JOSÉ LOPES MOREIRA FILHO

DESENVOLVIMENTO DE UM *SOFTWARE* PARA PREPARAÇÃO DE AULAS DE INGLÊS COM CORPORA

MESTRADO EM

LINGÜÍSTICA APLICADA E ESTUDOS DA LINGUAGEM

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO

2007

JOSÉ LOPES MOREIRA FILHO

DESENVOLVIMENTO DE UM *SOFTWARE* PARA PREPARAÇÃO DE AULAS DE INGLÊS COM CORPORA

Dissertação apresentada à Banca Examinadora da Pontifícia Universidade Católica de São Paulo, como exigência parcial para obtenção do título de Mestre em Lingüística Aplicada e Estudos da Linguagem, sob orientação do Prof. Dr. Antonio Paulo Berber Sardinha.

PUC - SP
2007

BANCA EXAMINADORA

____

____

____

Dedico este trabalho a toda a minha família, pela compreensão e apoio.

AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Tony Berber Sardinha, pelo auxílio, confiança, paciência, pelo exemplo, e por acreditar neste trabalho.

À profa.Dra. Rosinda de Castro Guerra Ramos, pelo carinho, pela confiança, compreensão e contribuições para o trabalho.

A Mona pelas valorosas contribuições para o trabalho escrito na qualificação.

À profa. Dra. Leila Bárbara pela acolhida no programa.

À Coordenadora, Profa. Dra. Elisabeth Brait e a todos os professores do LAEL, pela manutenção do alto nível de qualidade do programa.

A todos os amigos do seminário de orientação e disciplinas, pelas sugestões e pelo companheirismo: Agnes, Ana Júlia, Ana Maria, Antonieta, Carlos, Ciça, Cláudia, Cristiane, Daniela, Demario, Denise, Edivânia, Elaine, Eliane, Flávia, Gisele, Lílian, Luciene, Márcia B.B.B., Márcia Veirano, Michel Monica, Pat, Paulo, Regina, Roberto, Telma, Tony Santos, Vivian, Zeli.

Aos meus professores da graduação: Vera, Inês, Neusa, Frank, José Maria, Márcia Arouca,Valdite, Renata e Osvaldo Succi.

À Secretaria da Educação do Estado de São Paulo pelo Programa Bolsa Mestrado.

A todos da Diretoria de Ensino da Região Leste 3.

A todos os colegas da Oficina Pedagógica da Leste 3 pelo ambiente de trabalho e auxílio.

À minha família toda.

Ao meu pai e minha mãe, José e Rosa, por terem sempre acreditado em mim, terem me ensinado a importância do conhecimento e do estudo, sempre fazendo de tudo para que eu pudesse estudar.

Aos meus irmãos Aline e Wítalo pelo amor e carinho.

À minha esposa Luciana pelo amor e compreensão durante a caminhada.

RESUMO

O trabalho teve como objetivo principal o desenvolvimento de um *software* para preparação semi-automática de aulas de leitura de inglês com corpora para elaboração de materiais didáticos que privilegiem a língua em uso.

Para tanto, o trabalho encontrou suporte teórico na Lingüística de Corpus. A Lingüística de Corpus é uma área inserida na esfera da Lingüística Aplicada, que utiliza uma abordagem empirista e vê a linguagem como sistema probabilístico com base em análises de grandes quantidades de dados lingüísticos reais, língua em uso, por meio de computador (Berber Sardinha, 2004).

Mais especificamente, o trabalho aqui apresentado fundamentou-se na área de pesquisa baseada em corpus que se preocupa com o processo de ensino e aprendizagem de línguas estrangeiras (Berber Sardinha, 2004; Hunston, 2002; Kennedy, 1998; Johns, 1991; Sinclair, 1991; Flowerdew, 1993; Fox, 1998; Trible & Jones, 1990).

Além da Lingüística de Corpus, o projeto fundamentou-se nos pressupostos teóricos e metodológicos do Ensino de Leitura (Grellet, 1981; Nuttal, 1988; Scott et al., 1984) para criação de uma atividade padrão para o desenvolvimento de uma ferramenta de preparação semi-automática de aulas de leitura de inglês.

O projeto buscou fazer uma contribuição original para o estudo do ensino de língua inglesa no Ensino Médio da escola pública, visto que não há estudos dessa natureza para o contexto educacional mencionado, e também por aproveitar textos autênticos e corpora, conjuntamente, em atividades didáticas.

Embora haja uma série de publicações e trabalhos que enfoquem a utilização dos instrumentos da Lingüística de Corpus no ensino de línguas, há uma grande necessidade de pesquisas sobre o desenvolvimento de ferramentas computacionais que possibilitem o ensino de línguas por meio de recursos usados na exploração de corpora para o contexto da escola pública brasileira.

Para a criação do *software* por meio de programação em linguagem *Visual Basic 6* foram utilizados um corpus de treinamento composto por 80 textos de diversos gêneros, e o BNC (*British National Corpus*) como corpus de referência, com 4027 textos que totalizam mais de 100 milhões de palavras.

ABSTRACT

The main aim of this study is the development of semi-automatic software to prepare reading materials for English as a Second Language classes, using corpora to elaborate teaching material that explores real language.

The theoretical underpinning of the research is provided by Corpus Linguistics; a study area in the domain of Applied Linguistics that takes an empirical approach to language and sees it as a probabilistic system. It is based on the analysis of large amounts of real linguistic data by means of computers (Berber Sardinha, 2004).

More specifically, the corpus based research area concerned with the teaching process and foreign language learning informs this study. Some references can be found in Berber Sardinha (2004), Hunston (2002), Kennedy (1998), Johns (1991), Sinclair (1991), Flowerdew (1993), Fox (1998), Trible & Jones (1990).

In addition to Corpus Linguistics, this investigation is also based on previous studies in the Teaching of Reading, such as those from (Grellet, 1981; Nuttal, 1988; Scott et al., 1984) for creating a standard activity model to develop a semi-automatic tool to prepare English reading lessons in Visual Basic 6 (VB6).

Our intention is to make an original contribution to English Language Teaching at the High School level in public schools. The goals are twofold: to develop innovative studies in this particular subject and to provide an opportunity for the use of authentic texts and corpora in didactic activities.

Although literature in the area is extensive, there still seems to be a need for research on the development of computational tools that will make it possible to use the same resources used in corpora exploration for language teaching in Brazilian public schools.

Two corpora were used in the study: a reference corpus and a training corpus. The reference corpus was the British National Corpus (BNC), released in 1995 with more than 100,000,000 words. The training corpus, whose goal is to test the software functions, was collected from the Internet and has a total of 80 texts from varied genre.

SUMÁRIO

Introdução

Uma das grandes dificuldades enfrentadas por professores tanto de Língua Inglesa como de Língua Portuguesa do Ensino Médio é maximizar a exposição do aluno à língua em uso. Para conseguir alcançar seus objetivos, muitos professores têm à sua disponibilidade apenas o livro didático, e no caso da disciplina de Língua Inglesa surge um outro problema, muitas vezes não há sequer livro didático.

A disponibilidade de livros e materiais didáticos para a disciplina de língua inglesa na rede pública de ensino pode não ser ideal, mas o livro didático ainda se constitui como a principal fonte para o professor preparar aulas e expor a língua aos alunos. O professor utiliza o livro como fonte de exercícios, ensino de gramática e textos para leitura.

No entanto, os livros didáticos, em geral e para esse contexto de ensino, podem ser questionados em relação à sua qualidade, por serem limitados e por não oferecerem a língua realmente em uso, conforme mostram vários estudos (Kennedy 1987; Holmes 1988; Mindt 1992; Ljung 1990; Gilmore 2004; Berber Sardinha 2006b; Alonso 2006; Vicentini, 2006).

Muitos livros didáticos, por uma série de questões, utilizam textos inventados para o ensino de línguas. Para Guariento & Morley (2001), a utilização de textos inventados pode trazer efeitos negativos para a aprendizagem: sentimento do aluno em estar aprendendo algo que não é útil, habilidades não plenamente desenvolvidas e uso não natural da língua. Para uma parte da sociedade, esse tipo de ensino é tachado de ‘the book is on the table’.

Segundo Berber Sardinha (2006a:1), a frase ‘the book is on the table’, bordão de programas humorísticos para ridicularizar aquele que não sabe falar inglês de verdade, resume um tipo de inglês ‘de mentira’ que é ensinado na escola. Geralmente por meio de textos inventados, frases soltas e vazias de significado, mas corretas gramaticalmente, para ver se o aluno sabe formar frases com o verbo ‘to be’.

Por sua vez, a utilização de textos autênticos, isto é, textos que não foram criados para ensinar língua, é vital se o objetivo de ensino é a comunicação, a interação social e a execução de

tarefas no mundo real. Para Gilmore (2004:367), se o objetivo é preparar o aluno para utilizar a língua de forma independente, então devemos, em algum estágio, apresentar discursos reais.

De acordo com Guariento & Morley (2001:347), há uma unanimidade de que a utilização de textos autênticos em sala de aula é benéfica para o processo de ensino-aprendizagem. Entre os argumentos a favor da utilização de linguagem autêntica estão os de que ela promove um aumento da motivação, pois o aluno sente que está aprendendo a "língua de verdade".

Dessa forma, é imprescindível que o professor leve textos autênticos para utilização em sala de aula. Porém, é preciso também que o professor possua instrumentos para poder trabalhar com textos autênticos em sala de aula. O professor precisa de ferramentas para auxiliá-lo em seu trabalho com linguagem autêntica. Uma delas, a nosso ver, seria um programa de computador que auxiliasse o professor a criar materiais de ensino a partir de textos autênticos e de corpora. Este estudo propôs-se a criar tal programa.

Para tanto, o trabalho encontrou suporte teórico e metodológico principalmente na Lingüística de Corpus. A Lingüística de Corpus é uma área inserida na esfera da Lingüística Aplicada, que utiliza uma abordagem empirista e vê a linguagem como um sistema probabilístico com base em análises de grandes quantidades de dados lingüísticos reais, língua em uso, por meio de computador (Berber Sardinha, 2004).

Mais especificamente, o trabalho aqui proposto fundamenta-se na área de pesquisa baseada em corpus que se preocupa com o processo de ensino e aprendizagem de línguas estrangeiras (Berber Sardinha, 2004; Hunston, 2002; Kennedy, 1998; Johns, 1991; Sinclair, 1991; Flowerdew, 1993; Fox, 1998; Trible & Jones, 1990).

A Lingüística de Corpus pode fornecer um importante para o desenvolvimento de materiais para o ensino de línguas. Há um grande interesse na utilização da instrumentação da Lingüística de Corpus no ensino de línguas. A disponibilidade de corpora[1] propiciou abordagens para o ensino com base na exploração de corpora, como o *Lexical Syllabus* (Willis, 1990), o *Lexical Approach* (Lewis, 2000) e o *Data Driven Learning* (*DDL*) (Johns, 1986).

[1] De forma geral, entende-se corpora (plural de corpus) como banco de dados formados pela linguagem em uso, falada e/ ou escrita.

A crescente popularização de corpora e a disponibilidade de programas de análise lingüística vêm oferecendo também novas e interessantes direções para o desenvolvimento de materiais didáticos. São muitos os trabalhos que defendem a utilização da instrumentação da Lingüística de Corpus no desenvolvimento de material didático para o ensino de línguas: o uso de ferramentas de exploração de corpora como auxílio para o *design* de curso (Flowerdew 1993), o desenvolvimento de material didático (Thurstun & Candlin 1998), as possibilidades de uso dos dados de pesquisa de corpora para ensino de vocabulário contextualizado, gramática e pragmática (Fox 1998), e os estudos de Ferrari (2004), Barbosa (2004) e Souza (2005) que materializam teorias e metodologias em atividades didáticas para o ensino da língua em uso.

Os estudos da linguagem por meio de corpora, intensificados pelo avanço tecnológico, especificamente o computador, têm mudado a maneira como estudamos a língua. Graças a esses estudos, pode-se afirmar que a linguagem é padronizada. Cada vez mais, os lingüistas de corpus mostram evidências de que as palavras se combinam para gerar significado, e que o significado das palavras em uso está relacionado ao seu contexto. A idéia central é a de que utilizamos unidades pré-fabricadas, padrões léxico-gramaticais, na comunicação, as quais podem ser evidenciadas e quantificadas por meio de estudos com corpora (Sinclair, 1991). Desse modo, a melhor maneira de descobrir os significados das palavras é observá-las em seu contexto, o que justifica as novas abordagens criadas, mencionadas anteriormente, e a inclusão dos dados provenientes de análise de corpora em materiais didáticos.

Para McEnery & Wilson (1996:104), os exemplos extraídos do corpus são importantes para a aprendizagem de línguas, pois expõem os alunos desde os estágios iniciais do processo de aprendizagem aos tipos de frases e vocabulários que possivelmente serão encontrados em textos autênticos da língua ou no uso da língua em situações reais de comunicação.

O trabalho com textos autênticos conjuntamente com o uso de corpora na sala de aula pode ajudar o professor a ensinar a língua em uso. Com eles, pode-se dar condições para que os alunos se conscientizem sobre as unidades pré-fabricadas da língua, que são os padrões léxico-gramaticais, e a característica probabilística da linguagem, isto é, como a língua realmente funciona. Vicentini (2006:16) aponta o entusiasmo de autores de materiais didáticos com o valor pedagógico dos padrões léxico-gramaticais provenientes de estudos com corpora. Para isso, a Lingüística de Corpus possui uma ampla instrumentação que pode ser aproveitada.

Segundo Berber Sardinha, (2004:255). A utilização dos recursos da Lingüística de Corpus no ensino tem como principal instrumento a concordância[2]. As concordâncias são extraídas de um corpus por *software* específicos, chamados concordanciadores[3]. Há uma grande variedade de *software* oferecidos no mercado, tal como o *MicroConcord*, o *Monoconc* e o *WordSmith tools*, que além de criarem linhas de concordâncias, também disponibilizam para o lingüista outros recursos avançados para a análise de corpora, mostrando freqüência de palavras e estatísticas. A função dessas ferramentas é maximizar o potencial dos corpora, possibilitando a reorganização das informações no corpus e a descoberta de fatos interessantes sobre o uso da língua.

Uma proposta interessante para o ensino de línguas seria disponibilizar essas ferramentas para uso de professores e alunos. Segundo Johns (1991, apud Berber Sardinha, 2004:291), "a pesquisa é uma ferramenta valiosa demais para ficar nas mãos dos pesquisadores". Contudo, tendo em vista que a Lingüística de Corpus é uma área em pleno desenvolvimento e o uso de corpora no ensino ainda ser novidade para o contexto de ensino brasileiro, é preciso primeiro levar algumas questões em consideração.

Embora haja muitos trabalhos que enfoquem a utilização dos instrumentos da Lingüística de Corpus no ensino de línguas, não há uma ferramenta específica para brasileiros. Do mesmo modo, ainda não há estudos sobre o desenvolvimento de ferramentas computacionais específicas que possibilitem o ensino de línguas por meio de recursos usados na exploração de corpora para o contexto da escola pública. Em muitas pesquisas que utilizam os instrumentos computacionais da Lingüística de Corpus, são utilizadas concordâncias já preparadas e impressas pelo computador provenientes de intensivas análises de corpora.

Além disso, há alguns entraves para a utilização desses tipos de programas na escola pública. A maioria das ferramentas computacionais no mercado é desenvolvida para o uso de lingüistas e estudiosos da linguagem, com uma série de opções e terminologias que dificultariam o uso tanto para o aluno como para o professor que ainda não tenha conhecimento aprofundado sobre o assunto. Outras ferramentas são on-line e requerem uma boa conexão à Internet, o que nem todas as escolas possuem e, além disso, mesmo uma boa conexão não suportaria um número

[2] "Uma concordância consiste de uma listagem de cotextos, (palavras ao redor) nos quais um dado item (palavra isolada, composta, estrutura, pontuação) ocorre" (Berber Sardinha, 2004:272)
[3] "Programas que permitem que o usuário procure por palavras específicas em um corpus, fornecendo exaustivas listas para as ocorrências da palavra em contexto" (Biber et al.,1998:15)

grande de usuários simultâneos. Uma outra barreira é a questão da língua. Não há *software* desse tipo em língua portuguesa. Um dos concordanciadores mais conhecidos, o *MicroConcord*, por exemplo, está em inglês e não lida bem com textos acentuados. Por fim, outro fator decisivo é a dificuldade de aquisição desses *software*. Os preços são proibitivos para a escola. O preço da licença do *software WordSmith tools*, por exemplo, para o uso de até 50 usuários é de aproximadamente R$1.900,00 reais (€753).

Buscando preencher tal lacuna, conforme dissemos, esta pesquisa propõe o desenvolvimento de um *software* para auxiliar o professor da escola pública na elaboração de material didático baseado em corpus. Especificamente, pretende-se a criação de um *software* para preparação de atividades de leitura em inglês como língua estrangeira. O usuário cria o material por meio de um *wizard* que lhe faz perguntas e oferece opções acerca do conteúdo a ser incluído no material. O *software* proposto nesta pesquisa deverá funcionar como uma ferramenta para o professor utilizar textos autênticos em sala de aula.

O foco da presente pesquisa é o desenvolvimento e avaliação das funções executadas por esse *software*, que possibilite a criação de atividades didáticas de leitura com textos autênticos e dados de análise de corpora, que chamamos inicialmente de Preparador de Aulas e, por fim, de *Reading Class Builder*[4].

O *Reading Class Builder* deve permitir a preparação de uma atividade de leitura em inglês por meio de quatro etapas. Na primeira etapa, o usuário seleciona um modelo de atividade. Na segunda etapa, escolhe um texto, salvo previamente pelo próprio usuário, que será utilizado em sala de aula. Na terceira etapa, o programa exibe informações sobre o texto escolhido. As informações referem-se à dificuldade do texto, extraída por meio de uma fórmula que considera o tamanho das palavras nas frases do texto; densidade lexical[5], conseguida por meio da etiquetagem morfológica das palavras do texto; número de possíveis palavras cognatas, palavras com escrita parecida com palavras do português; lista de palavras do texto e palavras-chave, palavras que mais se destacam. As informações fornecidas na terceira etapa ajudam o usuário a verificar se o texto é adequado para seus objetivos e, também, a explorar o potencial didático do texto escolhido. Na quarta etapa, o usuário define o título da atividade, o idioma dos enunciados,

[4] Nome escolhido para divulgação do *software* após atualização e nova compilação.
[5] Proporção de palavra de conteúdo no texto, tal como substantivos, adjetivos, verbos e advérbios.

palavras para ensinar e um corpus de onde serão extraídos exemplos para exercícios com linhas de concordância. Ao final, o programa traz a aula pronta para ser impressa.

A implementação do *Reading Class Builder* na prática do professor e a aplicação das atividades desenvolvidas por seu uso dependeriam ainda do fornecimento de condições para o professor se conscientizar da necessidade de uma abordagem de análise baseada em corpora e, por isso, não são contempladas pela pesquisa. Sabe-se ainda que a introdução do *Reading Class Builder*, como uma nova tecnologia, possivelmente poderia ser recebida com resistência de alguns professores, pois poderia representar, a princípio, uma carga de trabalho maior. Contudo, apresentamos algumas primeiras impressões do seu uso por professores.

A motivação para o desenvolvimento deste trabalho surge de minha experiência como professor de inglês da rede estadual pública de São Paulo. Lecionando para diversas turmas do Ensino Fundamental e Médio, percebi uma grande carência de livros e materiais didáticos de inglês que privilegiassem textos autênticos, principalmente para o ensino de compreensão escrita em inglês, que conforme os Parâmetros Curriculares Nacionais (Linguagens, códigos e suas tecnologias):

> A competência primordial do ensino de línguas estrangeiras modernas no ensino médio deve ser a da leitura e, por decorrência, a da interpretação. O substrato sobre o qual se apóia a aquisição dessas competências constitui-se no domínio de técnicas de leitura – tais como *skimming*, *scanning*, *prediction* – bem como na percepção e na identificação de índices de interpretação textual (gráficos, tabelas, datas, números, itemização, títulos e subtítulos, além de elementos de estilo e gênero). (Brasil, 2002:97)

Os poucos livros disponíveis na escola, alguns apresentados como coletânea única para as três séries do Ensino Médio, traziam textos e exercícios inventados, com frases soltas sem contexto, com o único objetivo de ensinar gramática. Para suprir a falta de material didático, recorria a textos encontrados na Internet, revistas e em outros meios para a preparação de aulas, o que por um lado poderia caracterizar certa autonomia, constituía também um problema, devido à falta de tempo para elaborar todo o material paras as diferentes turmas e séries em que lecionava.

Assim, buscamos inspiração para esta pesquisa que visou o desenvolvimento de uma ferramenta para facilitar o trabalho do professor. O presente estudo busca trazer uma contribuição original para o ensino de língua inglesa no Ensino Médio da escola pública, visto que não há

estudos dessa natureza para o contexto educacional mencionado, e também por tirar proveito de textos autênticos e corpora, conjuntamente, em sala de aula.

A Lingüística de Corpus destaca-se como a principal área, cujo arcabouço teórico forneceu embasamento para todo trabalho e a justificativa para seu uso é que fornece a instrumentação necessária para base lingüística do quê deve ser ensinado e por fornecer ferramentas e instruções para o desenvolvimento de aplicações computacionais para o tratamento e análise de linguagem autêntica.

Além da Lingüística de Corpus, fizemos uso dos pressupostos teóricos e metodológicos do ensino de leitura de inglês como língua estrangeira (Nuttal 1988; Grellet 1981; Scott 1984) para criação de uma atividade padrão, a fim de servir como base para o desenvolvimento do *Reading Class Builder*. A proposta e os tipos de exercícios dessa abordagem se relacionam perfeitamente com os objetivos da presente pesquisa, dado o contexto de ensino em que se pretende atuar.

