Universidade de São Paulo - USP
Universidade Federal de São Carlos - UFSCar
Universidade Estadual Paulista - UNESP

Avaliação da Inteligibilidade de Textos para Simplificação Textual

Tiago F. Pereira
Sandra Maria Aluísio

NILC-TR-08-12

Agosto, 2008

Série de Relatórios do Núcleo Interinstitucional de Lingüística Computacional
NILC - ICMC-USP, Caixa Postal 668, 13560-970 São Carlos, SP, Brasil

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este projeto de Iniciação Científica (IC) está inserido no escopo do projeto PorSimples (Simplificação Textual do Português para Inclusão e Acessibilidade Digital), aprovado no âmbito do Edital Microsoft-Fapesp (proc. nro. 2007/54565-8). O objetivo do projeto de IC foi o estudo de características lingüísticas que tornam um texto complexo, tanto do ponto de vista lexical como sintático, a explicitação delas via parser e listas de palavras simples e sua posterior visualização adequada para apoiar um humano na tarefa de anotação de simplificações. Um Editor de Anotação de Simplificação foi projetado, implementado e avaliado por um usuário real que está criando um córpus paralelo de 100 bi-textos compostos de textos originais e textos com simplificação chamada de natural (não guiada) e mais 100 bi-textos compostos de textos de simplificação natural e simplificação chamada de forte, isto é, guiada por um manual de simplificação criado no escopo do PorSimples. Além de possibilitar o conhecimento de ferramentas de processamento de língua natural, este trabalho servirá, futuramente, como base para a construção de um Editor de Simplificação para textos da Web, um dos resultados do PorSimples. Também, possibilitará que o córpus de textos paralelos sendo criado com a ajuda do Editor produza o outro resultado do PorSimples: um Simplificador Estístico para o Português do Brasil.

1 - Introdução

1.1 Contextualização

Este projeto de Iniciação Científica (IC) está inserido no escopo do projeto PorSimples[1] (Simplificação Textual do Português para Inclusão e Acessibilidade Digital), aprovado no âmbito do Edital Microsoft-Fapesp (proc. nro. 2007/54565-8). O projeto PorSimples trata, principalmente, da tarefa de **simplificação textual**, via construção de sistemas para promover o acesso de textos escritos em português por pessoas com níveis de **letramento** rudimentares e básicos, com problemas cognitivos como afasia e dislexia, além de adultos e crianças em fase de aprendizado da leitura e escrita.

Entende-se por **letramento** a capacidade de utilizar a linguagem escrita para informar-se, expressar-se, documentar, planejar e aprender continuadamente, isto é, os usos efetivos da leitura e escrita nas diferentes esferas da vida social (Ribeiro, 2006). Segundo o Indicador Nacional de Alfabetismo Funcional (INAF)[2] que vem sendo calculado anualmente pelo IBOPE desde 2001 para mensurar os níveis de alfabetismo funcional da população brasileira, há quatro níveis de habilidades de leitura/escrita (letramento) na população brasileira:

1) Analfabetismo: Corresponde à condição dos que não conseguem realizar tarefas simples que envolvem decodificação de palavras e frases.
2) Alfabetismo nível rudimentar: Corresponde à capacidade de localizar informações explícitas em textos curtos, um anúncio ou pequena carta.
3) Alfabetismo nível básico: Corresponde à capacidade de localizar informações em textos um pouco mais extensos, podendo realizar pequenas inferências.
4) Alfabetismo nível pleno: Corresponde à capacidade de ler textos longos, orientando-se por subtítulos, localizando mais de uma informação, de acordo com

[1] http://caravelas.icmc.usp.br/wiki/index.php/Principal
[2] http://www.acaoeducativa.org.br

condições estabelecidas, relacionando partes de um texto, comparando dois textos, realizando inferências e sínteses.

A tarefa de **simplificação textual** é definida como qualquer processo que reduza a complexidade léxica ou sintática de um texto enquanto tenta preservar seu significado e informação (Max, 2006; Mapleson, 2006). O objetivo desta tarefa é tornar o texto mais simples de ser compreendido por um leitor humano ou um programa computacional.

O projeto PorSimples prevê a criação de dois tipos de sistemas, um de autoria para auxiliar a edição de textos simplificados por autores que desejam colocar um conteúdo na Web e outro que permite pessoas lerem um texto da Web na sua versão simplificada (funciona como um sistema de pós-processamento de um texto original da Web). Duas abordagens estão sendo exploradas: uma lingüística que faz a simplificação via regras desenvolvidas manualmente com ajuda de informação sintática e também discursiva, e outra que explora a indução destas regras a partir de córpus alinhados de sentenças originais e suas correspondentes simplificadas. Esta última abordagem faz uso de métodos estatísticos que precisam de um grande volume de textos anotados manualmente. Para explorar esta abordagem estatística está sendo criado, por uma lingüista experiente em simplificação textual, um córpus[3] paralelo de 100 bi-textos compostos de textos originais e simplificados de forma natural e mais 100 bi-textos compostos de textos simplificados de forma natural e de forma forte[4]. Esta tarefa de simplificação, a anotação das operações de simplificação utilizadas em cada passo do processo e o alinhamento sentencial entre os bi-textos está sendo apoiada por um Editor de Anotação de Simplificação (http://caravelas.icmc.usp.br/anotador) que foi o tema da IC em questão.

1.2 Objetivos do projeto

O objetivo deste projeto de Iniciação Científica foi o estudo de ferramentas de processamento de língua natural (PLN) – tokenizadores, parsers, alinhadores sentenciais, e

[3] Neste relatório, utilizamos o aportuguesamento da palavra *córpus* (plural *corpora*) tendo a mesma ortografia para o plural e singular.
[4] Estes dois tipos de simplificação serão melhor explicados na Seção 4.

ferramentas de suporte à anotação lingüística – e sua aplicação na construção do Editor de Anotação de Simplificação do PorSimples. Este Editor apóia a tarefa de simplificação e anota (ou registra) todos os passos (ou decisões) realizados pelo usuário que está simplificando textos, via uma interface gráfica na Web. Com a ajuda do Editor pode-se criar córpus anotados de textos originais e simplificados, alinhados sentencialmente. A construção deste córpus é importante no contexto do projeto PorSimples, pois ele será usado na construção de ferramentas de simplificação que seguem a abordagem estatística, isto é, aprendem a tarefa a partir de um grande volume de textos anotados. Este Editor auxilia o anotador na tarefa de simplificação apontando palavras consideradas difíceis, marcadores discursivos pouco usuais ou ambíguos e alertando para estruturas sintáticas que podem ser de difícil assimilação para pessoas de baixa escolaridade, além de sugerir a operação de simplificação de acordo com o manual de simplificação sintática desenvolvido no escopo do PorSimples (Specia, et al., 2007).

