Cadeia – cadeia analisada
Sentença – número da sentença analisada
Gênero – gênero do termo analisado
Número – número do termo analisado
SN – sintagma nominal
PSN – primeiro sintagma nominal
PSTS – preferência por sintagma em título de seção
TP – termo preferencial
SNP – sintagma preposicionado
SNI – sintagma indefinido
VI – verbo indicativo
LookAhead – função de leitura do termo seguinte
Lista – conjunto de atributos do elemento no qual a lista está posicionada
ListaI – conjunto de atributos do termo que será inserido na lista
Termo – conjunto de atributos que caracterizam o termo analisado
Candidato – conjunto de atributos que caracterizam o termo candidato a antecedente
id – identificador único
tp – tipo do *token*
vl – valor do *token*
gn – gênero
nm – número
sa – sentença
p – pontos
lk – ponteiro para o próximo elemento

VII. EXPERIMENTO

Foi realizado um experimento no qual a implementação conceitual descrita na seção anterior foi codificada. Foi utilizada a linguagem de programação Python[9], pois ela dispõe de recursos de processamento de expressões regulares (*Regular Expression* – RE), processamento de linguagem natural (*Natural Language Tool Kit* – NLTK) e autômatos finitos (*Determinist Finite Automon* – DFA) necessários para a realização do trabalho.

O programa foi testado com o texto 'O flúor fortifica o esmalte, uma espécie de capa protetora dos dentes. Com a difusão de seu uso, outro problema surgiu: a fluorose, o excesso de flúor no organismo. Afinal, a substância não

se encontra apenas na água e cremes dentais: ela também está presente em diversos alimentos' [3]. O objetivo foi identificar o termo ao qual o pronome 'ela' faz referência (substância).

Como a frase apresenta diversos substantivos (flúor, esmalte, espécie, capa, dentes, difusão, uso, problema, fluorose, substância e água), foi possível testar todas as heurísticas do algoritmo implementadas pelo autômato.

Os elementos da frase foram etiquetados manualmente de acordo com suas classificações morfológicas e com identificações usadas pelo algoritmo de Mitkov: verbos indicativos (VI), termos preferenciais (TP) e preferências por SN em títulos de seção (PSTS). Foram identificados substantivos, verbos, preposições, pronomes, contrações e artigos indefinidos. Os demais elementos foram classificados em um único identificador ('D'). Também foram identificados o gênero e o número dos substantivos e do pronome. A Tabela I apresenta os elementos da frase e suas respectivas etiquetas.

TABELA I
ELEMENTOS E ETIQUETAS

Elemento	Etiqueta	Elemento	Etiqueta
O	<D>	Excesso	<NN\\masc\\sing\ \ntp\\npsts>
flúor	<NN\\masc\\sing\ \tp\\psts>	De	<D>
fortifica	<V>	Flúor	<NN\\masc\\sing\ \tp\\psts>
O	<D>	No	<D>
esmalte	<NN\\masc\\sing\ \ntp\\npsts>	Organismo	<NN\\masc\\sing\ \ntp\\npsts>
uma	<ind>	Afinal	<D>
espécie	<NN\\fem\\sing\\ ntp\\npsts>	A	<D>
de	<D>	Substância	<NN\\fem\\sing\\ ntp\\npsts>
capa	<NN\\fem\\sing\\ ntp\\npsts>	Não	<D>
protetora	<D>	se	<D>
dos	<D>	encontra	<V>
dentes	<NN\\masc\\plur\ \ntp\\npsts>	apenas	<D>
Com	<D>	na	<prep>
a	<D>	água	<NN\\fem\\sing\\ ntp\\npsts>
difusão	<NN\\fem\\sing\\ ntp\\npsts>	e	<D>
de	<D>	cremes	<NN\\masc\\plur\\ ntp\\npsts>
seu	<D>	dentais	<D>
uso	<NN\\masc\\sing\ \ntp\\npsts>	ela	<pron\\fem\\sing>
outro	<D>	também	<D>
problema	<NN\\masc\\sing\ \ntp\\npsts>	está	<V>
surgiu	<V>	presente	<D>
a	<D>	em	<D>
fluorose	<NN\\fem\\sing\\ ntp\\npsts>	diversos	<D>
o	<D>	alimentos	<NN\\masc\\plur\\ ntp\\npsts>