Para o desenvolvimento do *Reading Class Builder*, buscou-se também o embasamento metodológico em algumas especificações de *Engenharia de Software* (Sommerville, 2001), visto que todo o processo de programação foi realizado pelo próprio pesquisador. Embora as orientações fornecidas pela área de *Engenharia de Software* sejam gerais e para o desenvolvimento de sistemas mais complexos, foram úteis para o presente contexto de pesquisa, uma vez que podem ser adaptadas.

A metodologia empregada nesta pesquisa consistiu em: (i) levantar quais características e recursos fariam parte do *software* a ser desenvolvido, por meio de leituras, observação dos *software* disponíveis no mercado, levantamento das características do público-alvo do *software*, (ii) criar o *software* por meio de programação em *Visual Basic*[6], a partir das características levantadas no primeiro momento e (iii) testar a funcionalidade e aplicação do *software*, a fim de evidenciar sua eficiência e avaliá-lo de acordo com os objetivos a que se propõem.

[6] Uma das linguagens de programação mais populares no mundo, produzida pela empresa Microsoft, em grande parte da programação é feita visualmente, adicionando janelas, botões, figuras e outros controles existentes no ambiente Windows, ou objetos criados pelo próprio programador.

Após o desenvolvimento (codificação por meio de programação) do *Reading Class Builder*, iniciou-se o seu processo de avaliação. O processo foi realizado por meio de dois procedimentos. Primeiro, realizou-se o teste de funcionalidade do programa, utilizando o *Reading Class Builder* para preparar uma atividade a partir de um determinado texto, escolhido apenas para este propósito. As informações geradas durante o processo foram analisadas e avaliadas de acordo com os aspectos utilizados em seu desenvolvimento. Segundo, realizamos sua a aplicação com o público-alvo para o qual foi desenvolvido. O *Reading Class Builder* foi apresentado a alguns professores do Ensino Médio, que utilizaram o programa na preparação de uma aula e o avaliaram.

1. Objetivos e Questões de Pesquisa

O objetivo principal do estudo é o desenvolvimento e avaliação de um *software* para o ensino de inglês para alunos do Ensino Médio, que utilize os princípios e as ferramentas utilizadas na exploração de corpora da Lingüística de Corpus (Berber Sardinha, 2004), e princípios do ensino de leitura (Nuttal 1988; Grellet 1981), permitindo que o professor possa preparar um material que propicie o contato do aluno com a língua em uso, de modo que este possa construir conhecimentos na língua estudada por meio de descobertas propiciadas pela observação e análise da língua em uso, e também facilitando o processo de criação de materiais de leitura de inglês como língua estrangeira.

Os objetivos da pesquisa são seguintes:

1- Criar um *software* para preparação de atividades de leitura em inglês como língua estrangeira que possa utilizar textos autênticos e corpora.

2- Avaliar o desempenho técnico do *software* criado à luz dos princípios teóricos e metodológicos utilizados na sua elaboração.

3- Avaliar o uso do *software* entre professores.

As questões de pesquisa investigadas foram:

1. Quais as características e recursos essenciais o *software* deve possuir, dado o público alvo e o contexto educacional a que se destina?

2. Como se cria um módulo em *Visual Basic 6* para fazer lista de palavras e concordâncias?
3. Como se cria um módulo em *Visual Basic 6* para descobrir as palavras-chave?
4. Como se cria um módulo em *Visual Basic 6* para identificar as palavras cognatas?
5. Como se cria um módulo em *Visual Basic 6* para etiquetar as palavras de um texto?
6. Como se cria um módulo em *Visual Basic 6* para descobrir a densidade lexical de um texto?
7. Como se cria um módulo em *Visual Basic 6* para definir a dificuldade de um texto?
8. Qual a avaliação do *software*, isto é, até que ponto o produto final atende às especificações de *design*?

2. Organização da dissertação

A dissertação está organizada da seguinte maneira. O capítulo a seguir discute a fundamentação teórica da pesquisa, mostrando os principais conceitos e trabalhos prévios nas áreas da Lingüística de Corpus, e também os pressupostos de ensino de leitura e Engenharia de *software* necessários para o desenvolvimento do *software*. O capítulo 2 apresenta em detalhes a metodologia empregada na pesquisa, incluindo a descrição dos corpora e os procedimentos de desenvolvimento e avaliação do *software*. O capítulo 3 apresenta as ferramentas para a programação e descreve os procedimentos de codificação do *software*. O capítulo 4 apresenta o *software* construído e o funcionamento de seus principais recursos. O capítulo 5 apresenta os resultados do teste e avaliação do *software*, além das considerações finais. As referências bibliográficas e os anexos encerram a dissertação.

Capítulo 1

Fundamentação Teórica

Este capítulo apresenta as áreas que forneceram embasamento teórico para o trabalho e está organizado de acordo com a visão de linguagem e pressupostos teóricos e metodológicos para os procedimentos de desenvolvimento do *software*. Primeiramente, com relação à visão de linguagem, são apresentados os princípios teóricos da Lingüística de Corpus. A seguir, passa-se para os pressupostos que basearam a criação de uma atividade padrão para ser utilizado no *software*. Por fim, são apresentados os princípios utilizados para o desenvolvimento do *software*.

1.1. Lingüística de Corpus

1.1.1. Definição de Lingüística de Corpus

Conforme dito na Introdução, o trabalho aqui proposto tem como fundamentação teórica principal a Lingüística de Corpus. Segundo Berber Sardinha (2004:3),

> *"a Lingüística de Corpus pode ser definida como uma área que se ocupa da coleta e exploração de corpora, ou conjunto de dados lingüísticos textuais coletados criteriosamente, com o propósito de servirem para a pesquisa de uma língua ou variedade lingüística."*

É importante salientar que o papel do computador tem sido central no desenvolvimento da área, sem deixar de lado as grandes contribuições dos corpora compilados e analisados manualmente.

A Lingüística de Corpus tem como principal objetivo estudar a língua a partir da observação de grandes quantidades de textos autênticos, língua natural, que coletados criteriosamente e armazenados em formato eletrônico para fins de pesquisa lingüística formam um corpus.

De acordo com Hunston (2002:2), a palavra corpus tem sido atualmente utilizada para se referir a um conjunto de textos (ou partes de textos) que são armazenados e acessados eletronicamente.

Uma definição completa que envolve todas as características de um corpus é citada por Berber Sardinha (2004:18):

> *"Um conjunto de dados lingüísticos (pertencentes ao uso oral ou escrito da língua, ou ambos), sistematizados segundo determinados critérios, suficientemente extensos em amplitude e profundidade, de maneira que sejam representativos da totalidade do uso lingüístico ou de algum de seus âmbitos, dispostos de tal modo que possam ser processados por computador, com a finalidade de propiciar resultados vários e úteis para a descrição e análise."*

Dada a definição, podemos destacar três atributos básicos da constituição de um corpus: os dados devem ser autênticos, devem ser legíveis por computador e representativos de uma linguagem ou variedade para estudo lingüístico.

Os estudos baseados em corpus vêm auxiliando o desenvolvimento de visões da Lingüística Aplicada. Para Hunston (2002:1), não é um exagero dizer que tais estudos revolucionaram a maneira de estudar a linguagem e suas aplicações durante as últimas décadas.

O advento do computador tem um papel crucial no cenário de desenvolvimento atual em estudos com Lingüística de Corpus. A disponibilidade de computadores trouxe mudanças radicais. Os computadores possibilitaram o trabalho com grande variedade de textos, proporcionando descobertas que extrapolam a intuição de alguns lingüistas Biber (1998).

As principais áreas de interesse da Lingüística de Corpus são: compilação de corpora, desenvolvimento de ferramentas para análise de corpora, descrição de linguagem, exploração do uso de descrições baseadas em corpora para várias aplicações tal como ensino-aprendizagem de línguas, processamento de linguagem natural por máquinas, reconhecimento de voz e tradução (Kennedy, 1998).

Este trabalho trata de duas áreas específicas de aplicação da Lingüística de Corpus, o ensino-aprendizagem de línguas estrangeiras e o desenvolvimento de ferramentas de análise de corpora.

Para entendê-las de forma adequada, é importante conhecermos o quadro conceitual no qual a Lingüística de Corpus baseia-se. Ele se compõe de uma abordagem empirista e de

uma visão de linguagem como sistema probabilístico, cujo detalhamento é exposto logo após um breve histórico da Lingüística de Corpus.

1.1.2 Breve histórico

A Lingüística de Corpus como a conhecemos hoje faz extenso uso de ferramentas computacionais. Pode-se dizer que a história da Lingüística de Corpus se confunde com o desenvolvimento tecnológico.

Antes do advento do computador, já se fazia uso de corpus. Na Grécia Antiga foi criado o Corpus Helenístico. Na Antiguidade e Idade Média, produziam-se corpora de citações da bíblia. Durante boa parte do século XX, o uso de corpora para descrição da linguagem (Berber Sardinha, 2004).

Os corpora dessas épocas eram coletados, armazenados e analisados manualmente. A dificuldade de se realizar estudos desse tipo era enorme. Mesmo assim, havia grande interesse na coleta e exploração de corpora. É importante ressaltar o papel dos estudos baseados em corpora realizados manualmente pela dificuldade e pelo pioneirismo.

O período considerado crítico para os estudos baseados em corpus se deu com a ‘mudança’ de paradigma da Lingüística, com as idéias de Chomsky, por volta de 1950. Houve uma preferência muito forte por estudos baseados em teorias racionalistas da linguagem. Os estudos empíricos receberam muitas críticas nessa época. As críticas eram relacionadas à necessidade de se coletar dados empíricos e o meio pelo qual se realizava a coleta e a análise desses dados. Um dos argumentos era a falta de confiabilidade em analisar manualmente grandes quantidades de dados lingüísticos (Berber Sardinha, 2004).

Embora o cenário fosse desfavorável, os estudos baseados em corpora não pararam. Muitos pesquisadores continuaram seus estudos por meio de corpora. Firth e os neo-firthianos defendiam a descrição da linguagem por meio de dados reais. O corpus SEU (*Survey of English Usage*), por exemplo, foi compilado e etiquetado manualmente em 1959. O SEU influenciou a criação de corpora eletrônicos e serviu para o desenvolvimento de etiquetadores computadorizados contemporâneos.

Com o advento do computador nos anos de 1960 e a queda de prestigio das pesquisas puramente racionalistas, o cenário começou a mudar. Segundo Berber Sardinha (2004:2), o lançamento do corpus Brown em 1964, com 1 milhão de palavras, é considerado como o fato propulsor do desenvolvimento da Lingüística de Corpus. O corpus Brown é o pioneiro dos corpora eletrônicos por ter nascido em um período ainda desfavorável para os estudos empiristas e, também, pela dificuldade de compilação em computadores *mainframe*.

A popularização dos estudos com corpora ocorreu nos anos de 1980 com o aparecimento dos computadores pessoais. Com o desenvolvimento dos computadores, especificamente o aumento da capacidade de armazenar e processar dados, maiores números de corpora e ferramentas foram disponibilizadas para pesquisas, contribuindo para a consolidação da Lingüística de Corpus.

Atualmente, a Lingüística de Corpus exerce influencia em várias áreas da Lingüística Aplicada (Ensino de Línguas, Tradução, Análise do Discurso, Lexicografia, etc), fornecendo subsídios teóricos e metodológicos que possibilitam o estudo de vários aspectos da linguagem. Segundo Kennedy (1998:1), a Lingüística de Corpus é uma fonte de evidência para melhorar as descrições da estrutura e uso de línguas e para várias aplicações, incluindo o processamento de linguagem natural por maquina e o entendimento de como aprender ou ensinar uma língua.

1.1.3 Abordagem empirista e visão probabilística da linguagem

Na Lingüística de Corpus, tem-se como central a noção de linguagem enquanto sistema probabilístico. Segundo essa noção, a linguagem “é padronizada (*patterned*)” (Berber Sardinha, 2004:31), ou seja, os traços lingüísticos não ocorrem de forma aleatória. Tal concepção de linguagem é essencial para o presente estudo.

O uso de uma abordagem empirista também é central para a Lingüística de Corpus. Diferentemente de uma abordagem racionalista, do ponto de vista lingüístico, a abordagem empirista privilegia o estudo da linguagem por meio da observação de dados lingüísticos reais, ou seja, a língua em uso.

Sob o ponto de vista da abordagem racionalista, não há necessidade de coletar dados empíricos para o estudo da linguagem. Segundo tal visão, todos os dados necessários já

estariam na cabeça do lingüista, sendo acessados por meio de introspecção. A intuição do lingüista é a ferramenta principal para desvendar o funcionamento da língua. A principal preocupação do lingüista nesta abordagem é a definição de regras gramaticais universais que regem a língua.

Um grande defensor da utilização de dados empíricos na Lingüística e também um dos expoentes máximos da Lingüística de Corpus, responsável por grande parte de seu desenvolvimento, é John Sinclair. Para Sinclair (1987), há dois modelos de interpretação da língua: princípio de escolha livre e princípio idiomático.

O primeiro refere-se ao princípio de escolha livre, que é essencialmente sintagmático, no qual a língua é observada e descrita como o preenchimento de lacunas gramaticais onde qualquer palavra pode ocorrer, tendo como único parâmetro a adequação gramatical. Tal princípio rege grande parte das gramáticas e está extremamente ligado às concepções do racionalismo de Chomsky.

Já o segundo princípio, o idiomático, defende que as palavras não ocorrem de forma randômica e o parâmetro gramatical não é suficiente para dar conta de tal fenômeno. A organização da língua é influenciada pelas relações no mundo. As escolhas que fazemos na língua são restringidas por tais relações, que podem, por exemplo, estar no nível do registro, gênero ou contexto em particular. Ainda neste princípio, gramática e vocabulário (léxico) são tratados como um único fenômeno para dar conta do funcionamento da língua. É com o princípio idiomático que a Lingüística de Corpus trabalha.

Os estudos da linguagem por meio de corpora durante algum tempo não eram privilegiados. Mesmo assim, nunca deixaram de existir. A disponibilidade de corpora devido aos avanços tecnológicos, marcadamente pelo desenvolvimento do computador impulsionou o que hoje chamamos de Lingüística de Corpus.

Por valorizar a língua como ela realmente é, e não como nossa intuição acha que deve ser, desvendando fatos da língua que antes não eram evidenciados, a Lingüística de Corpus vem tomando proporções cada vez maiores, encontrando aplicações em vários campos da linguagem, como é o caso da presente pesquisa que pretende aplicações da Lingüística de Corpus para o ensino de línguas.

Assim, a escolha da Lingüística de Corpus como principal aporte teórico justifica-se pelo fato de valorizar a língua em uso, estabelecendo que, para entender seu funcionamento, é necessária a observação de seu uso real, o que vai ao encontro dos objetivos da presente pesquisa.

1.1.4 Lingüística de Corpus e o computador

A Lingüística de Corpus não começou com o desenvolvimento dos computadores, mas não há duvidas de que os computadores deram à Lingüística de Corpus um grande impulso.

Para Kennedy (1998:5), embora a importante contribuição das análises manuais, a disponibilidade de computadores a partir da metade do século XX trouxe mudanças radicais para o estudo baseado em textos.

Os desenvolvimentos tecnológicos mudaram a maneira como trabalhamos com corpus. O computador introduziu velocidade, precisão, confiabilidade estatística e habilidades para lidar com grandes quantidades de dados (Kenedy,1998). Segundo Fox (1998:25), atualmente não há restrições para o número de textos em um corpus.

Berber Sardinha (2004:85) cita uma metáfora utilizada por Hoey (1993) para ressaltar o papel do computador na Lingüística de Corpus:

> *"O desenvolvimento do computador com memória poderosa seria para a lingüística o que o desenvolvimento do microscópio com lentes poderosas foi para a biologia – uma oportunidade não somente de ampliar nosso conhecimento, mas de transformá-lo."*

De acordo com Kennedy (1998), os computadores facilitaram o desenvolvimento de bases matemáticas para o processamento automático de linguagem natural, além de trazer para os estudos lingüísticos um grande grau de precisão, importante para todas as ciências.

Berber Sardinha (2004:85) defende um maior emprego de computadores na investigação da linguagem por três razões. Primeiro, porque os computadores garantem a consistência da análise em tarefas que não sejam adequadas para o pesquisador, como contagens exaustivas de itens lingüísticos. Segundo, por permitirem maior abrangência na

quantidade de dados analisáveis. Terceiro, por possibilitar a descoberta de fatos novos ou contestar crenças estabelecidas sobre a linguagem.

Hunston (2002:3) afirma que o corpus em si não pode fazer muito. O corpus não contém informações novas sobre a linguagem, mas as ferramentas eletrônicas utilizadas para reorganizar as informações contidas no corpus podem oferecer novas perspectivas para o pesquisador.

Dessa forma, o computador é essencial para o desenvolvimento de pesquisas em Lingüística de Corpus. Há uma grande variedade de *software* disponíveis para os estudos de Lingüística de Corpus. A maioria processa os dados de um corpus exibindo freqüências, fraseologias e colocações. Os mais conhecidos são etiquetadores e concordanciadores.

Berber Sardinha (2004) aborda uma série de ferramentas computacionais que podem ser utilizadas em pesquisas em Lingüística de Corpus. Tais ferramentas são citadas em seguida.

O WinHTTrack, um *offline browser*, diponível no sítio http://www.httrack.com/, utilizado para coleta em massa de textos na Internet.

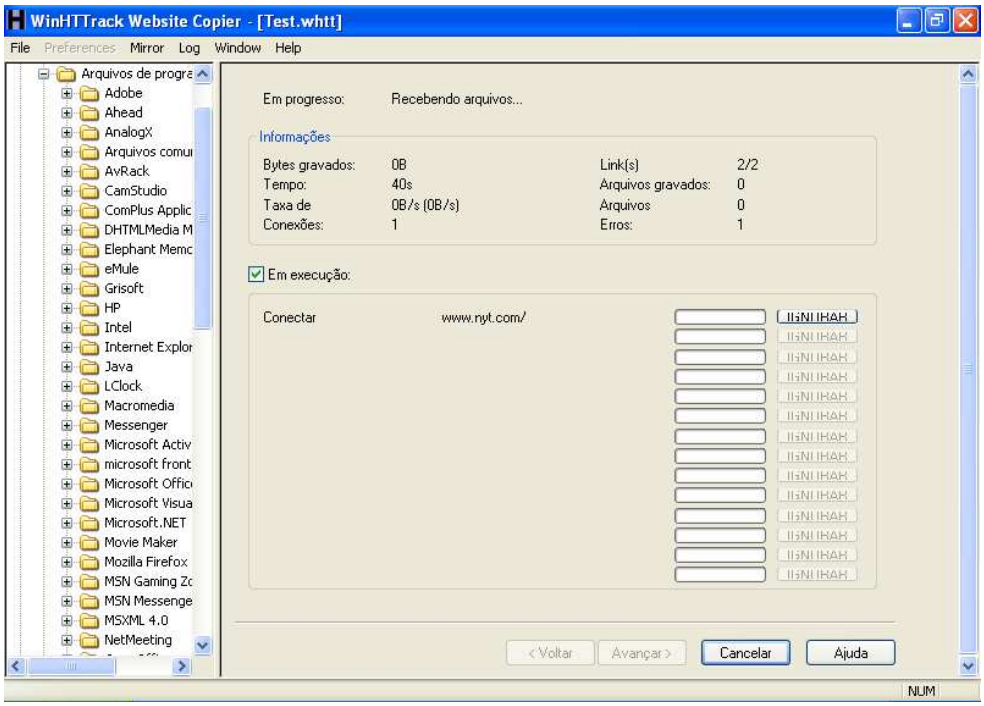


Figura 1.1 – Win HtTrack para coleta em massa de textos na Internet

O Cygwin, um emulador do sistema operacional Unix para Windows, utilizado para manipulação e limpeza de textos por meio de comandos de programação *shell*[1]. A janela abaixo é o terminal onde são digitados os comandos.

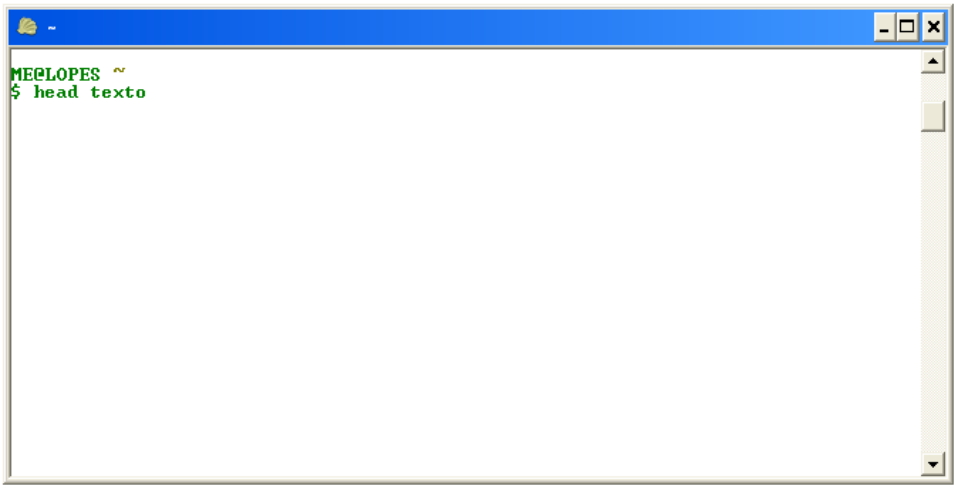


Figura 1.2 – Janela para digitação de comandos do Cygwin

Etiquetadores on-line e *desktop*[2], tais como o etiquetador VISL e o etiquetador Brill, utilizados para inserir automaticamente etiquetas que podem indicar, dependendo do tipo de etiquetagem, marcações morfossintáticas, sintáticas, semânticas ou discursivas.