1.3 Colaboração

Este projeto de Iniciação Científica foi orientado pela Professora Dra. Sandra Maria Aluísio em conjunto com a aluna de pós-Doutorado do projeto PorSimples, Helena Caseli Medeiros. Colaboraram também com o projeto os alunos de graduação do Bacharelado em Ciência da Computação do ICMC-USP: Carolina Evaristo Scarton (compilação do córpus de 100 textos jornalísticos do Jornal Zero Hora, para trabalho do anotador), Erick Galani Maziero (preparação dos dicionários de palavras simples), Felipe Viana Peres (desenvolvimento do LayOut do Editor) e Willian Watanabe (suporte com a tecnologia Ruby on Rails).

1.4 Organização do relatório

Na Seção 2 será apresentada a revisão bibliográfica deste projeto de Iniciação Cientifica, que tratou de textos e trabalhos da área de PLN e também da área de pesquisa em Interação Usuário Computador (IHC). Na Seção 3 será apresentado o cronograma previsto para este projeto. Na Seção 4 serão apresentados os detalhes da construção do Editor de

Anotação de Simplificação. Na Seção 5 serão apresentados os resultados do projeto. Na Seção 6 serão feitas as considerações finais.

2 - Revisão bibliográfica

A seguir será apresentada uma síntese das tecnologias que foram pesquisadas para o projeto.

2.1 Ruby on Rails

Criado por David Heinemeier Hansson em Julho de 2004, o Ruby on Rails é um meta-framework[5] desenvolvido em Ruby e de código aberto, assim como a linguagem Ruby. Ruby on Rails leva em sua arquitetura o design pattern "MVC" (Model-View-Controller) (Larman, 2007).

O Editor de Anotação de Simplificação foi desenvolvido na plataforma WEB e no ambiente Ruby on Rails[6]. A plataforma WEB foi escolhida devido a sua natureza onipresente que facilita a atualização de versões do editor e pelo acesso onipresente do mesmo devido à própria natureza da internet.

Com o uso da arquitetura MVC, houve vantagens significativas no desenvolvimento do projeto através da separação das camadas de negócio e de visualização permitindo um desenvolvimento mais ágil e limpo. Grande parte das funcionalidades do editor possui processamento assíncrono de páginas WEB com o uso da tecnologia AJAX. A biblioteca AJAX utilizada dentro do projeto é a prototype[7] e ela está embutida dentro do ambiente Ruby on Rails.

2.2 MMAX

O editor de anotação lingüística MMAX (Multi-Modal Annotation in XML) (Müller and Strube, 2001) foi estudado por ser bastante aceito pela comunidade de PLN no Brasil (Salmon-Alt and Vieira, 2002; Vieira et al., 2002b; Vieira et al., 2002a; Vieira et al., 2003), e

[5] Ruby on Rails é um "meta-framework", uma vez que é uma junção de cinco frameworks: Active Record, Action Pack, Action Mailer, Active Suport, Action WebServices.
[6] http://www.rubyonrails.pro.br/
[7] www.prototypejs.org/

para verificar a viabilidade da utilização do mesmo como ferramenta para de construção do córpus paralelo de simplificações.

O MMAX trabalha com o formato *stand-off* de anotação em XML que é organizado de forma separada. Dessa forma, cada arquivo XML que compõe a anotação possui uma função específica. Um texto escrito no formato MMAX é representado de forma hierárquica em uma estrutura de sentenças (*texts.xml*) e palavras (*words.xml*). Segue na Figura 1 e Figura 2 um exemplo de representação, no formato MMAX , do texto "João casou com Maria".

```
<words>
      <word id="word_1">João</word>
      <word id="word_2">casou</word>
      <word id="word_3">com</word>
      <word id="word_4">Maria</word>
</words>
```

Figura 1: Arquivo de palavras (*words*)

```
<text id="text_1">
     <sentence id="sentence_1" span="word_1..word_4"/>
</text>
```

Figura 2: Arquivo de sentenças de um texto (*texts*)

Esta é a estrutura básica para representar um texto no formato MMAX. Para iniciar uma anotação deve-se primeiro definir que tipo de informação se quer anotar. Para isso antes de começar o processo de anotação deve-se criar um arquivo de configuração inicial com os atributos que se quer anotar. Segue na Figura 3 um exemplo de configuração inicial para uma tarefa de anotação. Este exemplo ilustra uma classificação de um conjunto de tokens ou unidades léxicas (word) quanto a sua classificação (Tipo 1 ou Tipo 2) e quanto a sua dificuldade (Fácil / Dificil).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<annotationscheme>
    <attribute id="level_2" name="Classificacao" text="">
        <value id="value_2" name = "Tipo 1"/>
        <value id="value_99" name = "Tipo 2" text="Proper Name"/>
    </attribute>
    <attribute id="level_2" name="Dificuldade" text="">
        <value id="value_2" name = "Facil"/>
        <value id="value_99" name = "Dificil" text="Proper Name"/>
    </attribute>
</annotationscheme>
```

Figura 3: Arquivo de configuração MMAX

Conjuntos de tokens (word) são agrupados em um outro arquivo chamado arquivo de *markables*. Markables são entidades abstratas que agrupam tokens. Sozinhos, os markables não significam nada, mas cada atributo configurado pelo arquivo de configuração citado na Figura 3 se torna um atributo dentro de um markable. Na Figura 4 temos um exemplo de como ficaria um arquivo de markable anotado com a ferramenta MMAX e na Figura 5 temos uma tela da ferramenta MMAX sintetizando o uso da ferramenta.