O programa é composto de um analisador léxico, que identifica os tokens da frase, um gerador de autômatos, que cria os novos estados e transições, e de um classificador, que classifica os *tokens* de acordo com os critérios de pontuação descritos no algoritmo de Mitkov. A Fig.4 apresenta a situação inicial do autômato, o alfabeto utilizado e a função de transição. Nota-se que as transições não permitidas são identificadas na função de transição com a palavra "None'. Por exemplo, não existe estado para o autômato ir se estiver no estado 1 e ler um token do tipo 'NN' (substantivo).

```
This DFA has 2 states
States: [1, 2]
Alphabet: ['D', 'NN', 'V', 'VI', 'ind', 'prep', 'pron', '.']
Starting state: 1
Accepting states: [2]
Transition function:
        1       2
D       1       None
NN      None    None
V       None    None
VI      None    None
ind     None    None
prep    None    None
pron    2       None
.       None    None
Current state: 1
Currently accepting: False
```

Fig. 4. Situação inicial do autômato.

À medida que o programa lê os *tokens* da cadeia, ele usa o gerador de autômatos adaptativamente para criar novos estados e transições, que são realizadas em seguida. A Fig. 5 ilustra este comportamento passo a passo.

```
estado atual 1 token ['O', 'D']
estado atual 3 token ['fluor', 'NN', 'masc', 'sing', 'tp', 'psts']
estado atual 3 token ['fortifica', 'V']
estado atual 3 token ['o', 'D']
estado atual 3 token ['esmalte', 'NN', 'masc', 'sing', 'ntp', 'npsts']
estado atual 4 token ['uma', 'ind']
estado atual 3 token ['especie', 'NN', 'fem', 'sing', 'ntp', 'npsts']
estado atual 3 token ['de', 'D']
estado atual 3 token ['capa', 'NN', 'fem', 'sing', 'ntp', 'npsts']
estado atual 3 token ['protetora', 'D']
estado atual 3 token ['dos', 'D']
estado atual 3 token ['dentes', 'NN', 'masc', 'plur', 'ntp', 'npsts', '.']
estado atual 3 token ['Com', 'D']
estado atual 3 token ['a', 'D']
estado atual 3 token ['difusao', 'NN', 'fem', 'sing', 'ntp', 'npsts']
estado atual 3 token ['de', 'D']
estado atual 3 token ['seu', 'D']
estado atual 3 token ['uso', 'NN', 'masc', 'sing', 'ntp', 'npsts']
estado atual 3 token ['outro', 'D']
estado atual 3 token ['problema', 'NN', 'masc', 'sing', 'ntp', 'npsts']
estado atual 3 token ['surgiu', 'V']
estado atual 3 token ['a', 'D']
estado atual 3 token ['fluorose', 'NN', 'fem', 'sing', 'ntp', 'npsts']
estado atual 3 token ['o', 'D']
estado atual 3 token ['excesso', 'NN', 'masc', 'sing', 'ntp', 'npsts']
estado atual 3 token ['de', 'D']
estado atual 3 token ['fluor', 'NN', 'masc', 'sing', 'tp', 'psts']
estado atual 3 token ['no', 'D']
estado atual 3 token ['organismo', 'NN', 'masc', 'sing', 'ntp', 'npsts', '.']
estado atual 3 token ['Afinal', 'D']
estado atual 3 token ['a', 'D']
estado atual 3 token ['substancia', 'NN', 'fem', 'sing', 'ntp', 'npsts']
estado atual 3 token ['nao', 'D']
estado atual 3 token ['se', 'D']
estado atual 3 token ['encontra', 'V']
estado atual 3 token ['apenas', 'D']
estado atual 5 token ['na', 'prep']
estado atual 3 token ['agua', 'NN', 'fem', 'sing', 'ntp', 'npsts']
estado atual 3 token ['e', 'D']
estado atual 3 token ['cremes', 'NN', 'masc', 'plur', 'ntp', 'npsts']
estado atual 3 token ['dentais', 'D']
estado atual 2 token ['ela', 'pron', 'fem', 'sing']
```

Fig. 5. Processamento da frase de teste.