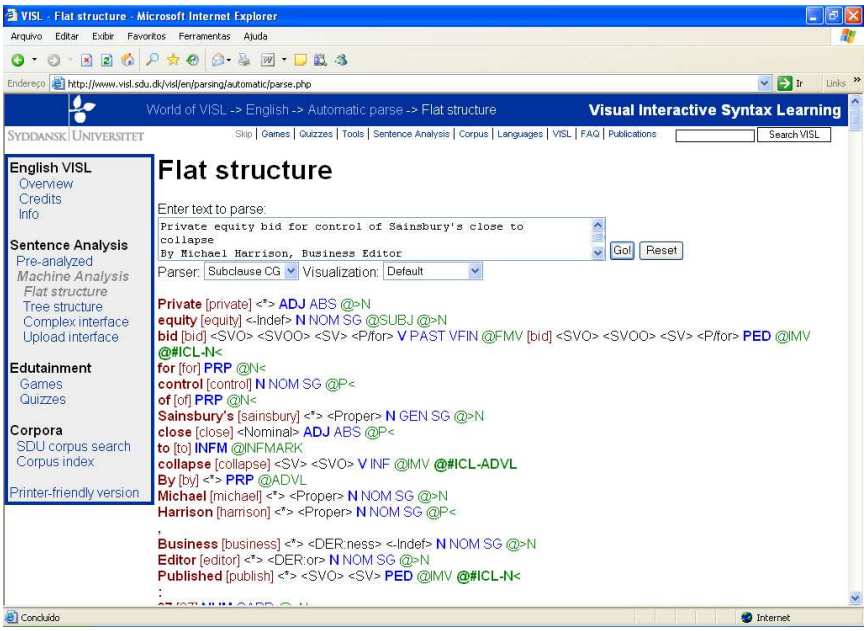


Figura 1.3 – Exemplo do etiquetador on-line VISL

[1] Termo que engloba programas de sistemas do tipo Unix, sistema operacional.
[2] Utilizado aqui para significar programas que não precisam de uma conexão à Internet.

Os concordanciadores são programas que permitem a busca e visualização das ocorrências de uma palavra ou expressão em um corpus (ou corpora), em que a palavra de busca fica centralizada em destaque. A figura abaixo mostra linhas de concordância da palavra ‘good’ no MicroConcord, um concordanciador para o sistema operacional DOS.

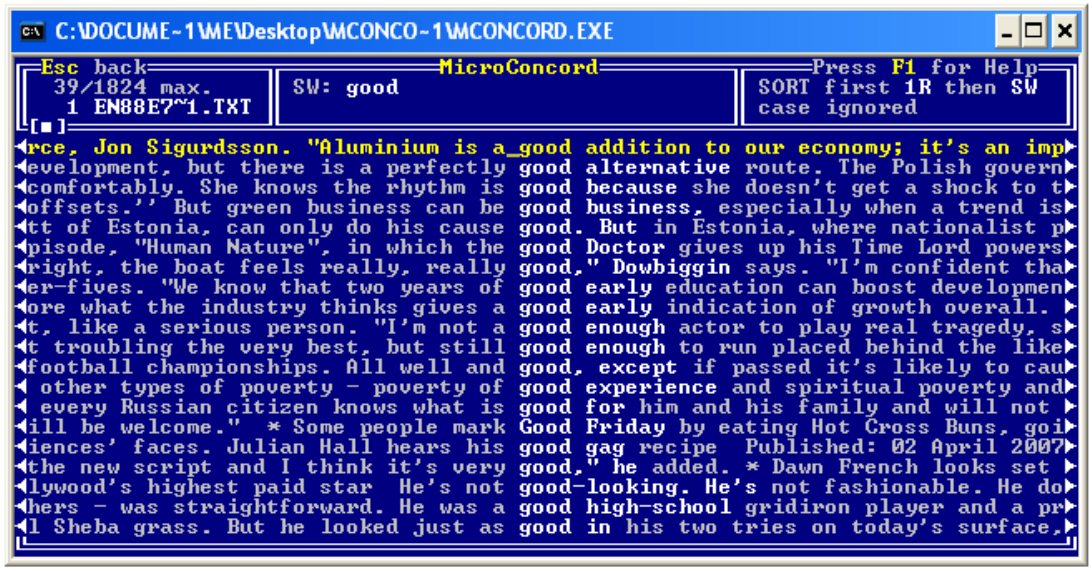


Figura 1.4 – Concordanciador MicroConcord

Existem programas que, além de concordâncias, também fornecem um conjunto de ferramentas que possibilitam a extração de lista de palavras e palavras-chave. Uma lista de palavras pode ser definida como uma listagem de formas/vocábulos e suas respectivas freqüências. Uma lista de palavras-chave é feita a partir da comparação entre uma lista de palavras de um determinado corpus com a lista de palavras de um corpus de referência, geralmente muito maior. A lista de palavras-chave mostra o contraste das freqüências, destacando as palavras que tiveram suas freqüências estatisticamente diferentes (Berber Sardinha, 2004).

Um dos mais utilizados em pesquisas lingüísticas é o Wordsmith tools, que possui três ferramentas principais: Concord, Wordlist e KeyWords.

A ferramenta Concord é utilizada para produção de linhas de concordância. A figura 1.5 mostra um exemplo de visualização.

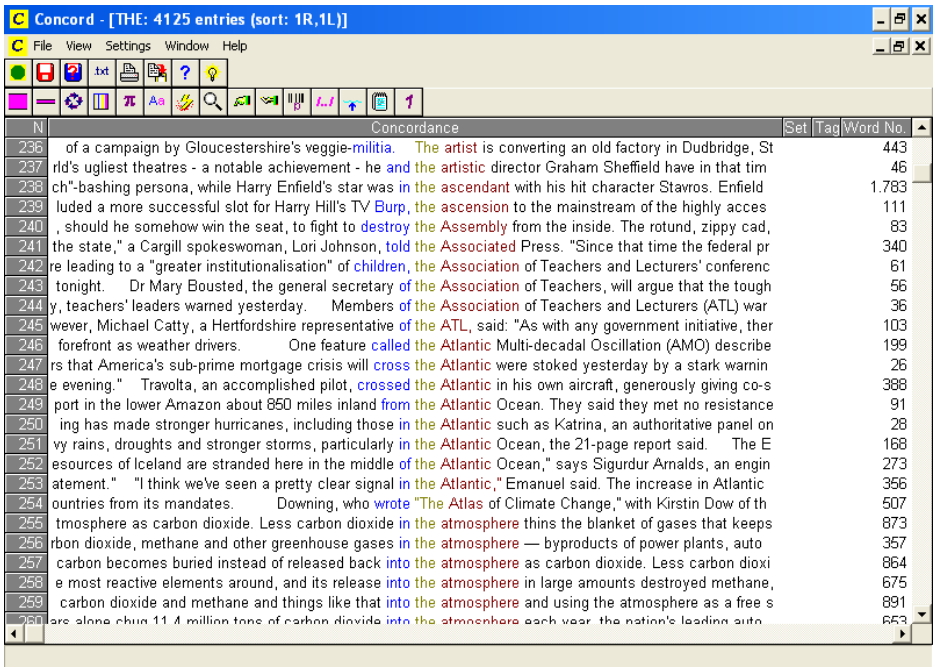


Figura 1.5 – Linhas de concordância na ferramenta Concord

A ferramenta Wordlist é utilizada para produção de listas de palavras. São fornecidos três tipos de listas: lista de estatísticas, lista alfabética e lista de freqüências. A figura 1.6 mostra um exemplo de lista de freqüências.

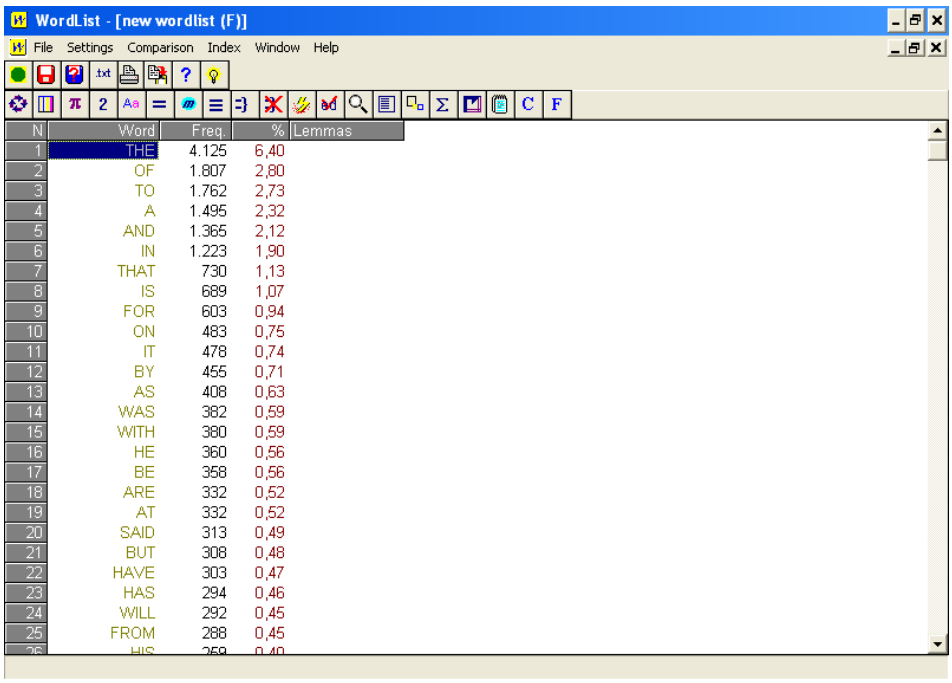
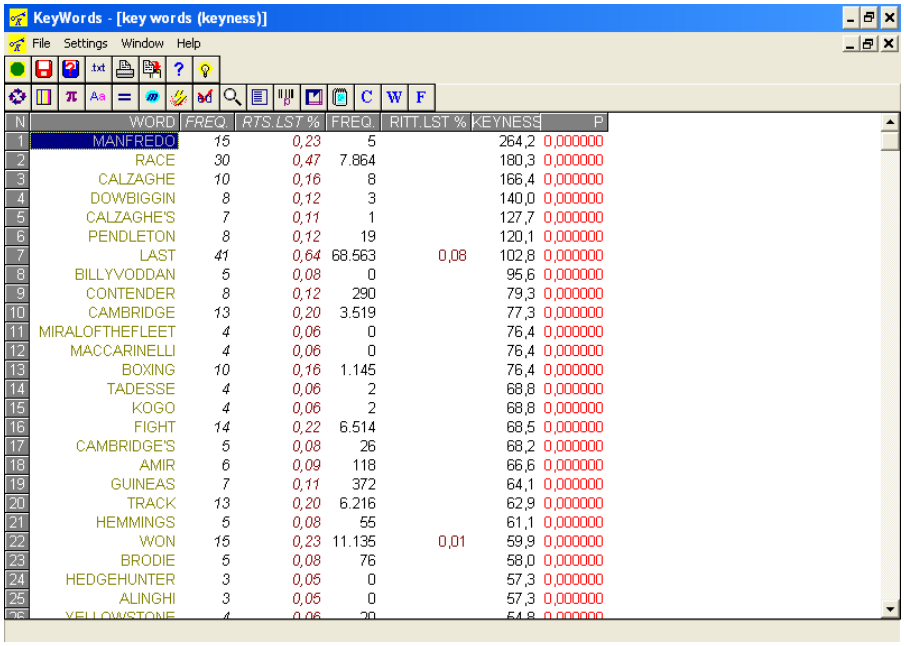


Figura 1.6 – Lista de freqüências na ferramenta Wordlist

A ferramenta KeyWords é utilizada para extração de palavras-chave. A figura 1.7 mostra um exemplo de lista de palavras-chave de um corpus formado por textos sobre esporte.



N	WORD	FREQ.	RTS.LST %	FREQ.	RITT.LST %	KEYNESS	P
1	MANFREDO	15	0,23	5		264,2	0,000000
2	RACE	30	0,47	7.864		180,3	0,000000
3	CALZAGHE	10	0,16	8		166,4	0,000000
4	DOWBIGGIN	8	0,12	3		140,0	0,000000
5	CALZAGHE'S	7	0,11	1		127,7	0,000000
6	PENDLETON	8	0,12	19		120,1	0,000000
7	LAST	41	0,64	68.563	0,08	102,8	0,000000
8	BILLYVODDAN	5	0,08	0		95,6	0,000000
9	CONTENDER	8	0,12	290		79,3	0,000000
10	CAMBRIDGE	13	0,20	3.519		77,3	0,000000
11	MIRALOFTHEFLEET	4	0,06	0		76,4	0,000000
12	MACCARINELLI	4	0,06	0		76,4	0,000000
13	BOXING	10	0,16	1.145		76,4	0,000000
14	TADESSE	4	0,06	2		68,8	0,000000
15	KOGO	4	0,06	2		68,8	0,000000
16	FIGHT	14	0,22	6.514		68,5	0,000000
17	CAMBRIDGE'S	5	0,08	26		68,2	0,000000
18	AMIR	6	0,09	118		66,6	0,000000
19	GUINEAS	7	0,11	372		64,1	0,000000
20	TRACK	13	0,20	6.216		62,9	0,000000
21	HEMMINGS	5	0,08	55		61,1	0,000000
22	WON	15	0,23	11.135	0,01	59,9	0,000000
23	BRODIE	5	0,08	76		58,0	0,000000
24	HEDGEHUNTER	3	0,05	0		57,3	0,000000
25	ALINGHI	3	0,05	0		57,3	0,000000

Figura 1.7 – Lista de palavras-chave na ferramenta KeyWords

Como já foi mencionado antes, o computador é essencial para pesquisas em Lingüística de Corpus. Existe uma gama de ferramentas eletrônicas, *software*, que auxiliam o pesquisador em suas descobertas por meio da análise e exploração de corpora.

Nesta pesquisa, procuramos aproveitar toda a instrumentação computacional da Lingüística de Corpus no desenvolvimento de um *software* para preparação de atividades didáticas especificamente para o professor de inglês brasileiro.

1.1.5 Lingüística de Corpus e ensino-aprendizagem de línguas

Com a crescente popularização dos estudos da Lingüística de Corpus, ferramentas e suas aplicações, tem havido um grande interesse de pesquisadores em usar a Lingüística de Corpus no ensino de línguas.

Atualmente, conforme Berber Sardinha (2004), podemos resumir a influência do acesso e exploração de corpora no ensino em quatro principais áreas:

1) Descrição da linguagem nativa;

2) Descrição da linguagem do aprendiz;

3) Transposição de metodologias de pesquisa acadêmica para a sala de aula;

4) Desenvolvimento de materiais de ensino, currículos e abordagens.

Já há uma considerável quantidade de trabalhos que utilizam corpora, direta ou indiretamente, para o ensino de línguas.

A Lingüística de Corpus também propicia uma série de abordagens para o ensino de línguas com base na exploração corpora. A seguir, citamos as principais.

O Currículo Lexical, desenvolvido por Dave Willis (1990) e concretizado na série de livros didáticos *Collins Cobuild English Course*, é uma proposta que tem como base os princípios e a metodologia da Lingüística de Corpus.

Principais características:

1. Materiais baseados em corpora;
2. Deve-se ensinar as palavras mais freqüentes da língua;
3. Não há divisão entre léxico e gramática;
4. Ensino por meio de dados reais;
5. Metodologia centrada no aluno;

A Abordagem Lexical, formulada por Michael Lewis (1993 e 1997), tem também como proposta o ensino baseado em conceitos da Lingüística de Corpus, embora não sejam mencionados os corpora ou ferramentas utilizadas.

Principais características:

1. O léxico é desempenha um papel central no conteúdo e na metodologia;
2. Diferentemente do Currículo Lexical, o léxico deve ser ensinado por meio de padrões ou porções (*chunks*).
3. Ensino de padrões em textos (escritos ou falados);
4. Os alunos podem manter cadernos lexicais para registrar os itens que aprendem;

5. Os alunos são expostos a exercícios que envolvem: identificar, comparar, completar, categorizar itens lexicais.

Tim Johns (1991) é o grande defensor da abordagem de Ensino Movido a Dados, uma das propostas mais sólidas para a utilização dos dados provenientes de pesquisa com corpus na sala aula. A proposta visa tornar o aluno um pesquisador.

Principais características:

1. O objetivo é desenvolver no aluno a habilidade da descoberta;
2. O papel do professor é despertar no aluno a autonomia e estratégias de descoberta;
3. O ensino da linguagem é baseado em linhas de concordância;
4. O aluno descobre o funcionamento da língua por meio da observação de dados lingüísticos autênticos indutivamente;
5. O aluno deve ser capaz de perceber padrões léxico-gramaticais.

Além das abordagens de ensino que utilizam dados provenientes de análises de corpora. A Lingüística de Corpus também contribui para o desenvolvimento de materiais didáticos.

Para Thurstun & Candlin (1998), a crescente popularização de corpora e a disponibilidade de programas de análise lingüística vêm oferecendo novas e interessantes direções para o desenvolvimento de materiais didáticos.

Flowerdew (1993) relata o *design* de um curso de *ESP* por meio de ferramentas de exploração de corpora, possibilitando não só o conhecimento dos itens léxico-gramaticais que deveriam ser ensinados, por meio de análises da linguagem que os alunos estariam expostos, mas também a criação de atividades para o próprio curso.

Fox (1998) aponta a disponibilidade de corpora, propiciada pelo desenvolvimento do computador, para a criação de materiais didáticos baseados em língua em uso e as possibilidades de uso dos dados de pesquisa de corpora para ensino de vocabulário contextualizado, gramática e pragmática.

Berber Sardinha (2006) cita uma série de trabalhos que produziram atividades de ensino com corpus em vários âmbitos do ensino de língua estrangeira. No ensino de inglês geral, com os trabalhos de Condi de Souza (2005), Vicentini (2006), Bértoli Dutra (2002) e Amarante (2005). No ensino de Inglês Instrumental, com os trabalhos de Ferrari de Oliveira (2004) e Barbosa (2004). No ensino de língua espanhola, Jacobi (2001) e Alonso (2006).

O presente estudo pretende aproveitar as possibilidades que a Lingüística de Corpus tem a oferecer ao ensino de línguas para o desenvolvimento de uma ferramenta para criação de materiais didáticos.

1.2 Plano de aula baseada em textos

Aqui, mencionamos brevemente os principais pressupostos teóricos e metodológicos que basearam a criação de uma atividade padrão para o ensino de compreensão escrita para o desenvolvimento do *Reading Class Builder*, *software* proposto nesta pesquisa.

A criação foi inspirada na idéia de utilização de uma atividade padrão (*standard exercise*) sugerida por Scott et al. (1984), por meio de uma série de questões abertas que poderiam ser utilizadas com quase qualquer texto.

Para a montagem da atividade, buscamos também as orientações de Nuttal (1981) e Grellet (1981) para o desenvolvimento de atividades de leitura, por proporcionar um quadro conceitual flexível e adaptável aos objetivos da presente pesquisa.

Segundo Nuttal (1981:149), o planejamento de uma aula de leitura baseada em texto deve iniciar com a escolha de um texto adequado para o aluno, de acordo com alguns critérios, como legibilidade, adequação do conteúdo e potencial didático do texto.

De acordo com Nuttal (1981:149), uma aula de leitura baseada em texto pode incluir alguns ou todos os seguintes itens:

1. Utilização de estratégias como *scanning*, *skimming*.

2. Interpretação do texto por meio da utilização de:

   a. informação não-textual;

   b. habilidades para lidar com vocabulário (*word attack skills*);

c. habilidades para lidar com texto (*text attack ski*lls);

d. outras questões ou atividades consideradas necessárias.

3. Produção de algum tipo de resultado com base no texto como um todo.

Com base na noção de atividade padrão e as orientações de Nuttal (1981) para a elaboração de uma aula de leitura baseada em texto, propusemos um modelo de material de ensino de leitura para servir de base para o desenvolvimento do *Reading Class Builder*.

1.3 Processo de desenvolvimento de *software*

Para o desenvolvimento do *software* proposto nesta pesquisa, buscou-se o embasamento teórico em algumas especificações de Engenharia de *Software*. Embora as orientações fornecidas sejam gerais e para o desenvolvimento de sistemas mais complexos, são úteis para o presente contexto de pesquisa, uma vez que podem ser adaptadas.

A Engenharia de S*oftware* é uma disciplina que se preocupa com os aspectos de produção de *software* desde os estágios iniciais de desenvolvimento até o produto final. Sua tarefa é fornecer abordagens sistemáticas e organizadas para o desenvolvimento de *software* (Sommerville, 2001).

De acordo com Sommerville (2001:5), *software* é todo o conjunto de programas e documentações necessárias para que estes programas funcionem corretamente.

Ainda segundo Sommerville (2001), há dois tipos de *software*:

1. Gerais (*generic products*);

2. Personalizados (*customised products*).

A diferença entre esses dois tipos de *software* está bastante relacionada ao público a que se destina o produto final e ao processo de seu desenvolvimento. Os *software* gerais são desenvolvidos geralmente por uma instituição ou organização para serem utilizados por qualquer usuário. Diferentemente, os *software* personalizados são desenvolvidos especificamente para um tipo de público, e todas as etapas e características do *software* são guiadas pelo público em particular.

Para o presente projeto, o tipo de *software* que se deseja desenvolver está inserido na segunda categoria. Suas características foram desenvolvidas de acordo com objetivos para satisfazer um determinado público-alvo, professores de inglês. Contudo, primeiramente é necessário um processo de desenvolvimento para que o projeto.

De acordo com Sommerville (2001:8), um processo de *software* é um conjunto de atividades e resultados associados que produzem um *software*. São quatro as atividades fundamentais:

1. Especificação (*Software specification*);

2. Projeto (*Software development*);

3. Validação (*Software validation*);

4. Evolução (*Software evolution*);

Essas atividades podem estar contidas em um modelo de processo. O modelo de processo é uma descrição simplificada de um processo de *software*. Há vários modelos ou paradigmas de desenvolvimento de *software*, alguns incluem algumas das atividades mencionadas acima. O uso de qualquer modelo depende do tipo de aplicação que se deseja desenvolver.