```
<?xml version="1.0"?>
<markables>
<markable id="markable_1" span="word_2..word_3" classificacao="tipo 1"  dificuldade="dificil" />
</markables>
```

Figura 4: Arquivo de markables

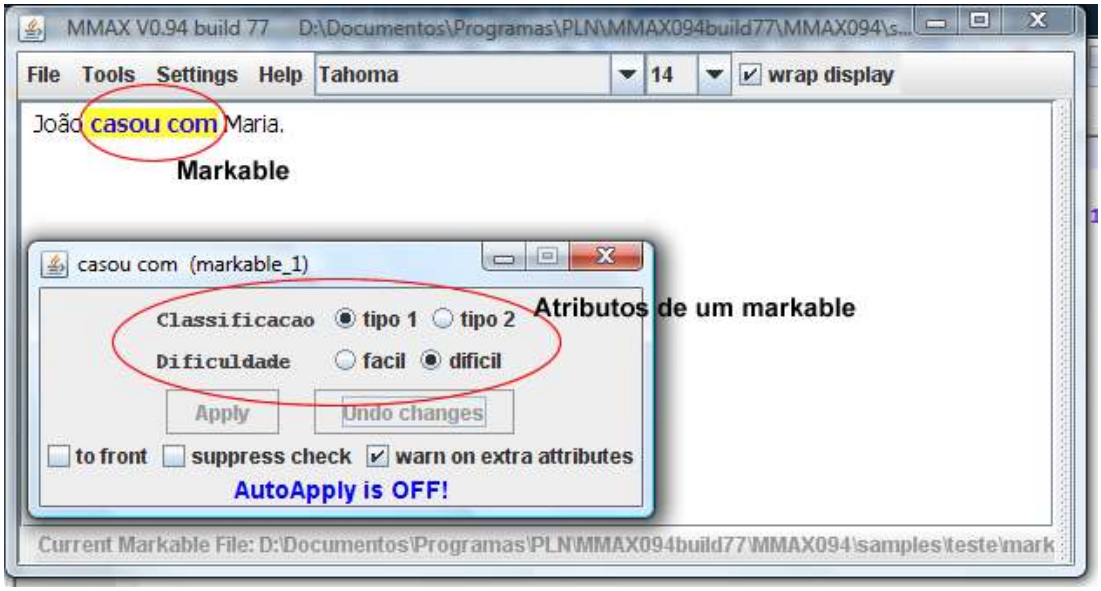

Figura 5: Tela da ferramenta MMAX

Apesar de ser auto-configurável, o MMAX não atendeu as expectativas para um editor de anotação de simplificação dentro do escopo do projeto PorSimples. Com ele não é possível fazer anotação de alinhamento das sentenças de dois textos e fazer acesso a recursos de PLN como dicionários e parsers. De acordo com os requisitos do Editor de Anotação de Simplificação do PorSimples, todos estes recursos têm que estar disponíveis na interação com o usuário e com a ferramenta MMAX isto não é possível. Devido a estes fatores, optou-se pelo projeto e desenvolvimento de um Editor de Anotação de Simplificação.

2.3 Alinhadores sentenciais

Na área de pesquisa em tradução, textos paralelos são conjuntos de textos que englobam um texto original e a sua tradução em uma ou várias línguas. Também chamados de bi-textos, são gerados por ferramentas de PLN chamadas de alinhadores, que alinham automaticamente a versão original com as traduções do mesmo texto, geralmente sentença a sentença.

O processo de alinhamento de textos paralelos consiste em identificar a correspondência entre as partes de um texto (parágrafos, sentenças e/ou palavras) e sua tradução. Caseli estudou e implementou vários métodos de alinhamento sentencial (Caseli, 2003). Para um deles, o *Translation Corpus Aligner* (TCA), foi criada uma interface gráfica – o VisualTCA – uma ferramenta visual on-line para alinhamento sentencial automático de textos paralelos que é independente de língua e encontra-se disponível para uso pela comunidade de pesquisa[8]. Esta ferramenta foi testada para verificar a sua eficácia ao alinhar os bi-textos gerados pelo Editor de Anotação de Simplificação.

Após o processo de anotação de um bitexto (versão original / versão simplificada) foi necessário fazer um alinhamento entre as sentenças do texto que foi anotado e o texto que foi gerado após a anotação.

O alinhador VisualTCA apresentou bons resultados, mas o alinhamento não ficou perfeito para os textos utilizados. Com isso, optou-se por fazer uma solução própria que,

[8] http://www.nilc.icmc.usp.br/nilc/tools/pagina-visualtca/visualtca.htm

dentro do contexto do projeto, a solução não seria complexa e o resultado teria uma precisão maior. Mais detalhes de como alinhamento sentencial foi resolvido estão descritos na Seção 4.5.

2.4 Parser PALAVRAS

O parser PALAVRAS (Bick, 2000) é uma ferramenta que realiza a etiquetação morfossintática, sintática, e inclusive semântica, para textos da língua portuguesa. Dentro do contexto deste projeto foi necessário identificar partículas que poderiam comprometer a compreensão de um texto e expô-las no editor para auxiliar o trabalho do anotador. Com o processamento dos tokens gerados pelo parser PALAVRAS foi possível encontrar as informações necessárias para o projeto.

Na Figura 6 segue um exemplo de como o parser PALAVRAS etiqueta tokens. Demos como entrada o seguinte parágrafo retirado de um texto do córpus do Jornal Zero Hora que está usado na criação do córpus paralelo de simplificações:

> *"O ano era 1978 . As salas de cinema de todo o mundo exibiam uma produção do diretor Joe Dante em que um cardume de piranhas escapava de um laboratório militar e atacava participantes de um festival aquático. ..."*

```
O    [o] <artd> DET M S @>N
ano  [ano] <temp> N M S @SUBJ>
era  [ser] <fmc> <mv> V IMPF 3S IND VFIN @FS-STA
1978     [1978] <date> <card> <NER:date> NUM M P @<SC
$.
</s>
As   [o] <artd> DET F P @>N
salas    [sala] <Lh> N F P @SUBJ>
de   [de] <np-close> PRP @N<
cinema   [cinema] <inst> N M S @P<
de   [de] <np-close> PRP @N<
todo=o=mundo     [todo=o=mundo] SPEC M S/P @P<
exibiam  [exibir] <fmc> <mv> V IMPF 3P IND VFIN @FS-STA
uma  [um] <arti> DET F S @>N
produção     [produção] <act> N F S @<ACC
de   [de] <sam-> <np-close> PRP @N<
o    [o] <artd> <-sam> DET M S @>N
diretor  [diretor] <Hprof> N M S @P<
Joe=Dante    [Joe=Dante] <hum> <np-close> PROP M S @N<
em   [em] <np-long> PRP @N<
que  [que] <rel> SPEC M S @P<
um   [um] <arti> DET M S @>N
cardume  [cardume] <Aich> N M S @SUBJ>
de   [de] <np-close> PRP @N<
piranhas     [piranha] <Aich> N F P @P<
escapava     [escapar] <fmc> <mv> V IMPF 3S IND VFIN @FS-STA
de   [de] PRP @<PIV
um   [um] <arti> DET M S @>N
laboratório  [laboratório] <inst> N M S @P<
militar  [militar] <np-close> ADJ M S @N<
e    [e] <co-fin> <co-fmc> <co-fin> KC @CO
atacava  [atacar] <fmc> <mv> V IMPF 3S IND VFIN @FS-STA
participantes    [participante] <H> N M P @<ACC
de   [de] <np-close> PRP @N<
um   [um] <arti> DET M S @>N
festival     [festival] <occ> N M S @P<
aquático     [aquático] <np-close> ADJ M S @N<
$.
```

Figura 6: Etiquetação feita pelo PALAVRAS

O formato do arquivo gerado pela saída do parser PALAVRAS segue os padrões abaixo:

1- `Token /t [ lema ] Informação morfossintática\n`

Neste primeiro padrão temos o token seguido de uma tabulação. Após a tabulação temos entre colchetes o token em sua forma lematizada. Após a forma lematizada do token temos toda a informação morfossintática do token.

```
2- <\s> \n
```

Este segundo padrão é a forma que o parser PALAVRAS identifica o final de sentença.