O autômato encontra-se inicialmente no estado 1. Ao

ler o *token* 'O', identificado como 'D', ele permanece no estado 1. Em seguida, ao ler o *token* 'flúor', o autômato cria os estados 3, 4 e 5 e realiza uma transição do estado 1 para o estado 3, no qual ele permanece até ler o *token* 'uma', quando realiza uma transição do estado 3 para o estado 4. Em seguida, ao ler o *token* 'espécie', o autômato volta ao estado 3, no qual permanece até ler a contração 'na', quando realiza uma transição para o estado 5. Em seguida, ao ler o *token* 'água', ele volta para o estado 3 e permanece nele até ler o *token* 'ela', quando realiza uma última transição para o estado de aceitação 2. A Fig.6 apresenta a situação final do autômato, o alfabeto utilizado e a nova função de transição após o processamento da frase. Nota-se que, neste momento, o autômato possui 5 estados e encontra-se no estado de aceitação 2.

```
This DFA has 5 states
States: [1, 2, 3, 4, 5]
Alphabet: ['D', 'NN', 'V', 'VI', 'ind', 'prep', 'pron', '.']
Starting state: 1
Accepting states: [2]
Transition function:
        1       2       3       4       5
D       1       None    3       4       5
NN      3       None    3       3       3
V       3       None    3       None    None
VI      3       None    3       None    None
ind     4       None    4       None    None
prep    5       None    5       None    None
pron    2       None    2       None    None
.       None    None    1       None    None
Current state: 2
Currently accepting: True
```

Fig. 6. Situação final do autômato.

A Fig.7 apresenta o resultado final gerado pelo programa: uma lista de substantivos e suas respectivas pontuações, e uma lista de termos candidatos a antecedente.

```
Lista de Sustantivos e Pontuações
[['fluor', 'NN', 'masc', 'sing', 'tp', 'psts'], [1, 2], 3]
[['cremes', 'NN', 'masc', 'plur', 'ntp', 'npsts'], [2, 1], 1]
[['substancia', 'NN', 'fem', 'sing', 'ntp', 'npsts'], [2, 1], 3]
[['organismo', 'NN', 'masc', 'sing', 'ntp', 'npsts', '.'], [2, 1], 1]
[['excesso', 'NN', 'masc', 'sing', 'ntp', 'npsts'], [1, 1], 1]
[['fluorose', 'NN', 'fem', 'sing', 'ntp', 'npsts'], [1, 1], 2]
[['problema', 'NN', 'masc', 'sing', 'ntp', 'npsts'], [1, 1], 1]
[['uso', 'NN', 'masc', 'sing', 'ntp', 'npsts'], [1, 1], 1]
[['difusao', 'NN', 'fem', 'sing', 'ntp', 'npsts'], [1, 1], 2]
[['dentes', 'NN', 'masc', 'plur', 'ntp', 'npsts', '.'], [1, 1], 1]
[['capa', 'NN', 'fem', 'sing', 'ntp', 'npsts'], [0, 1], 2]
[['esmalte', 'NN', 'masc', 'sing', 'ntp', 'npsts'], [0, 1], 1]
[['agua', 'NN', 'fem', 'sing', 'ntp', 'npsts'], [2, 1], 2]
[['especie', 'NN', 'fem', 'sing', 'ntp', 'npsts'], [0, 1], 1]

Candidatos
[['substancia', 'NN', 'fem', 'sing', 'ntp', 'npsts'], [2, 1], 3]
```

Fig. 7. Resultado final do processamento.