Há vários modelos de processo de *software*. Os principais modelos são: modelo cascata, desenvolvimento evolucionário, desenvolvimento formal de sistemas e desenvolvimento baseado em reuso.

Tendo em vista as características do tipo de pesquisa e o tempo disponível, optou-se por seguir um modelo simples, contendo apenas as três primeiras atividades mencionadas.

1. Levantamento de características do *software*;

2. Desenvolvimento do *software* por meio de programação;

3. Teste e avaliação do *software*.

Este capítulo apresentou as áreas que forneceram embasamento teórico para o trabalho. No capítulo seguinte, será apresentada a metodologia de pesquisa empregada na

pesquisa, incluindo a descrição dos corpora, bem como a especificação dos procedimentos de desenvolvimento e avaliação do *software*.

Capítulo 2

Metodologia

Neste capítulo, é apresentada a metodologia empregada na pesquisa, incluindo a descrição do público-alvo, dos corpora e ferramentas utilizadas, bem como a especificação dos procedimentos de desenvolvimento do *software* e de análise dos dados.

Inicialmente, são reiterados os objetivos da pesquisa e elencadas as questões que a nortearam. A seguir é apresentado o contexto de pesquisa e são detalhados os procedimentos de desenvolvimento do *software*. Por último, são detalhados os procedimentos de teste e avaliação do *software*.

2.1 Objetivo e questões de pesquisa

A pesquisa teve como objetivo geral o desenvolvimento e avaliação de um *software* para auxiliar o professor de inglês na elaboração de atividades didáticas de leitura com corpus.

Os objetivos da pesquisa são os seguintes:

1- Criar um *software* para preparação de atividades de leitura em inglês como língua estrangeira que possa utilizar textos autênticos e corpora.

2- Avaliar o desempenho técnico do *software* criado à luz dos princípios teóricos e metodológicos utilizados na sua elaboração.

3- Avaliar o uso do *software* entre professores.

A fim de atingir os objetivos, as questões de pesquisa a serem investigadas no projeto são:

1- Quais as características e recursos essenciais o *software* deve possuir, dado o público alvo e o contexto educacional a que se destina?

2- Como se cria um módulo em VB6 para fazer lista de palavras e concordâncias?

3- Como se cria um módulo em VB6 para descobrir as palavras-chave?

4- Como se cria um módulo em VB6 para identificar as palavras cognatas?

5- Como se cria um módulo em VB6 para etiquetar as palavras de um texto?

6- Como se cria um módulo em VB6 para descobrir a densidade lexical de um texto?

7- Como se cria um módulo em VB6 para definir a dificuldade estrutural de um texto?

8- Qual a avaliação do *software*, isto é, até que ponto o produto final atende às especificações de design?

2.2 O contexto da pesquisa

Nesta seção, descreve-se o contexto que sustentou o processo de criação do *software*. Esta descrição também fornece as informações que levaram ao levantamento das características do *software*.

As informações prestadas aqui estão diretamente relacionadas à minha experiência de trabalho, com mestrando, em uma Oficina Pedagógica da Região de São Paulo, em contato com professores da rede pública estadual.

O *software* tem como público-alvo direto professores de língua inglesa e, indireto, alunos do Ensino Médio da rede pública. Para os objetivos da pesquisa, foram levadas em consideração algumas variáveis importantes que influenciaram o desenvolvimento do *software*.

Primeiro, a acessibilidade de professores e alunos ao computador. Parece que ainda não são todas as escolas públicas que possuem computadores ou que o laboratório de informática esteja funcionando plenamente. Por diversos fatores, pode ser também uma dificuldade para professores e alunos o acesso ao computador na escola. Grande parte dos professores tem acesso somente em casa, e somente alguns poucos alunos têm acesso a um computador em suas residências.

Segundo, o grau de conhecimento digital. Pode ser uma realidade o desconhecimento digital de muitos professores e também alunos, mas o problema tende a ser maior para alunos, devido a poucas oportunidades de acesso ao computador.

Terceiro, o grau de conhecimento de professores sobre a Lingüística de Corpus. Como são professores que já se formaram há algum tempo e por ser ainda uma área em desenvolvimento, é certo o desconhecimento em relação aos pressupostos teóricos e práticos da Lingüística de Corpus e sua aplicação ao ensino de línguas, podendo ser necessário um trabalho de conscientização.

Foram consultados também as orientações nos Parâmetros Curriculares Nacionais a fim de evidenciar as sugestões de ensino. De acordo com os PCN+, Língua Estrangeira Moderna, a competência primordial do ensino de línguas estrangeiras modernas no ensino médio deve ser a da leitura, e por decorrência a da interpretação (Brasil, 2002:97). É sugerido que o professor trabalhe a partir de três frentes: estrutura lingüística, aquisição de repertório vocabular, leitura e interpretação de textos, utilizando estratégias de leitura como *skimming, scanning, prediction*.

Tendo em vista as características do público-alvo e os possíveis obstáculos evidenciados: a falta de acesso ao computador na escola, o possível desconhecimento digital e familiaridade com a Lingüística de Corpus, optou-se por desenvolver um *software* exclusivo para o professor, no qual este possa preparar aulas e imprimi-las para utilizar em sala de aula, já que a disponibilidade de computadores na escola ainda não é a desejável, e ainda há necessidade de conscientização do professor em relação à Lingüística de Corpus.

2.3 *Desiderata*

Antes de iniciar o desenvolvimento do *software*, relacionamos algumas características que o *software* deveria possuir (chamadas de ‘desiderata’ na literatura de Processamento de Linguagem Natural). As características desejadas foram formuladas de acordo com algumas informações sobre o público-alvo e o contexto de ensino a que se destina o *software*. É importante mencionar também que o desenvolvimento do *software* foi marcado por uma série de testes e versões com meu grupo de pesquisa, também professores de inglês.

Relacionamos abaixo algumas das características que guiaram o desenvolvimento do

software.

- O *software* deve ser desenvolvido para a para a plataforma Windows, visto que a maioria dos usuários utiliza esta plataforma.
- Não deve precisar de uma conexão à internet para ser utilizado.
- Deve possuir requisitos mínimos de *hardware* para poder funcionar.
- Deve possuir um instalador automático para que o usuário não tenha que instalar manualmente.
- Deve possuir uma interface amigável e lógica.
- Deve ser adequado para os objetivos do contexto educacional a que se destina.

2.4 Programação para fins de pesquisa lingüística

Atualmente, como já mencionamos, há uma variada gama de *software* para a análise de corpus disponíveis no mercado. Muitos disponibilizam uma série de recursos necessários para o lingüista de corpus, contudo nem sempre tais recursos atendem às necessidades do lingüista, pois também são variados os tipos de pesquisas, e pode-se precisar de diferentes tipos de análises e exibição de resultados.

Para resolver este problema, na maioria das vezes, o lingüista tem que adaptar algumas funções da ferramenta que está utilizando ou combinar mais de uma ferramenta para obter os resultados desejados quando não há um *software* específico que possibilite as análises requeridas. Tais procedimentos podem consumir tempo, fator essencial para toda pesquisa, e não propiciar os resultados desejados.

É nesse contexto que a habilidade de desenvolver *software* se torna imprescindível para o lingüista de corpus. Segundo Biber (1998:254), a capacidade do computador de realizar complexas análises é muito grande, maior que a de *software* já disponíveis, e o pesquisador que saiba escrever programas pode tirar vantagem de toda essa capacidade.

Um bom exemplo é o conjunto de ferramentas de análise de corpora on-line criadas por Berber Sardinha (2004). A caixa de ferramentas (*Toolkit*) disponível no sítio http://www2.lael.pucsp.br/corpora/index.htm reúne, além das principais funções realizadas por concordanciadores, ferramentas avançadas para estudos relacionados a metáforas, tradução, pronúncia e outras áreas. A maioria das ferramentas foi criada a partir das necessidades de pesquisas em Lingüística Aplicada.

Embora muitas pessoas pensem que programar um computador seja algo extremamente complexo, sendo necessários conhecimentos matemáticos e técnicos avançados, a realidade é menos aterrorizante e, pelo contrário, não se constitui como algo de outro mundo.

Biber (1998) cita dois requisitos básicos para que o lingüista possa escrever seus próprios programas de análise de corpus.

O primeiro requisito, considerado como ferramenta básica, é o conhecimento de uma linguagem que o computador possa entender. São muitas as linguagens de programação (Perl, C, C++, Basic, etc), basta que o lingüista conheça uma delas para dizer ao computador o que fazer.

O segundo requisito refere-se ao conhecimento lingüístico, necessário para dar instruções corretas ao computador. Os dois tipos de conhecimentos são essenciais para que o programa funcione corretamente.

Para a presente pesquisa, utilizou-se o programa Microsoft Visual Basic 6, por ser uma linguagem de programação fácil, que possibilita o desenvolvimento rápido de projetos, e também os conceitos e aportes teóricos e práticos da Lingüística de Corpus.

2.5 Descrição dos corpora utilizados no estudo

Foram utilizados no desenvolvimento do *software* dois tipos de corpora, um de treinamento e um geral de referência. O corpus de treinamento foi utilizado nos testes das funções do programa durante a codificação do *software*. O corpus de referência foi utilizado na criação de funções como etiquetagem de palavras e extração de palavras-chave. Ao contrário de outras pesquisas que fazem uso de corpora, não foi objetivo do estudo fazer a análise dos corpora

coletados. O uso dos corpora nesta pesquisa está relacionado ao desenvolvimento e teste do *software*.

É fornecida agora uma breve explicação a respeito destes tipos e de sua utilização nesta pesquisa.

2.5.1 O corpus de treinamento

Durante o desenvolvimento do *software* proposto nesta pesquisa, houve a necessidade constante de testar as ferramentas e os códigos, a fim de verificar sua eficiência para minimizar a quantidades de erros e garantir o bom funcionamento de cada função desempenhada. Para possibilitar os testes, foi necessária a compilação de um corpus de treinamento.

Para compor esse corpus, foram coletados mais de 80 textos de diversos gêneros disponíveis na Internet. Os textos encontrados na Internet em formato .html foram convertidos para o formato texto (comumente denotados pela terminação .txt no sistema Windows) e então salvos em uma pasta. O corpus coletado possui os seguintes dados estatísticos: 19.067 *tokens* (itens ou ocorrências) e 2.902 *types* (formas ou vocábulos).

Quanto à sua tipologia, segundo critérios listados por Berber Sardinha (2004:20-22), cabe esclarecer que se trata de um corpus:

- escrito;
- contemporâneo (representa o período de tempo corrente);
- de amostragem;
- de língua nativa (os autores são falantes nativos);
- de treinamento ou teste (construído para permitir o desenvolvimento de aplicações e ferramentas de análise).

É importante deixar claro que a coleta do corpus de treinamento está relacionada apenas à necessidade de possuir dados autênticos para os testes durante a programação do *software*. Assim, os textos coletados para este fim poderiam pertencer a qualquer assunto, gênero ou registro. As únicas restrições foram: os textos deveriam ser autênticos e estar em formato eletrônico para facilitar seu processamento.

2.5.2 O corpus de referência

O corpus de referência utilizado na pesquisa foi o *British National Corpus* (BNC), lançado em 1995 e que possui 100 milhões de palavras, das quais 90% são de inglês escrito e 10% de inglês oral.

O corpus de referência possui algumas características que diferem das do corpus de treinamento, ou seja, o corpus é composto por linguagem oral e escrita, de amostragem geral, composto por vários tipos de texto, para ser representativo da língua inglesa e possui a finalidade de ser usado para contrastar com outros corpora.

A utilização do corpus de referência é central para esta pesquisa. A partir dele, foram obtidas três listas: lista de freqüência das palavras, lista de palavras etiquetadas e uma lista com palavras parecidas com o português (cognatas). As listas foram incluídas no *software* para desempenhar funções específicas como: gerar lista de palavras-chave, fazer a etiquetagem de textos e identificar palavras cognatas.

Além das listas de palavras, os textos do corpus de referência foram utilizados para a extração de linhas de concordância de palavras selecionadas pelo *software* para a preparação de uma aula na demonstração realizada no capítulo seguinte.

2.6 Procedimentos de desenvolvimento do *software*

Os passos para o desenvolvimento do *software* por meio de programação em Visual Basic foram:

1. Especificação e descrição dos componentes a serem criados.
2. Criação da interface.
3. Codificação.

Após a criação dos componentes do *software* em Visual Basic, fizemos a compilação de todas as suas partes com uma ferramenta específica para criar o arquivo de instalação.

Mais detalhes sobre a codificação do *software* são fornecidos no próximo capítulo.

2.7 Procedimentos de teste e avaliação do *software*

Em primeiro lugar, apresentamos o *software* construído por meio de uma demonstração de sua utilização. Em seguida, avaliamos as informações geradas pelo *software*. Finalmente, apresentamos os relatos de professores que utilizaram o *software*.

Para avaliar o desempenho do *software*, utilizamos as mesmas medidas de desempenho utilizadas por Berber Sardinha (1997) e Beeferman et al. (1997), entre outros, para avaliar análises feitas automaticamente pelo computador: precisão (*precision*) e abrangência (*recall*).

Valores altos de precisão indicam que a maioria dos itens identificados é relevante. Valores altos de abrangência indicam que o programa identificou a maioria dos itens relevantes disponíveis.

Os valores de precisão e abrangência são geralmente utilizados em análises de segmentação de itens. A escolha dessas medidas é justificada pela natureza das análises feitas pelo *software* apresentado.

Para calcular valores de abrangência, dividimos o número de itens identificados pelo número de itens identificados mais o número de itens não identificados.

Os valores de precisão são calculados a partir da divisão do número de itens identificados corretamente (relevantes) pela soma do número de itens identificados corretamente e incorretamente (irrelevantes). Os valores resultantes são multiplicados por cem para obter a porcentagem.

A fórmula utilizada para calcular a abrangência é:

A: Número de itens identificados.
B: Número de itens relevantes não identificados.

$$\text{ABRANGÊNCIA} = \frac{A}{A+B} \times 100$$

Figura 2.1 – Fórmula para calcular o valor de abrangência

Por exemplo, suponhamos que o texto tenha 10 palavras cognatas, e o programa identificou 15, mas apenas 7 das palavras identificadas são realmente cognatas. O número de itens identificados seria 15 (A) e o número de itens relevantes não identificados seria 3 (B). Para calcular o valor de abrangência, faríamos: (15÷(15+3))x100. O resultado seria 83,33% de abrangência.

A fórmula utilizada para calcular a precisão é:

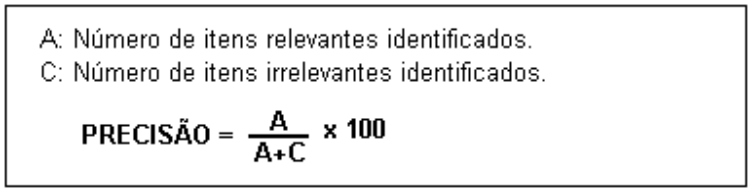


Figura 2.2 – Fórmula para calcular o valor de precisão

Continuando com o exemplo dado, os valores necessários para calcular o valor de precisão são o número de itens relevantes identificados, 7 (A), e o número de itens irrelevantes identificados, 3 (C). Faríamos o seguinte cálculo: (7÷(7+3))x100. O resultado seria 70% de precisão.

Para comparar com os valores gerados pelo programa, fizemos uma contagem manual de todos os dados gerados pelo programa.

A fim de qualificar os valores percentuais obtidos, criamos uma escala de valores:

Valores	**Avaliação do desempenho**
0% a 59%	ruim
60% a 69%	regular
70% a 79%	bom
80% a 89%	muito bom
90% a 99%	excelente
100%	perfeito

Quadro 2.1 – Escola de valores para avaliação das funções do *software*

Após a avaliação do desempenho das funções do *software*, apresentamos também as primeiras impressões do uso do *software* por professores.

Neste capítulo, foi apresentada a metodologia empregada na pesquisa, incluindo a descrição dos corpora, os procedimentos de desenvolvimento e avaliação do *software*. No próximo capítulo, detalhamos o desenvolvimento do *software* por meio de programação.

Capítulo 3

Desenvolvimento do *Software*

Neste capítulo, são detalhados os procedimentos de desenvolvimento do *software* por meio de programação. Apresentamos primeiramente a linguagem de programação e ferramentas utilizadas. Em seguida, detalhamos os procedimentos de codificação do *software*.

3.1 Linguagem de programação e ferramentas utilizadas

Nesta seção, apresentamos a linguagem de programação e as principais ferramentas utilizadas no desenvolvimento do *software*, bem como explicações sobre convenções e funções da linguagem de programação.

3.1.1 Visual Basic 6

Para a programação do *software* proposto nesta pesquisa, utilizamos o *Microsoft Visual Basic 6*, produzido pela empresa Microsoft, parte integrante do pacote *Microsoft Visual Studio*, como a principal linguagem de programação.

O *Visual Basic 6* é um *software* para o desenvolvimento de aplicações para ambiente Windows, baseado na linguagem *Basic*.

A linguagem do *Visual Basic* é uma das mais utilizadas por programadores, por ser uma ferramenta simples e poderosa para o desenvolvimento de aplicativos em Windows. Pode-se criar aplicações inteiras em pouco tempo com conhecimento mínimo em programação.

Como o próprio nome diz, grande parte da programação é feita visualmente, adicionando janelas, botões, figuras e outros controles[1] existentes no ambiente Windows.

[1] São objetos gráficos, padronizados do *Visual Basic* ou criados pelo usuário, que podem ser inseridos em formulários para criação da interface, tal como botões e caixas de texto.

Uma outra característica do *Visual Basic*, na minha opinião, é a facilidade para escrever os códigos. A maioria das funções e comandos utilizados é parecida com palavras do próprio inglês, tal como *Exit*, *End*, *Replace* e outras.

O processo de desenvolvimento de aplicações no *Visual Basic* é regido por três procedimentos básicos. Primeiro, desenvolve-se a interface gráfica, adicionando controles ao formulário[2]. Em seguida, definimos as propriedades[3] dos controles. Por fim, escrevemos o código para que o programa execute as funções desejadas.

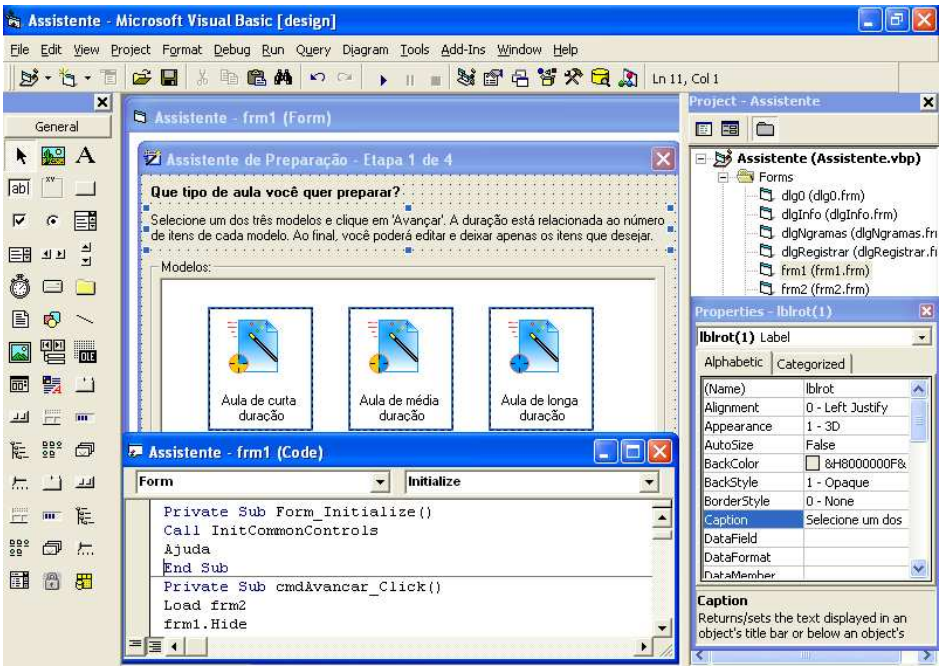


Figura 3.1 – Ambiente de desenvolvimento do Microsoft Visual Basic 6

Atualmente, as novas versões do *Visual Basic* são o *Visual Basic.NET* e o *Visual Basic 2005*. Optou-se por desenvolver o *software* em *Visual Basic 6* por ser a versão mais conhecida pelo pesquisador e por possuir um número maior de materiais de referência para sua utilização.

[2] Janela onde são inseridos os controles, base para a criação da interface.
[3] Podem ser definidas como as características ou atributos de um objeto. Podem ser acessadas visualmente em uma caixa de propriedades ou a partir da codificação em uma janela de códigos.

3.1.1.1 Criação da interface com o usuário no Visual Basic

Para criar a interface com o usuário, utilizamos três componentes essenciais do *Visual Basic*: A barra de controles, os formulários e a caixa de propriedades.