```
3- $pontuação \n
```

Este terceiro padrão é a forma como o parser PALAVRAS identifica qualquer tipo de pontuação, como “,”, “.” , “!”, “?”, “:”, etc.

De acordo com o manual de simplificação sintática do PorSimples (Specia et al., 2008) precisamos extrair do parser PALAVRAS informações sintáticas para identificar apostos, orações coordenadas, orações subordinadas, orações na voz passiva, orações com verbos não-finitos (infinitivo, gerúndio e particípio passado) e cláusulas relativas.

Com essas informações pudemos construir o motor sintático do Editor (mais informações sobre a extração é fornecida na Seção 4.4).

3 – Tarefas previstas

As tarefas previstas inicialmente, com os seus respectivos comentários sobre sua realização e os resultados, seguem abaixo:

1. **Revisão bibliográfica sobre análise de córpus simplificados, sobre a ferramenta Coh-Metrix e sobre as operações de simplificação sintática propostas no PorSimples.**

   Somente o estudo sobre a ferramenta Coh-Metrix não foi realizado, pois se tornou um projeto de pesquisa de outra Iniciação Científica. O artigo (Petersen & Ostendorf, 2007) e o Relatório Técnico (Specia et al., 2008) foram embasadores para o trabalho. As operações sintáticas são relatadas na Seção 4.4.

2. Familiarização com o parser Palavras.

Esta tarefa foi realizada e está detalhada nas Seções 2.4 e 4.4.

3. Familiarização com sistemas de alinhamento automático de textos em português

Esta tarefa foi realizada e é detalhada na Seção 2.3.

4. Estudo e avaliação de frameworks para desenvolvimento Web, como o Ruby on Rails e ASP.NET.

Esta tarefa foi realizada e é detalhada na Seção 2.1.

5. Estudo sobre formatos de anotação para definição da saída do Editor.

Esta tarefa foi atribuída para outro aluno de Iniciação Científica, o Felipe Viana Perez, que faz parte da equipe do projeto PorSimples.

6. Projeto e implementação da versão 1 do Editor de Anotação, que indicará construções complexas de um texto com a ajuda do parser Palavras.

Esta tarefa foi realizada e é detalhada na Seção 4.

7. Avaliação da interface da versão 1 do editor com usuários reais (anotador de simplificação).

Esta tarefa foi realizada em dois momentos do projeto: no final de maio e no final de junho. Foi importante para testes de corretude do funcionamento do Editor e também para refinamentos na interface gráfica. Mais detalhes na Seção 4.

8. Projeto e implementação da versão 2 do Editor de Anotação, que indicará construções complexas de um texto com a ajuda de índices e medidas utilizadas pela ferramenta Coh-Metrix, adaptada para o Português.

Esta tarefa não foi realizada, pois as tarefas 6 e 7 tomaram mais tempo do que o previsto devido aos problemas que serão apresentados na Seção 4.4.

9. Avaliação da interface da versão 2 do editor com usuários reais (analistas de córpus e anotador de simplificação).

Esta tarefa não foi realizada.

10. Participação nas reuniões do projeto PorSimples, escrita de Relatórios Técnicos e artigos.

De março a maio houve 6 reuniões do projeto. Dentro do contexto do projeto PorSimples o editor foi tema de artigos para o SIGDOC 2008 (Aluisio et al, 2008) e para o TIL 2008 (Aluisio et al, 2008).

4 – O Editor de Anotação de Simplificação

O Editor de Anotação de Simplificação foi construído para dar suporte à criação do córpus paralelo de simplificações. Nos tópicos que seguem são descritos o seu funcionamento, sua arquitetura e os detalhes de seu desenvolvimento. As dificuldades encontradas são também relatadas, assim como as decisões de projeto relacionadas.

4.1 Funcionamento

O editor de anotação de simplificação segue uma arquitetura de três passos. A Figura 7 mostra os processos pelos quais passam um texto original quando submetido à simplificação apoiada pelo Editor.

Primeiramente, dado um texto como entrada, chamado de texto original, um usuário (preferencialmente um lingüista experiente com a tarefa de simplificação textual) faz uma revisão manual no texto para corrigir erros gramaticais e de pontuação, que podem impactar na análise sintática realizada pelo parser. Estas revisões devem ser mínimas; se o texto exigir muita revisão ele deve ser descartado.

O próximo passo é o começo da simplificação de fato. Neste passo, chamado de simplificação natural, o lingüista é livre para fazer qualquer tipo de operação de simplificação, ou seja, não há regras a seguir, porém há um consenso de que as simplificações léxicas, mudança das partes componentes de uma oração para adequar à ordem Sujeito-Verbo-Objeto e inversão de cláusulas serão privilegiadas. Neste passo, o lingüista é auxiliado pelo "Modo Léxico" de apoio no qual apenas um dicionário de palavras simples é usado para ressaltar as palavras consideradas complexas e marcadores discursivos ambíguos ou não usuais. Detalhes sobre a construção do léxico de palavras simples são descritos na Seção 4.3.