A lista de substantivos está ordenada pela pontuação obtida durante o processamento. Ao encontrar o primeiro pronome, o programa atualiza a pontuação de acordo com os critérios de distância referencial (DR) e reiteração lexical (RL), percorrendo a lista até o momento em que os pontos acrescidos já não influenciem mais no resultado final.

A lista de termos candidatos a antecedente apresenta corretamente o termo 'substância', o mais bem pontuado e de mesmo gênero e número da anáfora analisada, o pronome 'ela'.

VIII. CONCLUSÕES

Este artigo apresentou uma proposta de autômato adaptativo para reconhecimento de anáforas pronominais segundo o algoritmo de Mitkov. O modelo proposto necessita de uma preparação léxica e recomenda um substantivo candidato ao final do processamento. Procurou-se favorecer o desempenho na concepção do mecanismo de funcionamento, evitando-se passos redundantes durante o processamento. A conclusão final foi de que a tecnologia adaptativa pode ser mais eficiente do que as técnicas de Extração de Informações usualmente utilizadas [7][8], pois dispõe de recursos para a implementação de regras semânticas durante o processo de reconhecimento sintático ao invés da técnica de uso de expressões regulares ao final do processamento. Para trabalhos futuros, propõe-se que sejam realizados testes quantitativos, visando avaliar estatisticamente a eficiência do autômato e que seja avaliada a possibilidade de utilizá-lo também para resolução de outros tipos de anáforas que não as pronominais.

REFERÊNCIAS

[1] S.A. Freitas. "Interpretação Automatizada de Textos: Processamento de Anáforas", Tese de Doutorado, orientado por C.S.Menezes e J.G.P.Lopes, Centro Tecnológico da Universidade Federal do Espírito Santo, 2005.

[2] C. Gasperin , R. Goulart , R.Vieira. Uma Ferramenta para Resolução Automática de Co-referência, 2003. Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, Unisinos, São Leopoldo, Brasil. Disponível em: http://www.rodrigo.goulart.nom.br/publicacoes/gasperin2003.pdf.

[3] A.R Chaves. "A Resolução de Anáforas Pronominais da Língua Portuguesa com Base no Algoritmo de Litkov", Dissertação de Mestrado, orientada por L.H.M. Rino, Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, 2007.

[4] r. Grishman, Information Extraction: Techniques and Challenges. Lecture Notes In Computer Science; Vol. 1299. International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, 1997.

[5] E. Riloff and J. Lorenzen. Extraction-based text categorization: Generating Domain Specific role relationships automatically. In Natural Language Information Retrieval, Dordrecht, The Netherlands: Kluwer Academic Publishers, 1999. p. 167-196.

[6] J.J. Neto. "Contribuições à Metodologia de Construção de Compiladores", Tese de Livre Docência, Escola Politécnica da Universidade de São Paulo, 1993.

[7] C.Y.O.Taniwaki e J.J.Neto. "Autômatos Adaptativos no Tratamento Sintático de Linguagem Natural", 2001. Disponível em: <http://www.pcs.usp.br/~lta/artigos/taniwaki_bt2001.pdf>.

[8] C. Luft,. Moderna gramática brasileira, 2000. São Paulo: Globo.

[9] Python. Python Programming Language – Official Website. Disponível em: <http://www.python.org>.

Djalma Padovani nasceu em São Paulo em 1964. Formou-se em administração de empresas pela Faculdade de Economia e Administração da Universidade de São Paulo, em 1987 e obteve o mestrado em engenharia de software pelo Instituto de Pesquisas Tecnológicas de São Paulo - IPT, em 2008. Trabalhou em diversas empresas nas áreas de desenvolvimento de software e tecnologia de informação e atualmente é responsável pela arquitetura tecnológica da Serasa S/A, empresa do grupo Experían.