Os três componentes são mostrados abaixo:

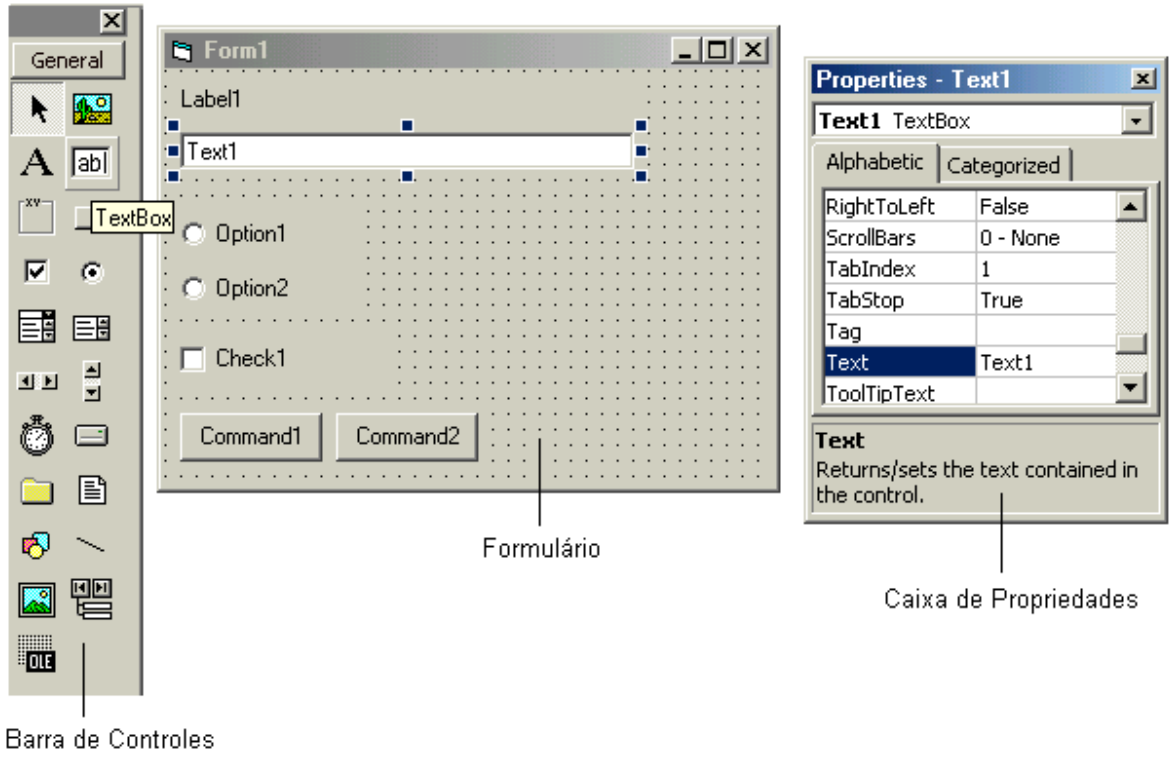


Figura 3.2 – Principais ferramentas para a criação da interface com o usuário

Os procedimentos de criação da interface com o usuário foram dois. Primeiro, inserimos os controles no formulário de forma a desenhar a interface. Para isso, clicamos no controle desejado e, depois, em qualquer área do formulário. Tendo inserido os controles no formulário, definimos as propriedades dos controles inseridos como devem ser iniciados quando o programa for executado. A definição das propriedades dos controles pode ser feita visualmente na caixa de propriedades ou na janela de códigos.

Após a criação da interface com o usuário, o próximo passo para o desenvolvimento da aplicação foi a codificação dos eventos.

3.1.1.2 Codificação

O Visual Basic é uma linguagem orientada a objetos e eventos. Os objetos podem ser definidos aqui como controles e classes formadas por conjuntos de funções. Os eventos podem ser considerados como determinadas ações relacionadas ao objeto. Cada evento cria uma sub-rotina que podemos chamar de procedimento. É dentro do procedimento que digitamos a codificação para que o programa execute as ações que desejamos quando o evento ocorrer.

A maneira mais fácil de se criar um procedimento é clicar duas vezes no objeto. Uma outra opção seria digitar a estrutura do procedimento. Há também a possibilidade de criar o procedimento a partir do menu ‘Tools’, opção ‘Add Procedure’.

Na figura abaixo, mostramos um exemplo de código para imprimir a palavra ‘Teste’ na tela quando o usuário clicar no botão ‘Command1’. O objeto é um ‘CommandButton’, o evento é ‘Click’ e o comando utilizado é ‘Print’. O código é digitado em uma janela que pode ser chamada de janela de código.

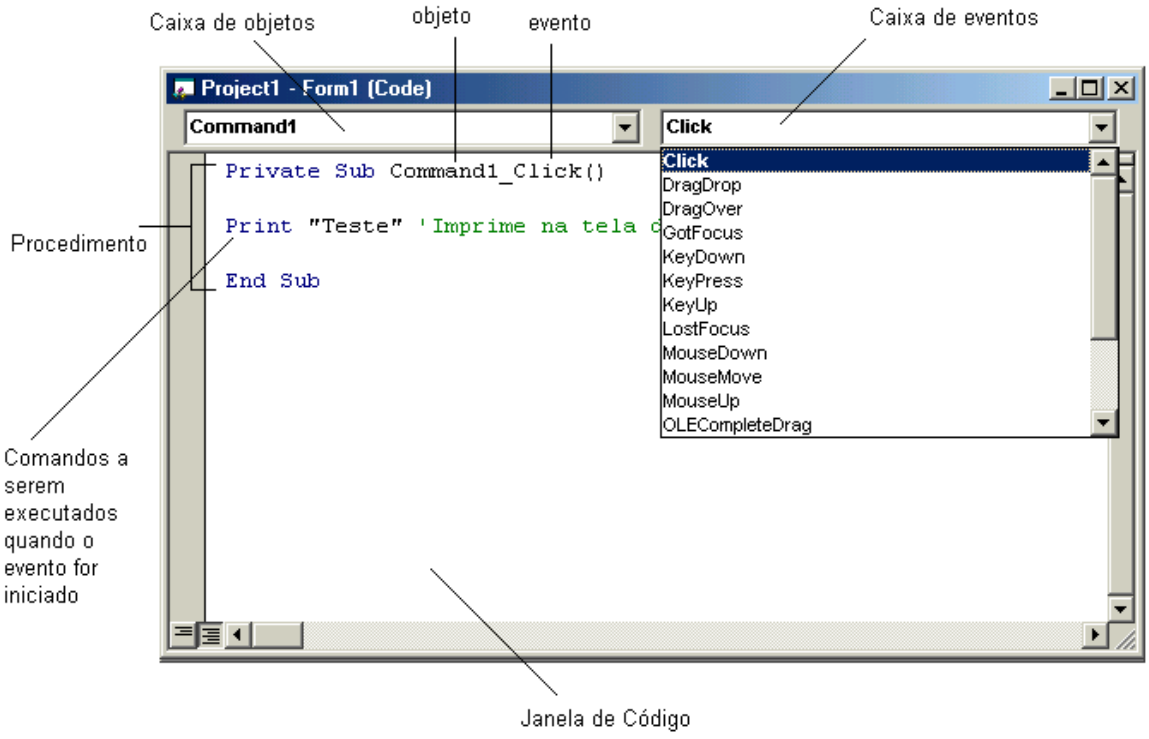


Figura 3.3 – Janela de código

A codificação do projeto foi feita basicamente dessa forma. Primeiro, a inserção dos controles. Em seguida a definição de suas propriedades. Por fim, a codificação de seus eventos.

Passamos, agora, para a descrição dos recursos de tratamento de texto disponíveis no *Visual Basic 6.*

3.1.1.3 Recursos para tratamento de textos no Visual Basic

O *Visual Basic 6* possui uma gama de recursos para o tratamento de textos e seqüências de caracteres: funções, operadores de comparação e o objeto RegExp (expressões regulares), entre outros. Aqui, descrevemos as principais funções e os objetos utilizados na codificação do *software*.

3.1.1.3.1 Funções de manipulação de *Strings* do Visual Basic 6

O Visual Basic 6 possui uma série de funções de manipulação de *Strings*[4], seqüência de caracteres, que são úteis para o tratamento de textos.

As principais funções de manipulação de *String* do *Visual Basic* utilizadas para o desenvolvimento do *software* são:

Len – Retorna o número de caracteres de uma variável do tipo *String*.

Chr – Retorna um caractere associado ao código ASCII[5].

Asc – Retorna um valor numérico que representa o código ASCII de um caractere.

Space – Retorna uma quantidade de espaços em branco indicada por um número.

String – Repete um determinado caractere indicado por um número.

Trim – Remove os espaços em ambos os lados de um texto ou conjunto de caracteres.

---

[4] Variáveis do tipo *String* podem conter somente texto.
[5] ASCII (*American Standard Code for Information Interchange*) é uma codificação para caracteres baseada no alfabeto do inglês.

Rtrim – Remove os espaços à direita de um texto ou conjunto de caracteres.

Ltrim – Remove os espaços à esquerda de um texto ou conjunto de caracteres.

Lcase – Converte todos os caracteres alfanuméricos para minúsculas.

Ucase – Converte todos os caracteres alfanuméricos para maiúsculas.

Left – Retorna uma determinada quantidade de caracteres a partir da esquerda definida por um valor numérico.

Right – Retorna uma determinada quantidade de caracteres a partir da direita definida por um valor numérico.

Mid – Retorna um conjunto de caracteres de um texto, iniciando por uma posição determinada por um valor numérico.

Istr – Retorna um valor numérico que indica a posição da primeira ocorrência de um conjunto de caracteres especificado em um texto.

Split – Divide um texto ou conjunto de caracteres a partir de um caractere utilizado como delimitador. Os resultados são armazenados em um vetor (*Array*).

Join – Concatena um texto ou conjunto de caracteres em uma única expressão.

Filter – Procura em um vetor uma determinada expressão e retorna os resultados em um outro vetor ou lista especificados.

StrReverse – Retorna um texto com os caracteres em ordem invertida.

Replace – Substitui as ocorrências de um texto ou conjunto de caracteres por outro valor.

Embora exista um grande número de funções de manipulação de textos no Visual Basic 6, que permitam uma gama de possibilidades para a criação das funções executadas pelo *software*, foi necessário utilizar funções externas do Windows em API's[6] (*Application Programming*

[6] Interface de Programação de Aplicativos composta por um conjunto de funções e rotinas que permitem acesso a recursos de um sistema operacional para o desenvolvimento de aplicações.

Interface), para gerenciar o uso de memória em funções de manipulação de textos.

A alocação e liberação de memória em funções internas de manipulação de textos no Visual Basic 6 são feitas automaticamente, sem que o programador tenha que se preocupar. Esse benefício tem um preço: as funções, quando utilizadas com grandes quantidades de texto, tendem a deixar o programa mais lento ou ocasionar problemas no sistema por falta de memória virtual.

Para solucionar o problema, visto que o *software* deve manipular grandes quantidades de texto, foi utilizado um conjunto de funções que utilizam a API *Copymemory*. As principais funções utilizadas no projeto são:

- ***Malloc*** – para alocar dados do tipo *String* diretamente na memória.
- ***Retmemory*** – para retornar os dados alocados na memória.
- ***Freememory*** – para limpar os dados da memória após o processamento de funções.

3.1.1.3.2 Operadores para comparação de dados do tipo *String*

Para comparar valores do tipo *String*, utilizamos os seguintes operadores:

Operador	Descrição
<	Menor que
<=	Menor que ou igual a
=	Igual a
<>	Diferente de ou não igual a
>=	Maior que ou igual a
>	Maior que
LIKE	Compara com um modelo. O modelo pode ser formado pelos metacaracteres[7]: *, ?, #, !, [0-9], [A-Z], etc.

Quadro 3.1 – Operadores para comparação de seqüências de caracteres

[7] Metacaracteres são caracteres que têm significados especiais (não literais), geralmente utilizados em buscas e expressões regulares.

3.1.1.3.3 Expressões Regulares no Visual Basic 6

Expressões regulares são padrões, que podem ser formados por caracteres literais ou metacaracteres, utilizados para manipulações complexas como buscar elementos do texto e realizar operações de substituição de caracteres ou cadeias de caracteres.

As expressões regulares surgiram em editores de texto do sistema operacional Unix, dando origem a aplicativos como *grep* e *sed*. A popularização das expressões regulares ocorreu com a criação do pacote gratuito chamado regex, que poderia ser incluído por qualquer programa. Desde então, as expressões regulares vêm sendo utilizadas por vários programas e linguagens de programação.

Para ter acesso a expressões regulares no Visual Basic, usamos o objeto RegExp. O objeto fornece as propriedades e funções necessárias para validação e comparação de um padrão definido.

As principais propriedades do objeto RegExp são:

1. *Pattern* – propriedade de valor tipo *String* que define a expressão regular.
2. *IgnoreCase* – propriedade de valor tipo *Bollean*[8] que indica se é necessário testar a expressão contra todas as possíveis comparações na string
3. *Global* – propriedade de valor tipo *Bollean* que indica se o padrão deve coincidir com todas as ocorrências no texto como um todo ou se o padrão deve coincidir apenas com a primeira ocorrência.

A definição da propriedade *Pattern*, utilizada para coincidir com um determinado padrão, pode ser feita a partir de uma seqüência de caracteres literais que se deseja buscar (expressão mais simples), por exemplo, buscar a palavra ‘casa’ em um texto, ou composta por metacaracteres e classes de metacaracteres (expressão mais complexa) para tornar a busca mais específica, por exemplo, fazer concordâncias da palavra casa com a expressão: "(.{40})\bcasa\b(.{40})". No exemplo dado, a expressão regular retornará todas as ocorrências da palavra ladeada por 40 caracteres da direita e da esquerda. Veja a seguir os principais

[8] Os dados do tipo *Bollean* podem ser apenas *True* ou *False*.

metacaracteres.

\	Usado para encontrar um metacaractere (ponto, interrogação, asterisco, etc.)
.	Qualquer caractere, exceto quebra de linha.
^x	Qualquer caractere, exceto x[9].
[x]	Qualquer instância de x dentro de uma seqüência entre colchetes – [abxyz] – [0-9].
|	Funciona como um operador: ou.
( )	Utilizado para agrupar seqüências de caracteres ou padrões.
{ }	Utilizado para definir quantificadores numéricos.
{x}	O caractere ou seqüência declarada à esquerda deve ocorrer exatamente x vezes.
{x,}	O caractere ou seqüência declarada à esquerda deve ocorrer pelo menos x vezes.
{x,y}	O caractere ou seqüência declarada à esquerda deve ocorrer pelo menos x vezes, mas não mais que y[10] vezes.
?	O caractere ou seqüência declarada antes é opcional ou somente uma.
*	Utilizado para encontrar 0 ou mais ocorrências de um caractere ou seqüência.
+	Utilizado para encontrar 1 ou mais ocorrências de um caractere ou seqüência de caracteres que precede.
^	Utilizado para coincidir com o início de uma linha.
$	Utilizado para coincidir com o final de uma linha.

Quadro 2.2 – Principais metacaracteres

Classes de metacaracteres:

\d	Qualquer dígito, o mesmo que [0-9].
\D	Qualquer caractere, exceto dígitos, o mesmo que [^0-9].
\s	Caracteres em branco (espaços, tabulações, quebras de linhas, etc).
\S	Qualquer caractere, exceto caracteres em branco.
\w	Qualquer caractere formador de palavras.
\W	Qualquer caractere não-formador de palavras.
\b	Caracteres delimitadores de palavras.
\B	Caracteres não-delimitadores de palavras.

Quadro 3.3 – Classes de metacaracteres

[9] Utilizado aqui para representar qualquer caractere para exemplificar o uso de algum metacaractere.
[10] Utilizado aqui para representar qualquer caractere para exemplificar o uso de algum metacaractere.

Antes de utilizar o objeto, é preciso definir os valores de suas propriedades, descritas acima, que são passadas para a instância do *RegExp*.

Após a definição das propriedades, é possível utilizar os seguintes métodos para determinar se um conjunto de caracteres coincide com um padrão determinado da expressão:

1. *Test* - Retorna um valor do tipo *Bollean* que indica se a expressão pode ser coincidida com sucesso com uma seqüência de caracteres.
2. *Replace* – Substitui todas as ocorrências de um determinado padrão por um valor.
3. *Execute* - Retorna um objeto *MatchColletcion* que contém um objeto *Match*[11] para cada coincidência obtida com sucesso.

3.1.2 Editor de ícones

Além do *Visual Basic*, foi utilizada uma versão *trial* do programa *ArtIcons* da *Aha-soft*, disponível no *site* www.aha-soft.com, que possibilita a edição e criação de ícones.

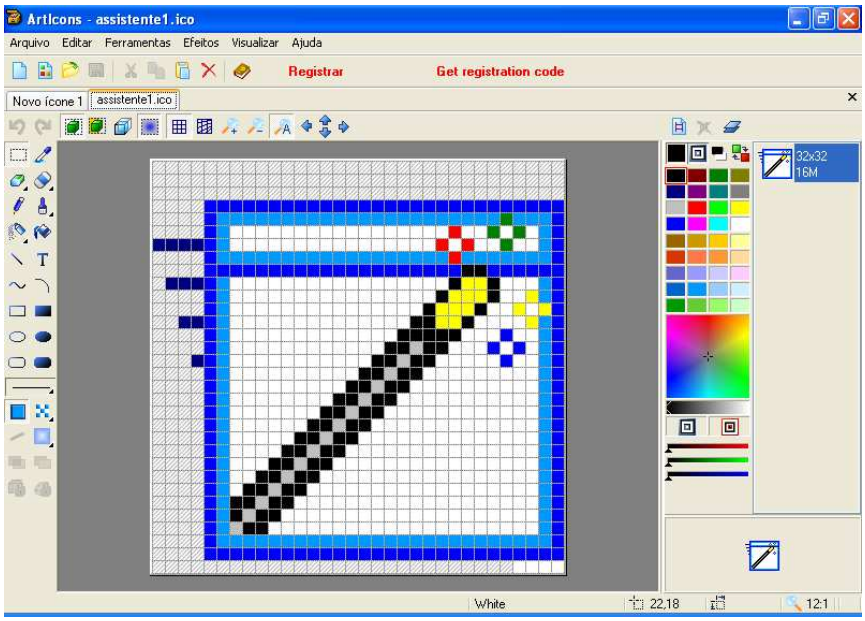


Figura 3.4 – Área de trabalho do editor de ícones ArtIcons

[11] O objeto Match é utilizado para armazenar os resultados da busca.

3.1.3 Molebox

O *Molebox* é uma ferramenta que faz a compilação de programas desenvolvidos em Windows. Ele compila o arquivo executável (.exe) com os arquivos .dll, .ocx e de dados (bancos de dados, figuras, etc) em um único arquivo executável.

Esse tipo de programa é muito utilizado para proteger a aplicação de eventuais modificações ou extração de arquivos de mídia, como figuras e sons (em games), ou dados que não devam ser distribuídos fora da aplicação.

O programa também é útil para garantir o funcionamento de *software* desenvolvidos em *Visual Basic*. O programa compila todos os arquivos necessários para o funcionamento do *software* para que os arquivos não precisem ser registrados no computador do usuário. Se as versões dos arquivos do *software* são diferentes das versões do computador do usuário, o *software* não funciona.

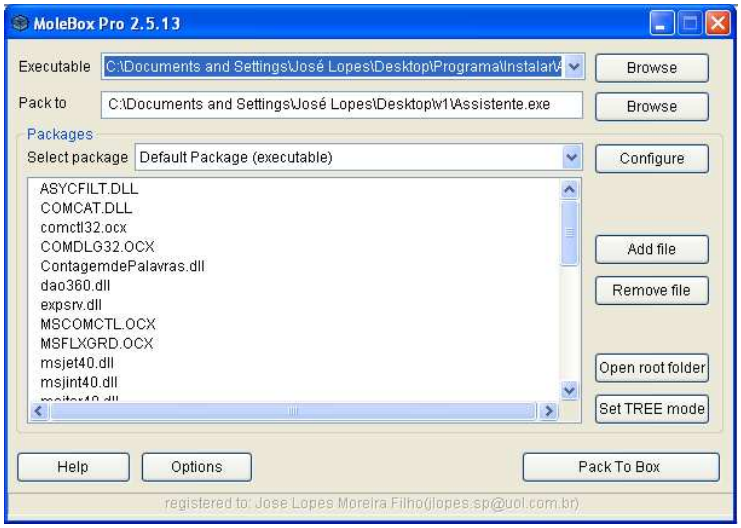


Figura 3.5 – Molebox Pro 2.5.13

O programa foi utilizado para compilar em pacotes todos os arquivos de sistema (.dll, .ocx, .mdb e outros) necessários para o funcionamento do *software*, de modo que o programa não precise registrar qualquer arquivo no computador do usuário e, portanto, garantindo seu funcionamento efetivo.

3.1.4 Programa para criação de arquivos de instalação

Utilizamos o *software* livre *Inno Setup Compiler*, disponível na Internet, para criar um arquivo de instalação.

O *Inno Setup Compiler* é um compilador que reúne todos os arquivos do *software* em um arquivo de instalação e permite que o usuário crie um assistente de instalação.

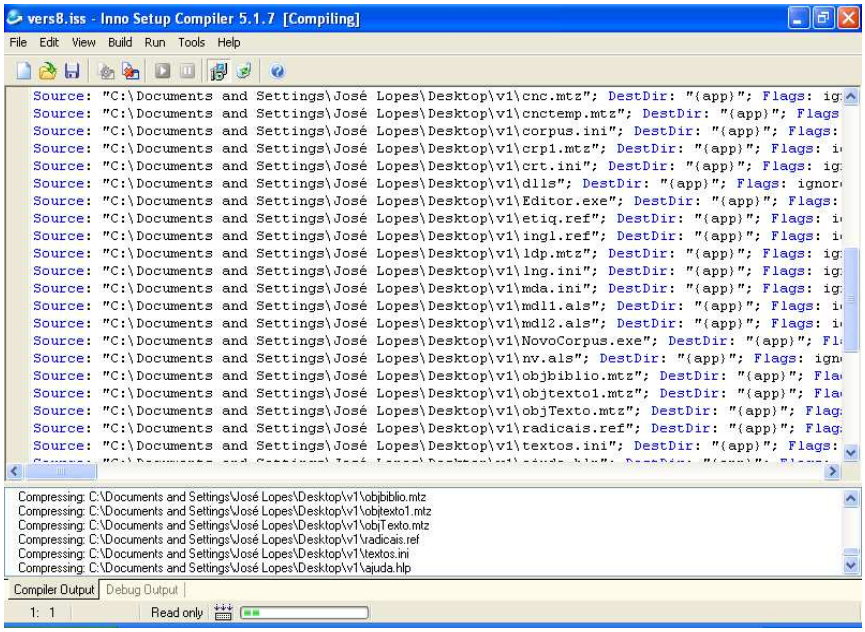


Figura 3.6 – Compilador *Inno Setup Compiler*

Tendo exposto a linguagem de programação e as principais ferramentas utilizadas na pesquisa, passa-se à descrição dos procedimentos de desenvolvimento do *software*.