O passo seguinte e último é chamado de simplificação forte. Este passo recebe este nome pois o lingüista é forçado a usar as regras de simplificação sintática descritas em Specia et al. (2007). Para auxiliá-lo neste processo,além do auxilio dos dicionários de palavras, o lingüista recebe o suporte do parser PALAVRAS para ressaltar os pontos em que se deve alterar o texto para torná-lo mais simples. Detalhes sobre a construção da ferramenta de suporte à simplificação sintática serão mostrados na Seção 4.4.

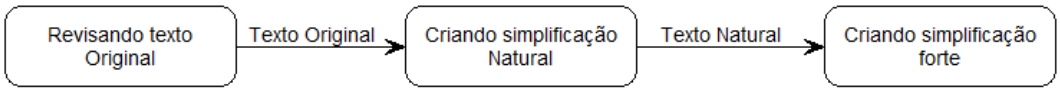

Figura 7: Processo de simplificação

4.2 Arquitetura

Conforme descrito na Seção 2.1, para o desenvolvimento do editor foi utilizado o meta-framework para desenvolvimento WEB *Ruby on Rails*. Nesta seção, descreveremos a sua arquitetura.

Dentro do editor, textos são gravados token a token. Um processo de simplificação dentro do editor é chamado de produção. Cada produção pode possuir no máximo 3 textos, um em sua versão original, um em sua versão natural e um em sua versão forte. Cada texto é segmentado em sentenças e nelas são marcados os parágrafos em que elas pertencem. Cada sentença possui um conjunto de tokens chamado de *word*. Cada *word* possui uma *feature*. Esta feature

possui toda informação morfossintática extraída do parser PALAVRAS. Na Figura 8 segue o diagrama de classes construído para a camada model do ambiente Ruby on Rails.

O Ruby on Rails possui uma camada de persistência de objetos para bancos de dados relacionais, dessa forma temos classes que são persistentes, ou seja, herdam de ActiveRecord, e classes que não são persistentes dentro da camada model.

Abaixo segue a descrição das funcionalidades de cada classe:

Production: responsável por armazenar uma produção de simplificação. Ela armazena o título dos textos e é uma classe persistente.

Texto: responsável por armazenar os textos de uma produção (Production) de simplificação. Ela armazena o texto em sua forma bruta (brute), ou seja, não processada e pode ter um tipo (Original, Natural ou Forte). Ela é uma classe persistente.

Sentence: responsável por armazenar as sentenças de um texto (Texto). Ela armazena o parágrafo ao qual uma sentença pertence dentro do texto. A sentença é uma unidade importante dentro do editor, pois cada operação de simplificação e os alinhamentos são feitos dentro e com de uma sentença, respectivamente. Esta é uma classe persistente.

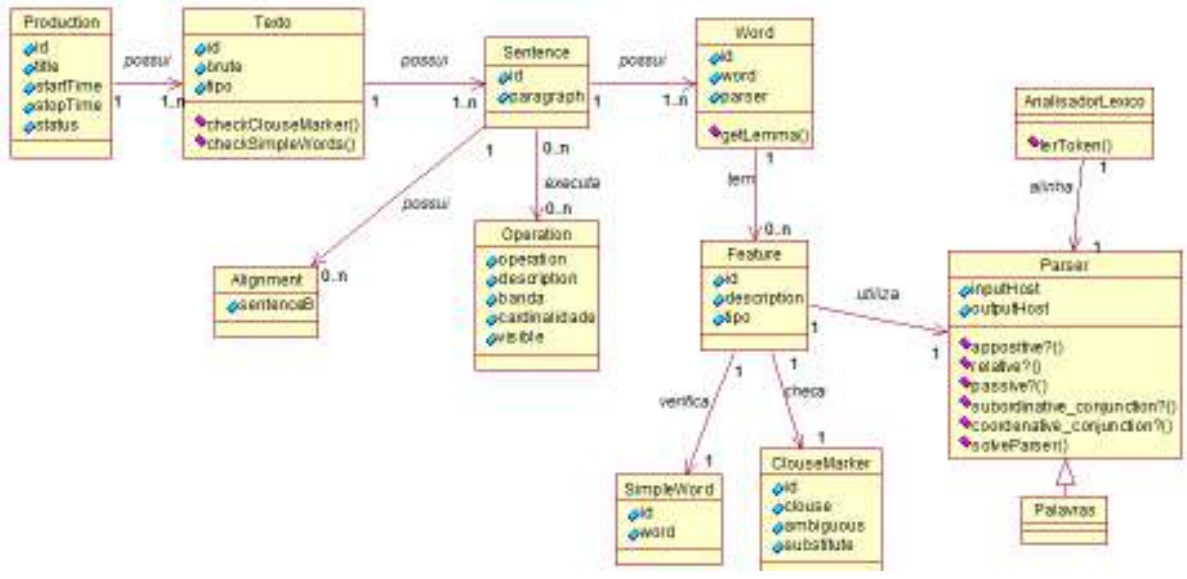

Figura 8: Diagrama de classes do model

Operation: responsável por armazenar as operações feitas em uma sentença. Cada operação possui uma descrição (description), uma classificação (banda) e uma cardinalidade inerente a

ela. Esta cardinalidade será importante futuramente, pois é com ela que iremos alinhar os bitextos. Esta é uma classe persistente.

Alignment: responsável por armazenar o alinhamento entre duas sentenças. Esta é uma classe persistente.

Word: responsável por armazenar os tokens de uma sentença (Sentence). Além do token (*word*) é armazenada a informação morfossintática do parser (parser). Esta é uma classe persistente.

Feature: responsável por armazenar todos os recursos que um token (*word*) possui. Estas features podem ser sintáticas ou léxicas e são extraídas das informações morfossintáticas do parser. Esta é uma classe persistente.

SimpleWord: responsável por armazenar todas as palavras, na forma lematizada da língua portuguesa, consideradas simples dentro do escopo do projeto. Mais detalhes sobre a construção do dicionário de palavras simples são dados na Seção 4.3. Esta é uma classe persistente.

ClouseMarker: responsável por armazenar os marcadores discursivos. Os marcadores discursivos podem ser ambíguos ou não e podem ter um candidato a substituto. Mais detalhes sobre a construção do dicionário de marcadores discursivos são dados na Seção 4.3. Esta é uma classe persistente.

Parser: é uma classe "abstrata"[9] responsável por apresentar um modelo de métodos que um parser teria que ter para ser utilizado dentro do editor. Esta classe não é persistente.

Palavras: é uma herança da classe parser, portanto implementa os métodos da classe parser. Esta classe envia um texto em sua forma bruta para o servidor no qual roda o parser PALAVRAS e processa sua saída obtendo os tokens do parser com suas respectivas etiquetações morfossintáticas. Esta classe não é persistente.

[9] Não há classes abstratas no Ruby. Para dar este comportamento foi criado um mecanismo para ter este resultado.

AnalisadorLexico: responsável por fazer a tokenização de um texto em sua forma bruta. A diferença da tokenização feita pelo parser PALAVRAS e a tokenização feita pelo analisador léxico criado para o editor será explicada com mais detalhes na Seção 4.4.

4.3 Léxico

No editor, a parte léxica auxilia o anotador ressaltando palavras consideradas complexas para a compreensão e ressaltando os marcadores discursivos ambíguos ou não usuais. Este recurso está disponível para o anotador tanto na simplificação natural quanto na simplificação forte.

O dicionário de palavras simples, que o editor utiliza, é composto de 7.532 lemas extraídos de um dicionário infantil (Biderman, 2005), da lista de palavras concretas de (Janczura et al., 2007) e do córpus de textos do jornal Zero Hora, coluna "Para seu filho ler" dos anos 2006 e 2007. Esta coluna apresenta temas da atualidade, como política, esporte e cultura para o público infantil. Com essas fontes temos um bom volume de palavras consideradas simples para o trabalho do editor em explicitar aquelas que são diferentes. Desta forma podem ser alvo de análise e possível troca pelo lingüísta.

Os marcadores discursivos foram extraídos do software DiZer[10] (Pardo and Nunes, 2006) desenvolvido no NILC. São 174 marcadores do discurso que são marcados como ambíguos/complexos ou não e com seu possível candidato a substituição dentro de uma sentença.

Vale lembrar que fica sempre a critério do anotador utilizar ou não o auxílio do processador léxico. Nas Figuras 9 e 10 podemos observar como o processador léxico auxilia o anotador no processo de anotação de córpus paralelo, fornecendo uma interface para substituição de palavras.

[10] http://www.icmc.usp.br/~taspardo/DiZer.htm

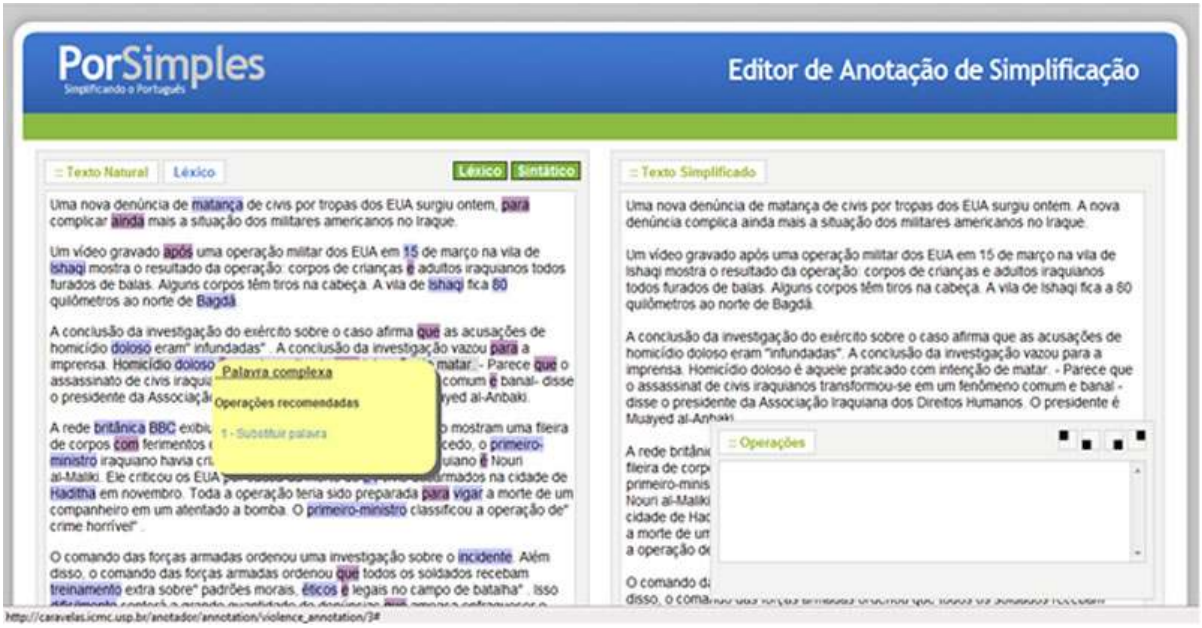

Figura 9: Editor oferecendo a opção de substituir palavra complexa para a palavra "doloso".

No editor, as palavras que estão marcadas com a cor azul são palavras consideradas complexas e a palavras que estão em roxo são os marcadores discursivos.

Figura 10: Interface de substituição de palavra

4.4 Sintático

No editor, a parte sintática auxilia o anotador ressaltando tokens em segmentos de uma sentença que prejudicam a compreensão de um texto e que é necessário fazer alguma simplificação.

Para cada fenômeno sintático apontado pelo editor tem-se um conjunto de operações possíveis que o anotador precisa fazer ou não (fica a seu critério) para simplificar a sentença. Estes fenômenos e as possíveis operações estão no manual de simplificação sintática (Specia et al., 2008). Segue abaixo os fenômenos gramaticais ressaltados pelo editor, as cores que são usadas pelo editor para grafar os termos e as possíveis operações sugeridas pelo editor:

- **Orações coordenadas:** grafadas com a cor verde pelo editor. É sugerida uma divisão da sentença para separar as orações.
- **Orações subordinadas (no geral):** grafadas no texto com a cor cinza pelo editor. É sugerida uma divisão da sentença para separar as orações, não simplificar a sentença ou inverter a ordem da sentença. As operações sugeridas aqui são bem distintas, pois o grupo de orações subordinadas é grande e não é possível saber com certeza com que tipo de oração está se trabalhando com a saída do parser PALAVRAS. De acordo com o manual de simplificação sintática é sugerida uma operação para um conjunto de orações subordinadas e para outros conjuntos são sugeridas outras operações. Neste ponto é muito importante a intervenção do anotador, pois só um usuário especialista saberá que atitude tomar diante desta situação.
- **Orações subordinadas adjetivas (relativas):** grafadas com a cor vermelha pelo editor. É sugerida uma divisão da sentença para separar as orações.
- **Orações na voz passiva:** grafadas com a cor amarela pelo editor. É sugerida uma mudança de voz (para a voz ativa) na sentença.
- **Aposto:** grafadas com a cor verde escuro pelo editor. É sugerida uma divisão da sentença.

Embora as orações com verbos não finitos seja um dos fenômenos reportados no Manual de Simplificação Sintática , elas não são ressaltadas, pois a operação relacionada com elas é não simplificação.

A Figura 11 mostra a tela de simplificação forte sendo apoiada pelo processador sintático do editor. Observe que o cursor do mouse está posicionado sobre a palavra "que" enfatizada em vermelho, considerado um idicador de orações relativas, segundo o processamento do parser PALAVRAS. Para este caso é oferecida a operação de dividir sentença.

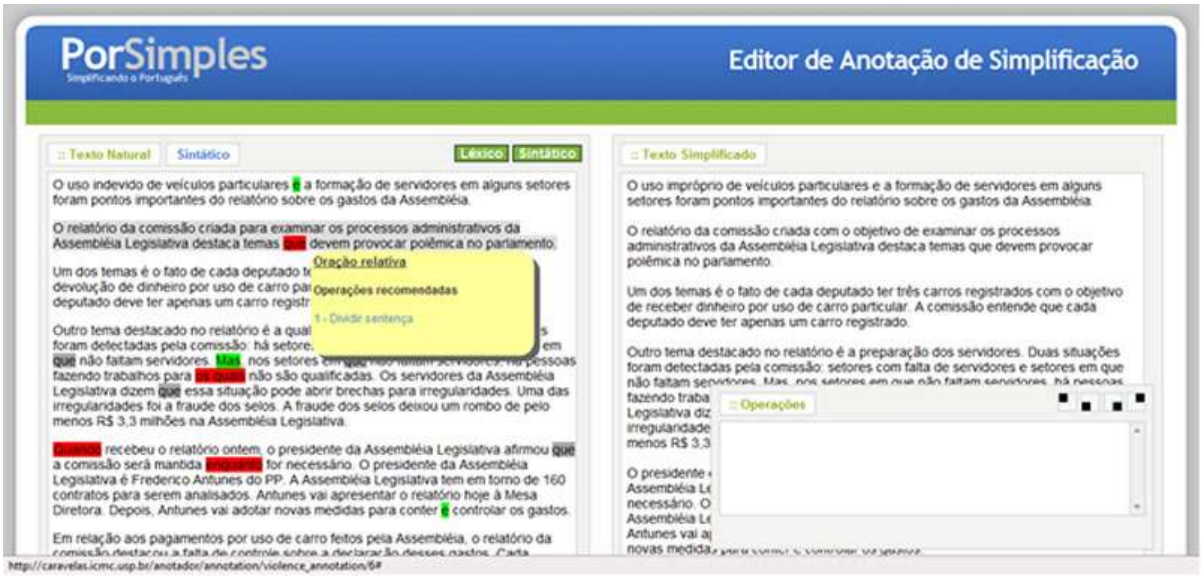

Figura 11: Processamento sintático do Editor

Conforme descrito anteriormente na Seção 2.1, foi utilizado o parser PALAVRAS para a construção do motor sintático do editor.

Um dos grandes problemas encontrados no projeto foi a renderização (ou tokenização) do texto proposta pela arquitetura texto/sentence/word explicado na Seção 4.2. Isso porque a etiquetação feita pelo parser PALAVRAS desmembra formas contraídas de palavras. Esta solução foi necessária, pois temos informações do motor léxico e sintático inerentes a cada token. Observando a Figura 6 nas linhas 16 e 17 temos um exemplo de token em sua forma contraída: os tokens "de" (preposição) e "o" (artigo), representam o token "do"

no texto cru. Existe uma grande lista de palavras contraídas na língua portuguesa; veja na Tabela 1 alguns exemplos.

Tabela 1 : Exemplares de formas contraídas

no	Em + o
no qual	Em + o qual
Em relação ao	Em relação a + o
à	a + a