3.5 Procedimentos de desenvolvimento do *software*

A metodologia de desenvolvimento e codificação do *software* ocorreu através dos procedimentos expostos nas subseções que se seguem. A análise e apresentação dos resultados das principais funções desempenhadas pelo *software* são apresentadas no próximo capítulo.

O desenvolvimento da ferramenta foi executado de acordo com os objetivos de pesquisa, ou seja, propor o desenvolvimento de um *software* que utilize a instrumentação da Lingüística de

Corpus, apresentar o *software* desenvolvido e avaliá-lo com base nos princípios utilizados na metodologia proposta.

Assim, descrevemos, na primeira subseção, o levantamento das características do *software*. Na segunda subseção, são descritos os procedimentos para o desenvolvimento do *software* por meio de programação.

3.5.1 Descrição das principais características do *software*

Antes de iniciar a programação (codificação), foi necessário fazer a descrição de todos os componentes necessários para codificar o sistema e sua funcionalidade.

Aqui, são explicitados apenas alguns exemplos, isto é, uma base para a programação do sistema, pois algumas funcionalidades e problemas foram surgindo durante a codificação.

O *software* deve conter, além dos componentes básicos comuns a todos os *software*, como ajuda e arquivos de instalação, os seguintes componentes principais especificados no quadro abaixo:

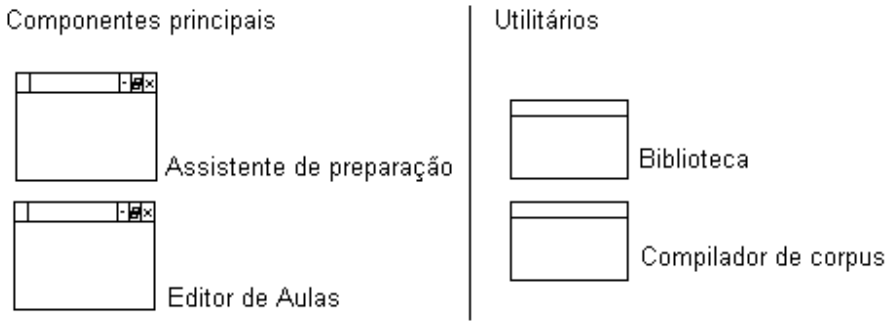


Figura 3.7 – Principais componentes do *software.*

1. Assistente de preparação – Funcionará como um *wizard,* que faz perguntas e oferece opções acerca do conteúdo a ser incluídos no material.

2. Editor de aulas – Funcionará como um editor de textos normal, mas com algumas funções específicas para a preparação de aulas.

3. Biblioteca – Será útil para armazenar textos que sejam considerados interessantes pelo usuário para a preparação de atividades.

4. Compilador de corpus – Adicionar textos (compilação de corpus) e fazer consultas.

O componente principal do *software* é o Assistente de Preparação de Aulas. A seguir, na figura 3.8, exemplificamos o seu funcionamento de forma detalhada por meio de esquemas. As letras M1, M2 e M3 representam os modelos de atividades; I representa as informações extraídas do texto para completar o modelo de atividade; M representa o modelo de atividade escolhido pelo usuário; A representa a atividade pronta.

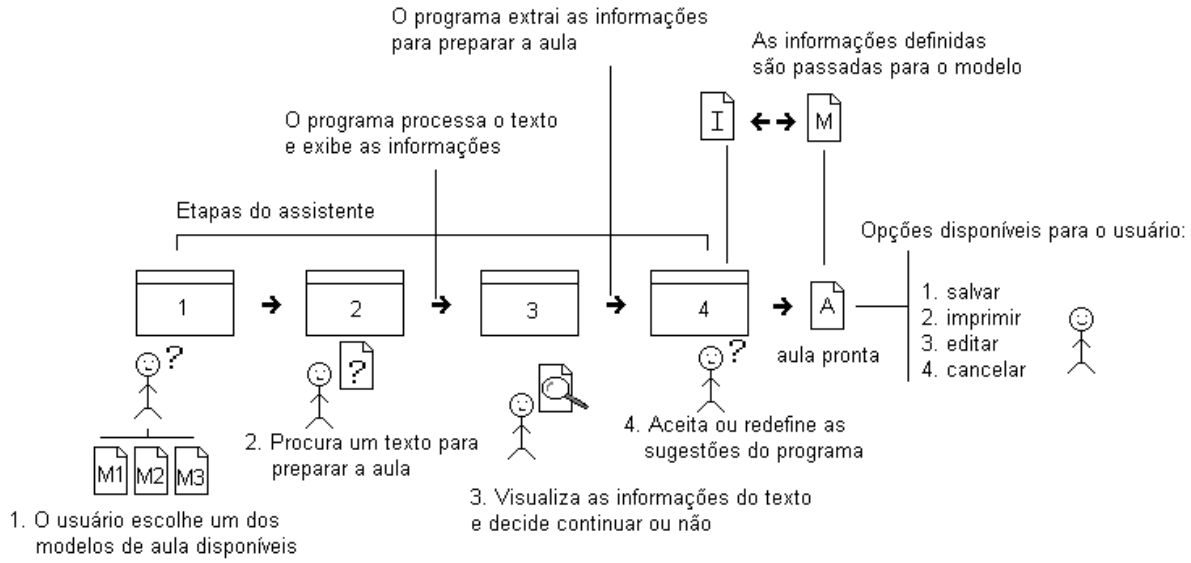


Figura 3.8 – Descrição do funcionamento do Assistente de Preparação de Aulas

Tendo especificado os componentes e suas funcionalidades o próximo passo foi a codificação do projeto.

3.5.2 Desenvolvimento do *software* por meio de programação

Nesta seção, descrevemos os procedimentos de programação para as principais análises lingüísticas feitas pelo *software*, as quais estão relacionadas às perguntas de pesquisa.

3.5.2.1 Fazer lista de palavras

Para fazer listas de palavras foi necessário definir o que é uma palavra, em termos de linguagem de computador. O que consideramos como um texto ou palavra, para o computador, é apenas uma seqüência de caracteres. O computador não consegue distinguir o que é uma palavra, frase ou parágrafo. Temos que ensiná-lo a identificar o que é uma palavra, frase, parágrafo ou outra unidade lingüística.

Para isso, tivemos que criar uma função para fazer um pré-processamento do texto. Este pré-processamento pode ser chamado de itemização. Conforme Berber Sardinha (2004:128), a itemização consiste na separação das unidades ortográficas, normalmente por meio da inserção de espaços em branco ou quebras de linhas entre elas.

Antes da itemização, criamos uma função para abrir o arquivo do texto e passá-lo para uma variável. A variável é útil para armazenar o texto para manipulações posteriores.

O código correspondente para essa função segue abaixo:

```
Public Function Abrir_Texto()
Open strCaminho For Input As #1
strTexto = Input(LOF(1), #1)
Close #1
End Function
```

Código 3.1 – Função para extrair um texto de um arquivo

A função de itemização que criamos utiliza expressões regulares para apagar toda a pontuação do texto, deixando apenas as palavras delimitadas por espaços em branco.

```
Public Function Limpar_texto(Texto As String) As String
Dim r As New RegExp

r.Pattern = "\||\\|<|,|>|\.|\.\.\.|:|;|\?|/|\^|~|\}|\]|\{|\[|\+|=|_|\)" & _
            "|\*|\&|¨|%|\$|#|@|\(|\!|[0-9]|´|`|\s+|" & Chr(34) & _
            "|" & vbCr & "|\s+|--+|''+|''|\W-+|\b-\b|- " & vbNewLine & _
            "|' " & vbNewLine
r.Global = True
r.IgnoreCase = True
Texto = Replace(Replace(Texto, "...", " "), Chr(34), " ")
Texto = UCase(r.Replace(Texto, " " & vbNewLine))
Limpar_texto = Trim(r.Replace(Texto, " "))

End Function
```

Código 3.2 – Função para fazer a itemização do texto

Após a itemização do texto, criamos uma função para identificar as palavras do texto e passar para uma outra função que faz a contagem das palavras. Na função criada, a identificação das palavras é feita com base em um caractere delimitador, no nosso caso, o espaço em branco.

A função que identifica as palavras no texto itemizado é:

```
Public Function Contar_Palavras&(Expression$, D As Dictionary, _
                                 Optional IncludeEmpty As Boolean)
Dim i As Variant
Dim j As Long
Const Delimiters = " "
Const ARR_CHUNK& = 1024
    Dim cExp&, ubExpr&
    Dim cDel&, ubDelim&
    Dim aExpr%(), aDelim%()
    Dim sa1 As SAFEARRAY1D, sa2 As SAFEARRAY1D
    Dim cTokens&, iPos&
    ubExpr = Len(Expression$)
    ubDelim = Len(Delimiters)
    sa1.cbElements = 2:     sa1.cElements = ubExpr
    sa1.cDims = 1:          sa1.pvData = StrPtr(Expression$)
    RtlMoveMemory ByVal VarPtrArray(aExpr), VarPtr(sa1), 4
    sa2.cbElements = 2:     sa2.cElements = ubDelim
    sa2.cDims = 1:          sa2.pvData = StrPtr(Delimiters)
    RtlMoveMemory ByVal VarPtrArray(aDelim), VarPtr(sa2), 4
    ubDelim = ubDelim - 1
    For cExp = 0 To ubExpr - 1
        For cDel = 0 To ubDelim
            If aExpr(cExp) = aDelim(cDel) Then
                If cExp > iPos Then
                  Contar_ocorrencias Mid$(Expression$, iPos + 1, cExp - iPos), D
                    cTokens = cTokens + 1
                ElseIf IncludeEmpty Then
                    Contar ocorrencias vbNullString, D
                     cTokens = cTokens + 1
                 End If
                 Contar_Palavras = cTokens - 1
                 RtlZeroMemory ByVal VarPtrArray(aExpr), 4
                 RtlZeroMemory ByVal VarPtrArray(aDelim), 4
             DoEvents
             End Function
```

Código 3.3 – Função para identificar as palavras do texto

A função recebe as palavras e as armazena em uma lista. Quando uma nova palavra é adicionada à lista, a função testa se já existe alguma palavra igual na lista. Se a palavra é encontrada na lista, o programa apenas consulta quantas vezes ela já ocorreu e registra mais uma ocorrência da palavra. Caso seja a primeira ocorrência da palavra na lista, a palavra é adicionada à lista e o valor ‘1’ é adicionado, indicando a freqüência única.

A função correspondente é mostrada na figura abaixo:

```
Private Sub Contar_ocorrencias(strPalavra As String, dicLista As Dictionary)
DoEvents
If dicLista.Exists(strPalavra$) = True Then
DoEvents
dicLista.Item(strPalavra$) = dicLista.Item(strPalavra$) + 1
y = y + 1
Else
DoEvents
dicLista.Add strPalavra$, 1
y = y + 1
End If
DoEvents
End Sub
```

Código 3.4 – Rotina para contar a ocorrência das palavras

Após a contagem, as palavras são armazenadas em um arquivo de banco de dados para serem utilizadas pelo programa em outras tarefas.

3.5.2.2 Fazer concordâncias

Para fazer concordâncias, utilizamos uma expressão regular que busca todas as ocorrências de uma determinada palavra em cada um dos textos de um corpus e as traz ladeada por 40 caracteres da esquerda e da direita da palavra.

```
r.Pattern = "(.{40})\b" & strP1 & "\b(.{40})"
r.Global = True
r.IgnoreCase = True
```

Código 3.5 – Expressão regular para fazer concordâncias

Na utilização do Assistente de Preparação de Aulas, quando o usuário escolhe um corpus para extrair exemplos de uso da língua, os caminhos dos arquivos dos textos são passados para uma variável. Uma função abre um arquivo por vez e faz a busca utilizando a expressão regular apresentada acima. A função extrai as linhas de concordância e as armazena em uma variável.

O código completo pode ser visualizado abaixo:

```
Public Function Concordancias1()
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim t As Long
Dim i As Integer
Dim j As Long
Dim n As String
Dim Mem As New clsM
Dim l As Long
r.Pattern = "(.{40})\b" & strP1 & "\b(.{40})"
r.Global = True
r.IgnoreCase = True
For j = 1 To UBound(Biblio)
l = l + 1
If l = 100 Then GoTo Sair
Open Biblio(j) For Input As #1
t = Mem.malloc(Input(LOF(1), #1))
Close #1
Set c = r.Execute(Replace(Mem.RetMemory(t), vbNewLine, ""))
For Each M In c
l = 0
i = i + 1
n = CStr(i)
If Len(CStr(i)) = 1 Then n = " " & CStr(i)
Conc1$ = Conc1$ & n & ". " & M.Value & vbNewLine
If i = nEx7 Then GoTo Sair
Next
Mem.FreeMemory (t)
Next j
DoEvents

Sair:
Set r = Nothing
Set M = Nothing
Set c = Nothing
Set Mem = Nothing
End Function
```

Código 3.6 – Função para extrair concordâncias no Assistente de Preparação de Aulas

3.5.2.3 Identificar as palavras-chave de um texto

A identificação das palavras-chave de um texto é feita a partir da lista de freqüência das palavras armazenada em um arquivo de banco de dados. O programa define as palavras-chave por meio de uma comparação estatística entre as palavras da lista do texto e as palavras da lista do corpus de referência (BNC). A fórmula estatística para calcular a chavicidade das palavras é *Log-likelihood*[12].

[12] É uma estatística de associação entre itens lexicais.

A função criada para calcular a chavicidade das palavras é:

```
Public Function Calcular_chavicidade(fpc1 As String, fpc2 As Long, _
ntpc1 As Long, ntpc2 As Long, Significancia As Long) As String


Dim E1, E2 As Double
Dim LL As Double


E1 = ntpc1 * (fpc1 + fpc2) / (ntpc1 + ntpc2)
E2 = ntpc2 * (fpc1 + fpc2) / (ntpc1 + ntpc2)
LL = 2 * ((fpc1 * Log(fpc1 / E1)) + (fpc2 * Log(fpc2 / E2)))

If LL <= Significancia Then
Calcular_chavicidade = Format(LL, "00.00") & "-"
Else
Calcular_chavicidade = Format(LL, "00.00") & "+"
End If

End Function
```

Código 3.7 – Função para calcular a chavicidade das palavras

A função apresentada é utilizada para cada uma das palavras da lista de freqüência. As variáveis que recebem os valores necessários para os cálculos são:

1. fpc1: freqüência da palavra no corpus de estudo.
2. fpc2: freqüência da palavra encontrada na lista do corpus de referência (*BNC*).
3. ntpc1: número total de palavras do corpus de estudo (número de *tokens*).
4. ntpc1: número total de palavras do corpus de referência (número de *tokens*)
5. Significância: valor de corte para o resultado da variável LL (valor de Log-likelyhood).

Após o cálculo da fórmula, o valor de chavicidade é armazenado na variável LL. Em seguida, a variável é testada. Se o valor de LL for menor ou igual ao valor de Significancia, é acrescentado um sinal de menos ‘-‘, para indicar que o valor de chavicidade da palavra é menor que o valor de corte. Caso o valor de LL seja maior que o valor de Significância, é acrescentado ao valor de chavicidade o sinal ‘+’, para indicar que o valor de chavicidade é maior que o valor de corte. Esta marcação é utilizada para indicar ao programa quais são realmente as palavras-chave.

O código apresentado abaixo extrai as palavras do texto armazenadas em banco de dados e as procura na lista de palavras do corpus de referência (BNC). Se o programa encontrar a palavras na lista do corpus de referência, os valores necessários para calcular a chavicidade são passados para a função 'Calcular_chavicidade'. Após o cálculo, as palavras são armazenadas em um banco de dados.

```
Public Function Fazer_palavras_chave()
Dim bd1 As Database
Dim bd2 As Database
Dim tb1 As Recordset
Dim tb2 As Recordset
Dim p1 As Long
Dim M As New clsM
Dim l As Long
Dim i As Long
Set bd1 = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb1 = bd1.OpenRecordset("SELECT * FROM [formas] WHERE [palavra] > '1'" _
                            & "ORDER BY [palavra] ASC")
DoEvents
tb1.MoveFirst
Do Until tb1.EOF
DoEvents
p1 = M.malloc(Replace(tb1("palavra"), "'", "´"))
Set bd2 = OpenDatabase(App.Path & "\ing1.ref")
Set tb2 = bd2.OpenRecordset("SELECT [palavra],[frequencia] FROM [eng]" _
                    & "WHERE [palavra] =" & "'" & M.RetMemory(p1) & "'" & " ")
M.FreeMemory (p1)
If tb2.RecordCount = 0 Then
tb1.Edit
tb1("chavicidade") = 0
tb1.Update
Else
tb1.Edit
tb1("chavicidade") = Calcular_chavicidade(tb1("frequencia"), tb2("frequencia"), _
lngPalavras, 102467488, 0.1)
tb1.Update
End If
tb1.MoveNext
Loop
tb1.Close
tb2.Close
bd1.Close
bd2.Close
Set tb1 = Nothing
Set tb2 = Nothing
Set bd1 = Nothing
Set bd2 = Nothing
Set M = Nothing

DoEvents
End Function
```

Código 3.8 – Função para fazer palavras-chave

As palavras que não são encontradas no corpus de referência são assinaladas com o valor

‘0’.

3.5.2.4 Identificar as palavras cognatas de um texto

Para identificar as palavras cognatas no texto, foi feita uma análise manual da lista de palavras do BNC. A análise consistiu em identificar palavras com ortografia parecida com palavras do português que tivessem o mesmo significado em ambas as línguas. Identificamos as primeiras 2.357 possíveis palavras cognatas mais freqüentes da lista do BNC para formar um banco de dados de referência.

O banco de dados formado serve de base para comparações com a lista de palavras de um texto escolhido na preparação de atividades com o Assistente de Preparação de Aulas. A comparação é feita apenas com os primeiros cinco caracteres da lista de palavras do texto e da lista do banco de dados de referência. O resultado da comparação é binário (0/1). Se os cinco caracteres da palavra do texto forem iguais aos cinco primeiros caracteres do banco de dados, o programa insere o valor 1, indicando que a palavra é cognata. Caso sejam diferentes, o programa insere o valor 0, indicando que a palavra não é cognata.

Embora o método utilizado para a identificação de palavras cognatas possa ser considerado limitado, por depender do banco de dados de referência extraído de uma pequena análise manual, foi a melhor maneira encontrada para tipo de aplicação desenvolvida. Uma outra opção seria utilizar a lista de palavras do corpus Banco de Português (BP) como banco de dados de referência, porém, a lista de palavras do BP contém também palavras em inglês, o que poderia influenciar negativamente no resultado da comparação.

3.5.2.5 Etiquetar as palavras de um texto

A etiquetagem morfossintática das palavras é feita a partir da lista de palavras etiquetadas. do BNC. Uma função extrai as palavras do texto armazenadas no banco de dados e procura na lista de palavras etiquetadas do corpus de referência (BNC). Se a palavra é encontrada, a função atribui a etiqueta da palavra do corpus de referência à palavra do texto. Se a palavra do texto não é encontrada na lista etiquetada do corpus de referência, a função atribui uma etiqueta de substantivo (nn1).

O código utilizado para etiquetar as palavras é:

```
Public Function Etiquetar()
Dim bd1 As DAO.Database
Dim tb1 As DAO.Recordset
Dim bd2 As DAO.Database
Dim tb2 As DAO.Recordset
Dim M As New clsM
Dim p1 As Long
Set bd1 = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb1 = bd1.OpenRecordset("formas", dbOpenTable)
Set bd2 = DBEngine.OpenDatabase(App.Path & "\etiq.ref")
DoEvents
tb1.MoveFirst
Do Until tb1.EOF
DoEvents
p1 = M.malloc(Replace(tb1("palavra"), "'", "´"))
Set tb2 = bd2.OpenRecordset("SELECT [tag] FROM [etiquetadas]" _
& "Where [palavra] = " & " '" & M.RetMemory(p1) & "'" & " ")
M.FreeMemory (p1)
If tb2.RecordCount <> 0 Then
tb1.Edit
tb1("tag") = tb2("tag")
tb1.Update
Else
tb1.Edit
tb1("tag") = "nn1"
tb1.Update
End If
tb1.MoveNext
Loop
tb1.Close
bd1.Close
tb2.Close
bd2.Close
Set tb1 = Nothing
Set bd1 = Nothing
Set tb2 = Nothing
Set bd2 = Nothing
Set M = Nothing
DoEvents
End Function
```

Código 3.9 – Função para etiquetar as palavras do texto

3.5.2.6 Calcular a densidade lexical do texto

Para calcular a densidade lexical de um texto, utilizamos a seguinte função:

```
Public Function Densidade_lexical()
Dim bd1 As DAO.Database
Dim tb1 As DAO.Recordset
Dim palavra As String
Dim r As New RegExp
r.Pattern = "AJO|AJC|AJS|NNO|NN1|NN2|NPO|VBB|VBD|VBG|VBI|VBN|VBZ|VDB|" _
            & "VVG|VVD|VVI|VVB|VV"
r.Global = True
r.IgnoreCase = True
Set bd1 = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb1 = bd1.OpenRecordset("formas", dbOpenTable)
DoEvents
tb1.MoveFirst
Do Until tb1.EOF
DoEvents
If r.Test(tb1("tag")) = True Then
lngNlexicais = lngNlexicais + tb1("frequencia")
tb1.Edit
tb1("categoria") = "1"
tb1.Update
Else
lngNgramaticais = lngNgramaticais + tb1("frequencia")
tb1.Edit
tb1("categoria") = "0"
tb1.Update
End If
tb1.MoveNext
Loop
dblDensidadelexical = Format((100 * (lngNlexicais / lngPalavras)), "00.00")
tb1.Close
bd1.Close
Set tb1 = Nothing
Set bd1 = Nothing
DoEvents
End Function
```

Código 3.10 – Função para calcular a densidade lexical do texto

O cálculo de densidade lexical é feito a partir da lista de palavras do texto já etiquetada. Primeiro, a função procura palavras que possuem etiquetas consideradas lexicais (de adjetivos, verbos e substantivos) e as marca com o valor “1”. As palavras que não possuem as etiquetas especificadas são marcadas com o valor “0”. Em seguida, a função faz a contagem das palavras que tiveram o valor “1”, palavras lexicais. A contagem leva em consideração quantas vezes a palavra ocorreu no texto[13]. Por fim, o valor da densidade lexical é calculado dividindo-se o

[13] Não confundir com o número de formas (*types*).

número de palavras lexicais pelo número de palavras do texto (*tokens*). O valor resultante é multiplicado por 100 para obter a porcentagem.