Para trabalhar com o texto em sua forma pura, ou seja, sem perder a coerência com as formas contraídas desmembradas, foi necessária a construção de um analisador léxico para extrair todos os tokens do texto cru. Estes tokens gerados pelo analisador léxico são os tokens que serão gravados no banco de dados para futura renderização. Para não perder as informações morfossintáticas geradas pelo parser PALAVRAS foi necessário criar um mecanismo de alinhamento entre os tokens gerados pelo parser PALAVRAS e os tokens gerados pelo analisador léxico construído para o projeto.

O desafio encontrado para construir este alinhamento é que estas listas não são equivalentes e não há uma lógica que faça esse alinhamento de forma direta. A solução encontrada foi construir uma abordagem em que se permitem perder alguns tokens do parser PALAVRAS. Mesmo perdendo algumas informações do parser PALAVRAS, as informações perdidas não estão em pontos críticos para a abordagem proposta para o projeto.

Uma possível solução para não perdermos nenhuma informação é obter a lista de contrações utilizada pelo parser PALAVRAS. Este pedido já foi feito para o seu projetista e estamos no aguardo de uma resposta.

Após o alinhamento dos tokens, as informações do analisador léxico com suas respectivas informações morfossintáticas extraídas do parser PALAVRAS são armazenadas na estrutura de words para futura renderização.

4.5 Operações de simplificação e o alinhamento dos bitextos

O manual de simplificação sintática do PorSimples prevê as seguintes operações para simplificação de sentenças dentro de um texto:

- **Não simplificar:** A sentença original é mantida intacta, ou seja, sem nenhuma alteração é sugerida.
- **Fazer reescrita forte:** A reescrita forte envolve modificações complexas na ordem e/ou no modo de apresentação dos fatos. Por exemplo, Petersen & Ostendorf (2007) apresentam como exemplo de reescrita forte o exemplo apresentado abaixo:

Original	*The park service says the solution is money*.
Simplificada	*Why hasn't the National Park Service kept up the park repairs? There is a lack of money.*

- **Fazer reescrita simples:** Ao contrário da reescrita forte, a reescrita simples envolve apenas substituição de palavras ou expressões.
- **Reescrever em SVO:** A estrutura da sentença é alterada para apresentar o conteúdo na ordem Sujeito-Verbo-Objeto.
- **Mudança de voz:** A voz da sentença é modificada. De acordo com o manual de simplificação sintática, a mudança prevista é da voz passiva para a voz ativa.
- **Inverter ordem da sentença:** A ordem das cláusulas de uma sentença/oração é invertida, ou seja, duas cláusulas trocam de posição.
- **Dividir sentença:** A sentença original é dividida (dá origem) em mais de uma sentença na versão simplificada, ou seja, o conteúdo da sentença original é apresentado em mais de uma sentença simplificada. O número de sentenças originadas com essa divisão deverá ser informado pelo usuário por meio da tela apresentada na Figura 12.
- **Unir sentença:** A sentença original (sobre a qual o usuário clicou) será unida com a sentença anterior (que a precede).

- **Remover sentença:** A sentença original não aparece na versão simplificada.
- **Remover parte da sentença:** Parte da sentença original não aparece na versão simplificada.

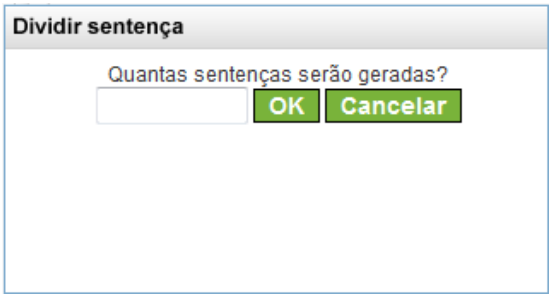

Figura 12: Operação de divisão de sentenças

- **Substituição léxica:** Substituição de uma palavra ou marcador discursivo presente nos dicionários (palavras simples e marcadores discursivos), marcadas pelo léxico por alguma palavra ou expressão fornecida pelo anotador conforme mostrado nas Figuras 9 e 10.

As operações são disponibilizadas através de um menu *pull down* sentença a sentença. Para o anotador acessá-los basta clicar em uma sentença do texto que se quer anotar conforme mostrado na Figura 13. Nesta figura temos em "1" a sentença 2 sendo clicada pelo usuário e o menu é exibido para oferecer as operações disponíveis para cada sentença. Em "2" temos as operações já realizadas para a sentença selecionada.

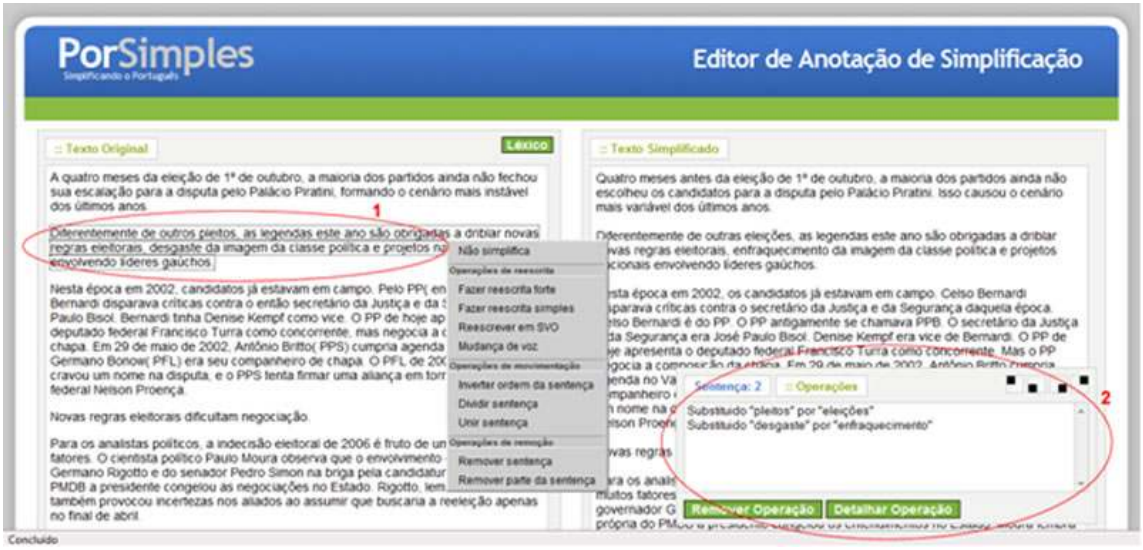

Figura 13: Operações disponíveis por sentença

Para que possamos alinhar as sentenças dos dois textos (versão original e versão simplificada) nos aproveitamos da propriedade de cardinalidade de uma operação. Cada operação feita sobre um texto que se está anotando nos diz quantas sentenças serão produzidas no texto que está sendo reescrito. A partir desta característica de cardinalidade de uma operação, após o processo de anotação é feito um pós-processamento para gerar todos os links das sentenças dos textos original e simplificado e armazenar a informação sobre o alinhamento no banco de dados.

A Tabela 2 apresenta as operações com suas respectivas cardinalidades.

Tabela 2: Cardinalidade das operações de simplificação

Operação	Cardinalidade
Não simplifica	1
Fazer reescrita forte	1
Fazer reescrita simples	1
Reescrever SVO	1
Mudança de voz	1
Inverter ordem da sentença	1
Dividir sentença	N
Unir sentença	-1
Remover sentença	0
Remover parte da sentença	1
Substituição léxica	1

A operação de divisão de sentença é a única operação de simplificação em que não se sabe quantas sentenças novas serão geradas no processo de simplificação. Conforme mostrado na Figura 12 fica a critério do anotador registrar quantas sentenças serão geradas.