3.5.2.7 Definir a dificuldade estrutural do texto

Para definir a dificuldade estrutural do texto, utilizamos uma adaptação da fórmula Flesch, utilizada para estimar o grau de dificuldade de um texto. Existem várias fórmulas para o cálculo da dificuldade (legibilidade) de um texto. Escolhemos a fórmula de Flesch por ser a mais conhecida e por produzir um resultado numérico que pode ser comparado com uma escala de níveis de dificuldade.

A função que representa a fórmula de Flesch é:

```
Public Function Flesh_formula(nfrases As Long, nsilabas As Long, nItens As Long)
Dim x As Long
Dim y As Long
Dim t As Long

If nfrases = 0 Then nfrases = 1

x = (1.015 * (nItens / nfrases))
y = (0.846 * (nsilabas / nItens))
t = (x + y) + 100
Flesh_formula = 206.835 - t

End Function
```

Código 3.11 – Função para calcular a dificuldade estrutural do texto

A fórmula leva em consideração o número de sílabas, o número de frases e o número de palavras do texto. A fórmula é representa por Fulcher (1997:500) da seguinte forma:

Índice = 206.835

- (0.846 x média do número de sílabas)

- (1.015 x média do número de palavras por frase)

Uma das dificuldades encontradas para implementar a fórmula foi a impossibilidade de fazer a contagem automática do número de sílabas das palavras do texto. Para superar essa limitação, utilizamos uma expressão regular que, na verdade, conta o número de vogais (aeiou)

no texto, pois o programa não lida com sons, mas com caracteres. Quando há duas ou mais vogais juntas, o a expressão regular conta apenas uma ocorrência. A expressão também considera as letras ‘y’ e ‘w’ na contagem.

O recurso utilizado é uma forma de fazer a estimativa do número de sílabas. Reconhecemos a limitação, porém acreditamos que possa ser utilizado, pelo menos, como um índice.

O código para fazer a estimativa do número de sílabas é apresentado abaixo:

```
Public Function Contar_silabas(Texto As String) As Long
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim f As Long
r.IgnoreCase = True
r.Global = True

r.Pattern = " a | e | i | o | u | y " & _
            "|qua|que|qui|quo" & "|gua|gue|gui|guo" & _
            "|b[aeiouAEIOUyY]+" & "|c[aeiouAEIOUyY]+" & _
            "|d[aeiouAEIOUyY]+" & "|f[aeiouAEIOUyY]+" & _
            "|g[aeiouAEIOUyY]+" & "|h[aeiouAEIOUyY]+" & _
            "|j[aeiouAEIOUyY]+" & "|k[aeiouAEIOUyY]+" & _
            "|l[aeiouAEIOUyY]+" & "|m[aeiouAEIOUyY]+" & _
            "|n[aeiouAEIOUyY]+" & "|p[aeiouAEIOUyY]+" & _
            "|q[aeiouAEIOUyY]+" & "|r[aeiouAEIOUyY]+" & _
            "|s[aeiouAEIOUyY]+" & "|t[aeiouAEIOUyY]+" & _
            "|v[aeiouAEIOUyY]+" & "|x[aeiouAEIOUyY]+" & _
            "|w[aeiouAEIOUyY]+" & "|y[aeiouAEIOUyY]+" & _
            "|z[aeiouAEIOUyY]+" & "|\ba\w|\be\w|\bi\w|\bo\w|" _
            & "\bu\w|\wa\b|\we\b|\wi\b|\wo\b|\wu\b" & "|\ba\w+|\be\w+|" _
            & "\bi\w+|\bo\w+|\bu\w+"

Set c = r.Execute(Texto)
For Each M In c
f = f + 1
Next
Contar_silabas = f
End Function
```

Código 3.12 – Função para contar o número de sílabas

A contagem do número de frases do texto é feita por uma expressão regular que busca no texto marcas de pontuação seguidas de espaço ou quebra de linha. A função correspondente é apresentada no código 3.13.

```
Public Function Contar_frases(Texto As String) As Long
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim f As Long

r.Global = True

r.Pattern = "[\.;\?!]\b|[\.;\?!]" & vbNewLine
Set c = r.Execute(Texto)
For Each M In c

f = f + 1

Next
Contar_frases = f
End Function
```

Código 3.13 – Função para contar o número de frases

As informações são passadas para a função flesh_formula, que retorna um índice numérico de 0 a 100. O valor é testado segundo a seguinte escala:

Valor	**Nível de dificuldade**
90 a 100	muito fácil
80 a 79	razoavelmente fácil
60 a 69	padrão
50 a 59	razoavelmente difícil
30 a 49	difícil
0 a 29	muito difícil

Quadro 3.4 – Escala de níveis de dificuldade

Neste capítulo, apresentamos os procedimentos de desenvolvimento do *software* por meio de programação. No próximo capítulo, apresentamos o *software* construído.

Capítulo 4

Apresentação do *software* construído

Neste capítulo, apresentamos o *software* construído a partir da metodologia especificada e procedimentos de desenvolvimento em capítulos anteriores. Para tanto, fazemos uma descrição das principais ferramentas e funções do *software*, assim como uma demonstração passo a passo de seu funcionamento, na tentativa de simular sua utilização por parte do público-alvo.

4.1 Descrição dos componentes do *software*

O *software* construído é composto por dois componentes principais:

1. Assistente de Preparação de Aulas.
2. Editor de Aulas.

Esses componentes são responsáveis pelas principais funcionalidades do *software*. O primeiro componente foi criado especificamente para o usuário poder preparar rapidamente um material didático por meio de etapas. O segundo componente pode ser considerado como uma extensão do primeiro, visto que podemos utilizá-lo para editar o material preparado com o primeiro componente. Contudo, ambos os componentes são independentes, podendo ser acessados independentemente.

Além dos componentes mencionados acima, o *software* possui ferramentas que também executam funções importantes, e podem, de certa forma, serem consideradas como utilitários do *software*, por executar funções auxiliares e ampliar as possibilidades de seu uso. Tais ferramentas são:

1. Biblioteca.

2. Novo Corpus.

Os dois utilitários podem ser acessados a partir do menu Arquivo do Editor de Aulas. Com o utilitário Biblioteca, o usuário pode armazenar textos que achar interessante para

preparação de materiais didáticos com o *software*. O utilitário Novo Corpus é utilizado para a seleção e armazenamento de coletâneas de textos que formam um corpus, do qual é possível extrair exemplos de uso da língua para a elaboração de exercícios com linhas de concordância.

A seguir, fazemos uma descrição mais detalhada de cada ferramenta e suas principais opções para elaboração de um material didático com corpora.

4.1.1 Assistente de Preparação de Aulas

O Assistente de Preparação de Aulas é a principal ferramenta do *software* e foco da presente pesquisa. Pode ser definido como um '*wizard'*, que faz perguntas e oferece opções acerca do conteúdo a ser incluído no material.

Assim, não é preciso grandes habilidades em informática para utilizá-lo. O processo de preparação da atividade é semi-automático, ou seja, grande parte da elaboração do material é feita automaticamente pelo *software*. O usuário é guiado através de quatro etapas até a preparação do material. Ao final, a ferramenta apresenta a atividade pronta e opções para que o usuário possa imprimir, salvar ou editar a atividade. O usuário decide aceitar a atividade preparada semi-automaticamente pelo *software* ou alterá-la para melhor atingir seus objetivos.

Para acessar o Assistente de Preparação de Aulas, o usuário pode clicar duas vezes no ícone de um atalho localizado na área de trabalho ou a partir do grupo Preparador de Aulas, no menu Iniciar do Windows, ambos criados após a instalação do *software*.

Quando o usuário clica duas vezes no ícone, a primeira etapa do Assistente de Preparação de Aulas aparece. Veja a figura 4.1 a seguir.

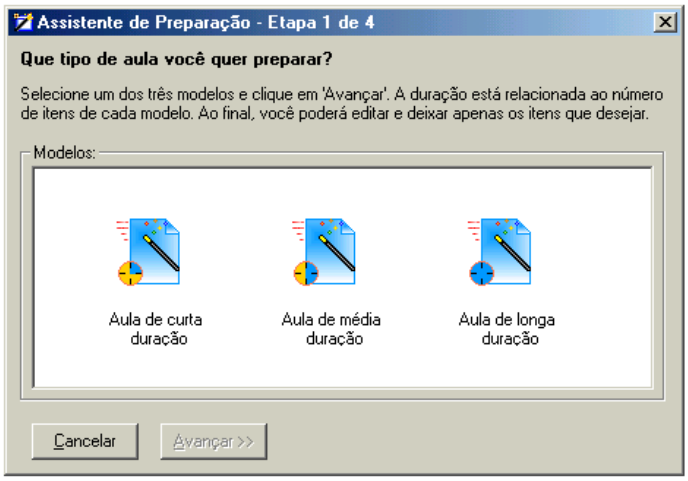


Figura 4.1 – Janela inicial do Assistente de Preparação de aulas

Para a preparação de uma atividade no Assistente de Preparação, são necessários dois requisitos: um texto em formato .txt (texto sem formatação), de escolha do próprio usuário, e um corpus criado com a ferramenta Novo Corpus. Sem os dois requisitos, não há como fazer a preparação do material.

O *software* é acompanho por um corpus, formado por alguns textos retirados da Internet, com o objetivo de manter sua funcionalidade caso o usuário não queira criar um novo corpus para utilizar o *software*.

4.1.2 Editor de Aulas

O Editor de Aulas é um editor de texto simples onde é possível realizar as seguintes funções: abrir, editar e salvar aulas criadas pelo Assistente de Preparação de Aulas, adicionar textos a uma biblioteca, selecionar textos e fazer a compilação de um novo corpus. O Editor de Aulas foi criado para fornecer opções de edição para as aulas preparadas pelo Assistente de Preparação de Aulas, sem que o usuário tenha que abrir um outro *software*.

As atividades criadas e salvas no Editor de Aulas possuem a extensão .als, definida na criação para identificar os arquivos gerados pelo *software*. São arquivos do tipo *Rich Text Format*[1] *(.rtf)* e podem ser abertos e editados por outros editores de texto que suportem este tipo

[1] O RTF é um formato de arquivo de documento desenvolvido pela Microsoft para intercâmbio de documentos entre diversas plataformas.

de arquivo. Se o usuário desejar utilizar recursos mais avançados de edição, pode abrir os arquivos no *Microsoft Word* e editá-los.

Para acessar o Editor de Aulas, o usuário pode clicar no ícone localizado no grupo Preparador de Aulas criado no menu Iniciar do Windows ou, ao utilizar o Assistente de Preparação de Aulas, quando a ferramenta apresentar a atividade pronta, clicar no botão Editar da barra de ferramentas Opções.

Como podemos ver abaixo, o Editor de Aulas possui características comuns a qualquer editor de texto.

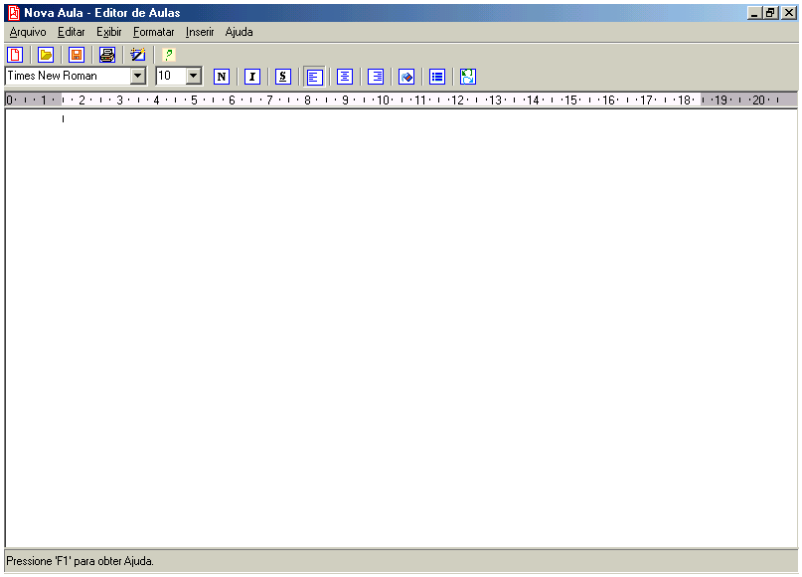


Figura 4.2 – Janela principal do Editor de Aulas

Além de possuir a aparência de um editor de texto como o *Microsoft Word*, também executa as principais funções de um editor de texto, como formatação, alinhamento, entre outras. Uma função que pode ser útil é a possibilidade de inserir figuras. Como já sabemos, para a preparação de aulas, podemos utilizar somente textos sem formatação e figuras. Se o usuário desejar utilizar um texto que contenha figuras, deverá manter a cópia original salva para posteriormente utilizar a opção Inserir figuras do menu Inserir.

4.1.3 Biblioteca

A Biblioteca pode ser acessada a partir do menu Arquivo do Editor de Aulas. A Biblioteca é uma ferramenta para armazenar textos que sejam considerados interessantes pelo usuário para a preparação de atividades. Depois de armazenados, os textos da Biblioteca ficam disponíveis para o Assistente de Preparação de Aulas, podendo ser acessados diretamente um uma caixa de listagem na segunda etapa da preparação da atividade.

A Biblioteca permite que o usuário visualize uma série de informações sobre os textos adicionados. Quando o usuário clica no texto adicionado, são exibidas informações como: número de palavras, número de formas, porcentagem de palavras cognatas, densidade lexical e dificuldade estrutural. Também é possível gerar listas de freqüência das palavras, lista de palavras-chave e saber quais são as possíveis palavras cognatas.

A figura abaixo mostra a ferramenta em funcionamento.

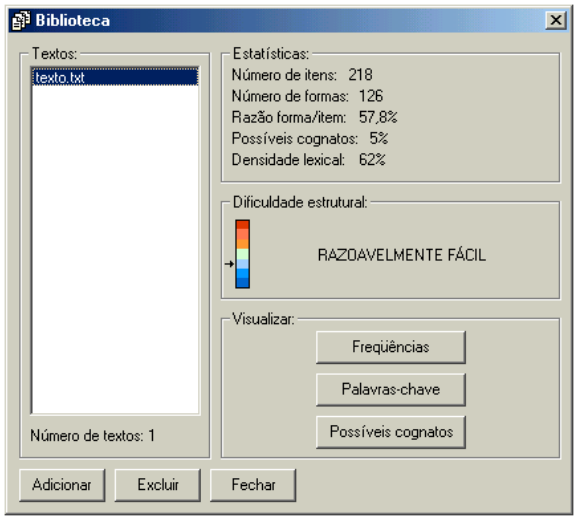


Figura 4.3 – Biblioteca

4.1.4 Novo Corpus

Esta ferramenta é utilizada para a compilação de um corpus. O usuário faz uma seleção de textos e os salva sob um nome escolhido por ele. O corpus formado é utilizado na preparação de aulas, servindo como fonte de exemplos para criação dos exercícios de concordância.

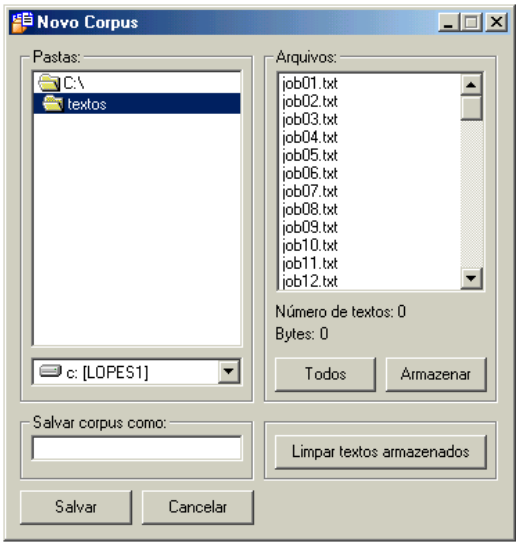


Figura 4.4 – Ferramenta para compilação de corpus

Para compilar um corpus, é necessário, primeiro, procurar os textos que farão parte do corpus. Em seguida, devemos selecioná-los e clicar no botão Armazenar. Depois, digitamos um nome em Salvar corpus como para identificá-lo e, finalmente, clicamos no botão Salvar. O processo é feito uma única vez. Os corpora salvos são armazenados em uma pasta e ficam disponíveis para o Assistente de Preparação para a extração de linhas de concordância.

4.2 Utilização do *software*

Nesta seção fazemos uma demonstração das principais funcionalidades do *software* construído. A demonstração é feita passo a passo para que fiquem claros os procedimentos que o usuário deverá executar para utilizar o *software* na preparação de atividades didáticas. O objetivo também é verificar a funcionalidade do *software*.

4.2.1 Instalar o *software*

Para que o *software* funcione corretamente, o computador deve possuir os seguintes requisitos básicos:

- Sistema operacional Windows 95/98/ME ou XP
- 128 MB de memória RAM
- 20 MB disponíveis para a instalação do *software*

Desenvolvido em Windows XP, o *software* foi testado com outras diferentes versões do Windows (98, 2000, ME) e apresentou funcionamento satisfatório em todas.

A performance do *software* varia de acordo com o nível de memória RAM do computador. O *software* foi testado em computadores com um nível mínimo de RAM de 128 e máximo de 512 MB. Um nível de memória RAM alto tende a favorecer a performance do *software*, aumentando a velocidade do processamento das informações.

Para instalar o *software* o usuário deverá clicar duas vezes no arquivo Instalar.exe. O arquivo possui 10.473 MB e contém todas as informações necessárias para instalação das ferramentas do *software* que nele estão compactadas. A instalação é realizada através de etapas, por meio de um assistente de instalação.

Figura 4.5 – Ícone do arquivo de instalação do *software*

A primeira tela a aparecer é a de apresentação, que recomenda o fechamento de todos os outros programas e janelas abertos para iniciar o processo de instalação. Para continuar, o usuário deve clicar no botão Avançar.

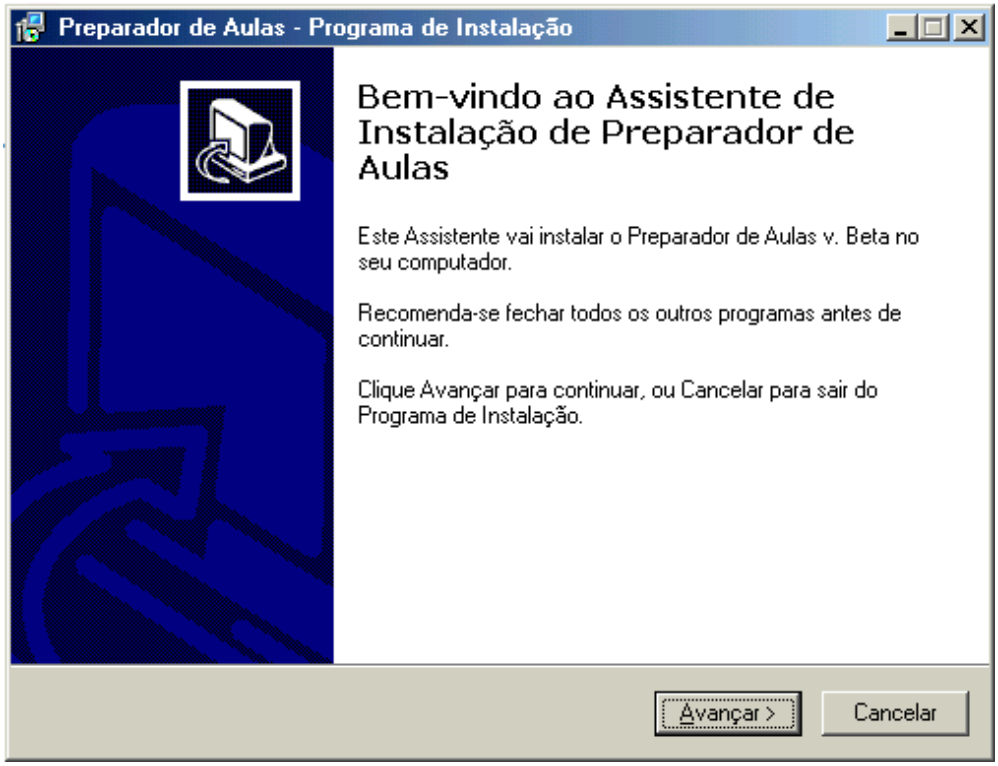


Figura 4.6 – Etapa inicial do Assistente de Instalação do *Software*

Após clicar no botão Avançar, é preciso escolher um caminho para instalar o *software*. Como padrão, o programa de instalação irá instalá-lo na pasta Arquivos de programas. É possível alterar o caminho de instalação clicando no botão Procurar para escolher um outro caminho.

Aqui, utilizaremos o padrão do programa de instalação. Instalaremos o *software* no caminho “C:\Arquivos de programas\Preparador de Aulas”, como podemos ver na figura 3.7. Além do local de instalação, o programa de instalação também mostra a quantidade de espaço em disco necessária para a instalação.