5 - Considerações finais

Este projeto de Iniciação Científica se mostrou bem interessante e motivador, pois houve uma mistura das áreas de PLN e IHC o que possibilitou um bom conhecimento das duas áreas.

Atualmente, o editor está sendo utilizado por uma lingüista experiente para a simplificação de 100 textos jornalísticos. Estes textos serão utilizados em estudo futuro para a construção da abordagem estatística do simplificador automático. A tarefa de inserção destes 100 textos na base de dados ainda está sendo executada. Temos atualmente 44 textos já inseridos e sendo simplificados.

Para disponibilizar o córpus publicamente ainda é necessário gerar as anotações em formato XCES[11], como decidido no escopo do PorSimples e para tornar o Editor um software disponível para outros projetos ainda é necessário criar um processo de login, tabelas de usuários & contas na base de dados. O primeiro trabalho será realizado por um bolsista do PorSimples, e o segundo pelo próprio autor deste projeto.

O Editor de Anotação de Simplificação foi a primeira experiência para a construção da ferramenta de auxilio à simplificação de textos proposta pelo projeto PorSimples. Com o desenvolvimento do editor antes do desenvolvimento da ferramenta de auxílio à simplificação, ganhou-se experiência principalmente na área de IHC.

Conforme foi descrito na Seção 4.4, a tarefa que dispendeu maior esforço e apresentou maior dificuldade foi o alinhamento entre a saída do parser PALAVRAS e a saída do analisador léxico desenvolvido no projeto.

[11] http://www.xces.org/

6 – Referências

Aluísio, S.M.; Specia, L.; Pardo, T.A.S.; Maziero, E.G.; Caseli, H.M.; Fortes, R.P.M. "A Corpus Analysis of Simple Account Texts and the Proposal of Simplification Strategies: First Steps towards Text Simplification Systems", Proceedings of the 26th ACM International Conference on Design of Communication, 2008, in press.

Aluísio, S.M.; Specia, L.; Pardo, T.A.S.; Caseli, H.M.; Pereira, T.F. "Building a parallel corpus of original and simplified texts", Webmedia 2008 - Workshop em Tecnologia da Informação e Linguagem Humana (TIL-2008)

Bick, E. (2000). The Parsing System "Palavras": Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework. PhD thesis – Aarhus University. Aarhus, Denmark: Aarhus University Press, November 2000.

Biderman, M. T. C. (2005). Dicionário Ilustrado de Português. São Paulo, Editora Ática. 1ª Edição. São Paulo: Ática.

Caseli, H.M. *Alinhamento sentencial de textos paralelos português-inglês.* Dissertação de Mestrado. ICMC-USP, Abril, 2003

Christoph Müller and Michael Strube. 2001. MMAX: A tool for the annotation of multi-modal corpora. In Proceedings of 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Seattle, Wash., 5 August 2001, pages 45–50.

Janczura, G. A., Castilho, G. M., Rocha, N. O. 2007. Normas de concretude para 909 palavras da língua portuguesa. Psic.: Teor. e Pesq., vol. 23, 195-204.

Larman, Craig (2007). Utilizando UML e Padrões - 3ª Edição pp 229-231

Mapleson, D.L.: Post-Grammatical Processing for Discourse Segmentation. PhD Thesis. School of Computing Sciences, University of East Anglia, Norwich (2006).

Max, A.: Writing for Language-impaired Readers. In the Proceedings of Seventh International Conference on Intelligent Text Processing and Computational Linguistics (CICLing), pp. 567-570. Mexico City, Mexico, (2006).

Pardo, T.A.S., Nunes, M.G.V.: Review and Evaluation of DiZer - An Automatic Discourse Analyzer for Brazilian Portuguese. PROPOR 2006, LNCS, vol. 3960, pp. 180-189. (2006)

Petersen, S. E., Ostendorf, M.: Text Simplification for Language Learners: A Corpus Analysis. 2007. In Proceedings of the Speech and Language Technology for Education Workshop (Pennsylvania, USA, October 1-3, 2007). SLaTE-2007. Carnegie Mellon University and ISCA Archive, http://www.isca-speech.org/archive/slate_2007. 69-72.

Ribeiro, V. M.: Analfabetismo e alfabetismo funcional no Brasil. Boletim INAF. São Paulo: Instituto Paulo Montenegro (2006).

Salmon-Alt, S. and Vieira, R. 2002. Nominal expressions in multilingual corpora: Definites and demonstratives. In Proceedings of the LREC 2002, Las Palmas de Gran Canaria.

Specia, L.; Aluisio, S.M.; Pardo, T.A.S. 2008. Manual de Simplificação Sintática para o Português. Technical Report NILC-TR-08-06. São Carlos-SP. http://www.nilc.icmc.usp.br/nilc/publications.htm#TechnicalReports

Vieira, R.; Salmon-Alt, S.; Gasperin, C.; Schang, E. and Othero, G. 2002a. Coreference and anaphoric relations of demonstrative noun phrases in multilingual corpus. In Proceedings of the DAARC 2002, Estoril.

Vieira, R.; Salmon-Alt, S. and Schang, E. 2002b. Multilingual corpora annotation for processing definite descriptions. In Proceedings of the PorTAL 2002, Faro.

Vieira, R.; Gasperin, C. ; Goulart, R. and Salmon-Alt, S. From Concrete to Virtual Annotation Mark-up Language: The Case of COMMOn-REFs. In The ACL-2003 Workshop on Linguistic Annotation: Getting the Model Right, pp. 6-13 (2003).