Após escolher o local de instalação, o usuário deve clicar no botão Avançar. Veja a figura 4.7 a seguir.

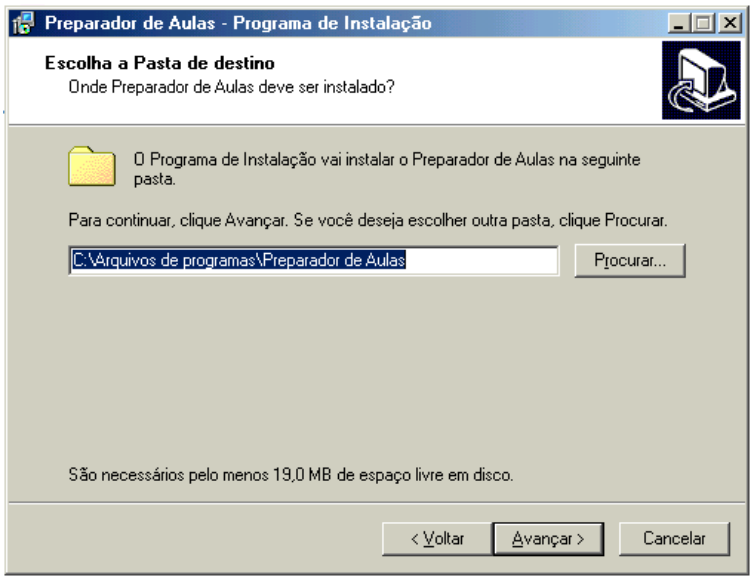


Figura 4.7 – Escolha do local de instalação do *software*

Em seguida, o usuário deve escolher uma pasta do menu Iniciar onde os atalhos das ferramentas do *software* deverão ser instalados. Como padrão, o programa de instalação cria uma pasta no menu Iniciar com o nome do *software*, Preparador de Aulas. É possível escolher uma outra pasta se o usuário desejar. Para isso, o usuário deve clicar no botão Procurar.

Utilizamos novamente o padrão recomendado pelo programa. Criaremos uma Pasta no Menu Iniciar com o nome do *software*, como na figura a seguir.

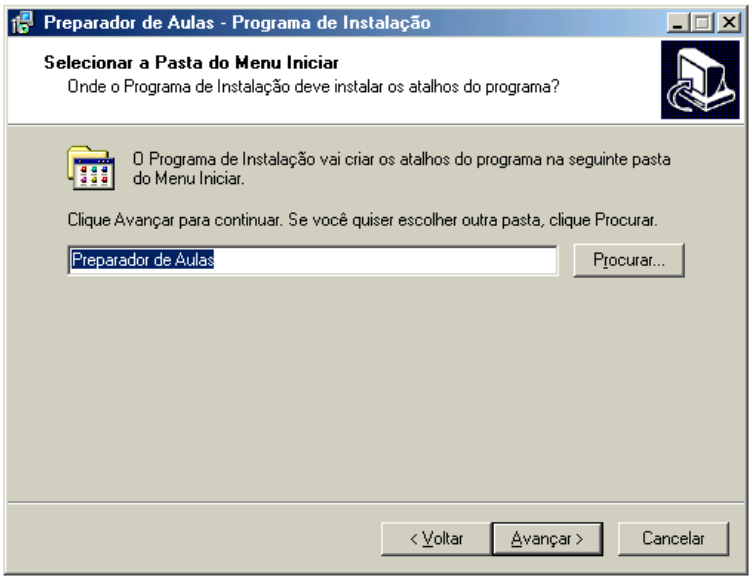


Figura 4.8 – Criação de uma pasta no Menu Iniciar

Na próxima etapa, o usuário pode decidir se deseja a criação de um ícone na Área de Trabalho. Para isso, deve marcar a caixa de verificação e clicar em Avançar para continuar.

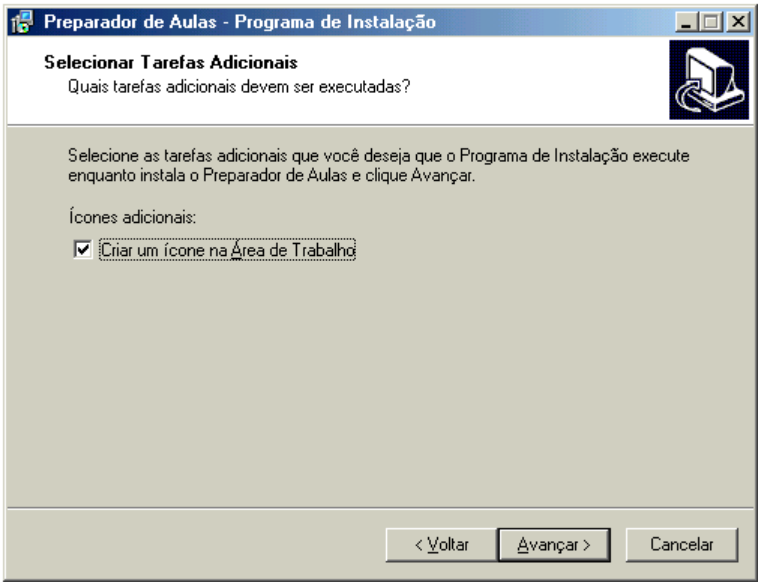


Figura 4.9 – Opção para criar um ícone na Área de Trabalho

Por fim, o programa de instalação exibe as opções de instalação feitas. Essa é a última etapa. Para instalar o *software*, o usuário deve clicar no botão Instalar.

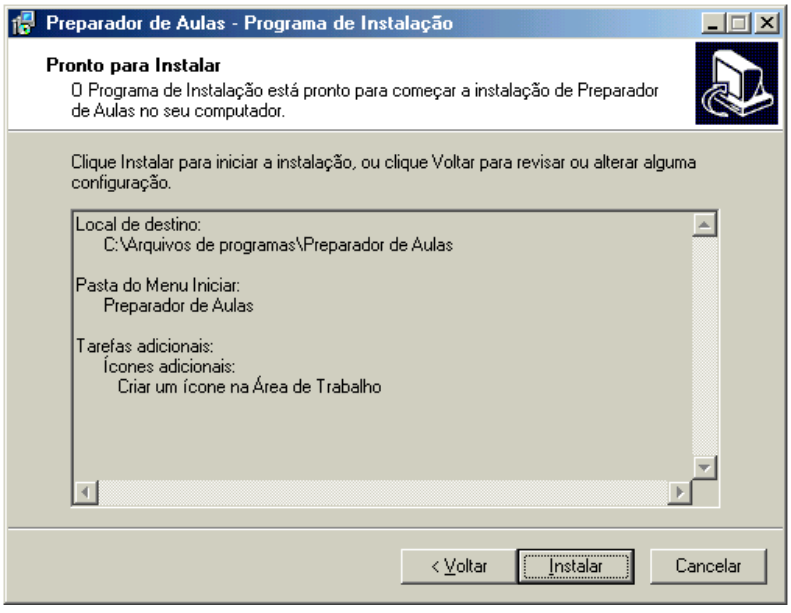


Figura 4.10 – Visualização das informações de instalação

Após clicar no botão Instalar, o programa inicia a instalação dos arquivos no computador. O processo é rápido, leva apenas alguns segundos.

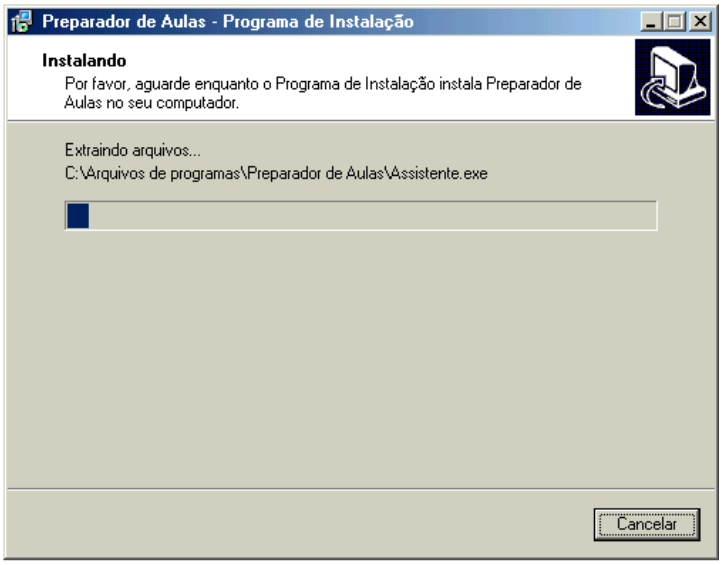


Figura 4.11 – Processo de instalação do *software*

Ao término da cópia dos arquivos, é exibida uma janela que informa o sucesso do processo de instalação e fornece a opção de executar o *software* instalado.

Se o usuário desejar executar o programa logo após a instalação, basta certificar-se de que a caixa de verificação Executar Preparador de Aulas esteja marcada e clicar no botão Concluir.

A figura abaixo mostra o término da instalação.

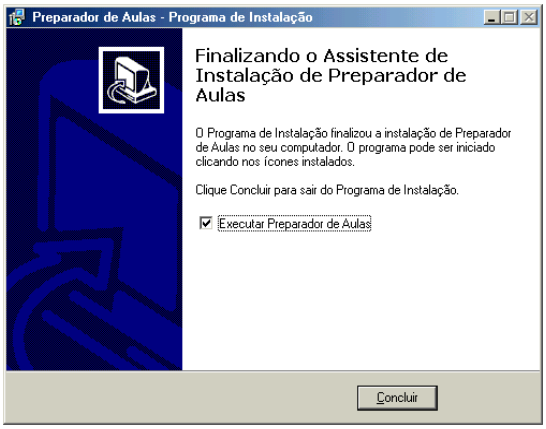


Figura 4.12 – Janela final do Assistente de Instalação

Como especificado em uma das etapas da instalação, o programa cria uma pasta no menu Iniciar com o nome do *software*.

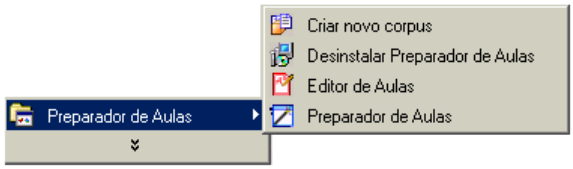


Figura 4.13 – Grupo criado no menu Iniciar

4.2.2 Preparar atividades

Para preparar atividades, utilizamos o Assistente de Preparação de Aulas. Podemos iniciá-lo clicando duas vezes no ícone da Área de Trabalho ou a partir do grupo Preparador de Aulas no menu Iniciar.

Figura 4.14 – ícone do Assistente de Preparação de Aulas

Ao iniciar, duas janelas aparecem: uma é a janela de ajuda do programa e a outra é a janela da primeira etapa para preparação do material. Devemos fechar a janela de ajuda do programa.

A seguinte janela deve aparecer:

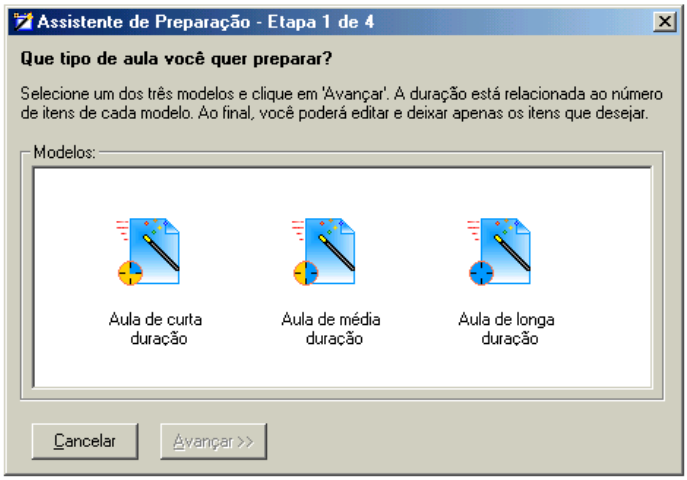


Figura 4.15 – Primeira etapa do Assistente de Preparação: escolha de um modelo de aula

O primeiro passo para preparar a aula é escolher um dos modelos disponíveis. Os modelos são: aula de curta, média e longa duração. A duração mencionada no modelo está relacionada ao número de itens em determinados exercícios da atividade.

Após decidir qual modelo utilizar, e selecioná-lo com um clique, devemos clicar no botão Avançar, que é ativado somente quando um dos modelos é clicado. Para a presente demonstração, escolhemos o primeiro modelo, ou seja, a Aula de curta duração.

Na segunda etapa do Assistente, devemos selecionar um texto para ser utilizado na preparação da atividade. A ferramenta fornece duas opções para a seleção do texto a ser utilizado na preparação da aula. Podemos utilizar um texto da Biblioteca, previamente adicionado na ferramenta Biblioteca ou, se o texto desejado não estiver na caixa de lista em Textos da biblioteca, clicar no botão Selecionar do arquivo para procurar o texto em alguma pasta ou diretório. É importante que o texto a ser utilizado esteja em formato .txt (sem formatação), que é o formato de arquivo que o programa lê para preparar a aula.

Na figura 4.16, mostramos a seleção de um texto já adicionado à Biblioteca. O texto utilizado nesta demonstração será mostrado na próxima subseção.

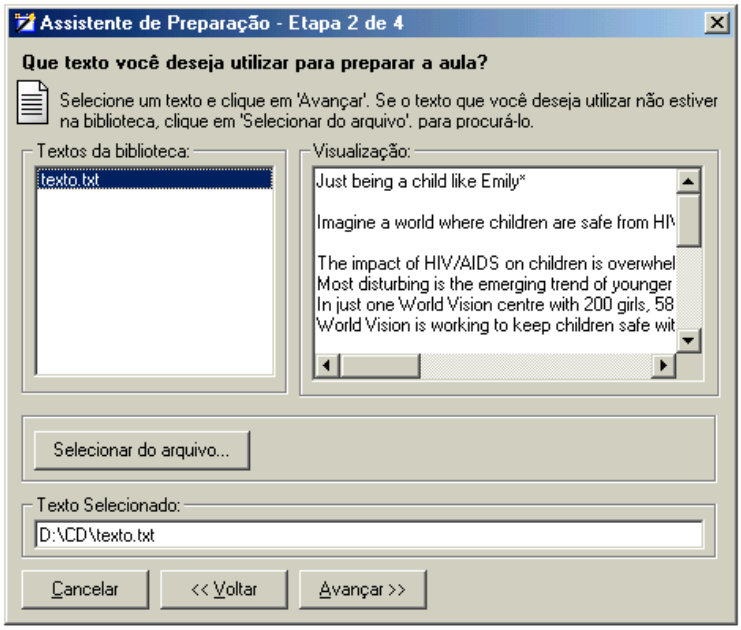


Figura 4.16 – Segunda etapa do Assistente de Preparação: escolha de um texto

Depois de selecionar o texto, clicamos no botão Avançar para continuar. O Assistente inicia o processamento do texto. O processamento do texto consiste na contagem das estatísticas do texto, etiquetagem das palavras, identificação de palavras-chave, palavras cognatas e outras informações importantes que servem de base para a preparação da aula.

Em poucos segundos, a janela da terceira etapa do Assistente aparece. As informações exibidas para o texto escolhido podem ser visualizadas na figura abaixo:

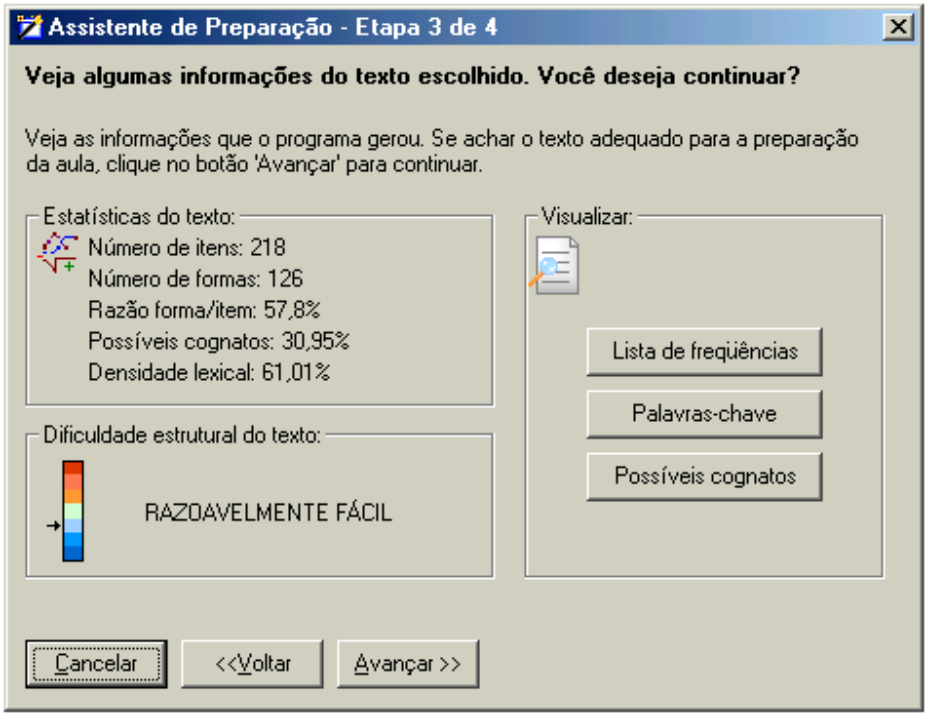


Figura 4.17 – Terceira etapa do Assistente de Preparação: Informações do texto escolhido

Nessa etapa, podemos verificar algumas informações importantes sobre o texto e decidir se desejamos continuar na preparação da aula ou escolher um novo texto para utilizar. As informações disponíveis são:

- Número de itens – número total de palavras do texto (*tokens*).
- Número de formas – número de palavras não repetidas (*types*).
- Razão forma/item – variedade vocabular do texto. Quanto maior o valor, mais variado é o vocabulário.

- Dificuldade estrutural do texto – classificação definida por uma fórmula que leva em consideração o tamanho das palavras e frases do texto.
- Lista de freqüências – lista das palavras mais freqüentes do texto, exibe o número de ocorrências e a porcentagem de cada palavra no texto.
- Palavras-chave – lista das palavras que mais se destacam no texto, geralmente revelando sua temática.
- Possíveis cognatos – lista de palavras do texto que são parecidas com o português, identificadas a partir de uma comparação com banco de dados de palavras definidas como cognatas.

Após verificar as informações geradas pelo programa e decidir continuar a preparação do material, clicamos no botão Avançar para ir para a próxima etapa.

A tela exibida para esta etapa é a seguinte:

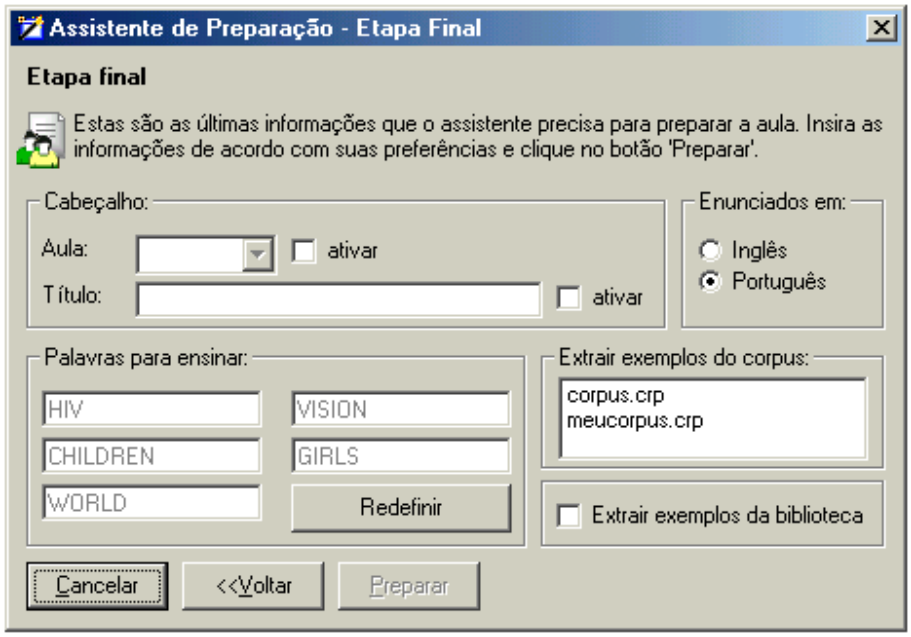


Figura 4.18 – Última etapa do Assistente de Preparação: últimas informações necessárias

Na última etapa do Assistente, especificamos as últimas informações necessárias para a preparação da aula. Em Cabeçalho, definimos o título da aula e o número, que são informações opcionais, pode-se deixar em branco se desejado. Também é possível escolher em que língua a ferramenta deve criar os enunciados da atividade, inglês ou português. Em Palavras para ensinar,

definimos quais palavras ou expressões do texto queremos ensinar, no total são cinco palavras. O Assistente traz as cinco primeiras palavras-chave como sugestão. O usuário pode alterar a definição padrão, clicando no botão Redefinir e digitando nas caixas de texto as novas palavras ou expressões. O programa utilizará as palavras sugeridas para fazer a busca em um corpus definido pelo usuário em Extrair exemplos do corpus e criará exercícios com linhas de concordância.

Uma outra fonte de exemplos é a Biblioteca. Pode-se marcar a caixa de verificação Extrair exemplos da biblioteca para que o programa busque os exemplos nos textos armazenados pelo usuário na biblioteca. É importante selecionar uma das duas opções de extração de exemplos, visto que o botão Preparar só é ativado após a escolha de uma das duas opções.

Para o propósito desta pesquisa, alteramos as opções de cabeçalho e escolhemos extrair os exemplos do corpus. As demais opções não foram alteradas.

Em seguida, clicamos no botão Preparar. O programa inicia o processo de preparação da aula. Uma barra de progresso aparece e, em poucos segundos, o Assistente traz a aula pronta.

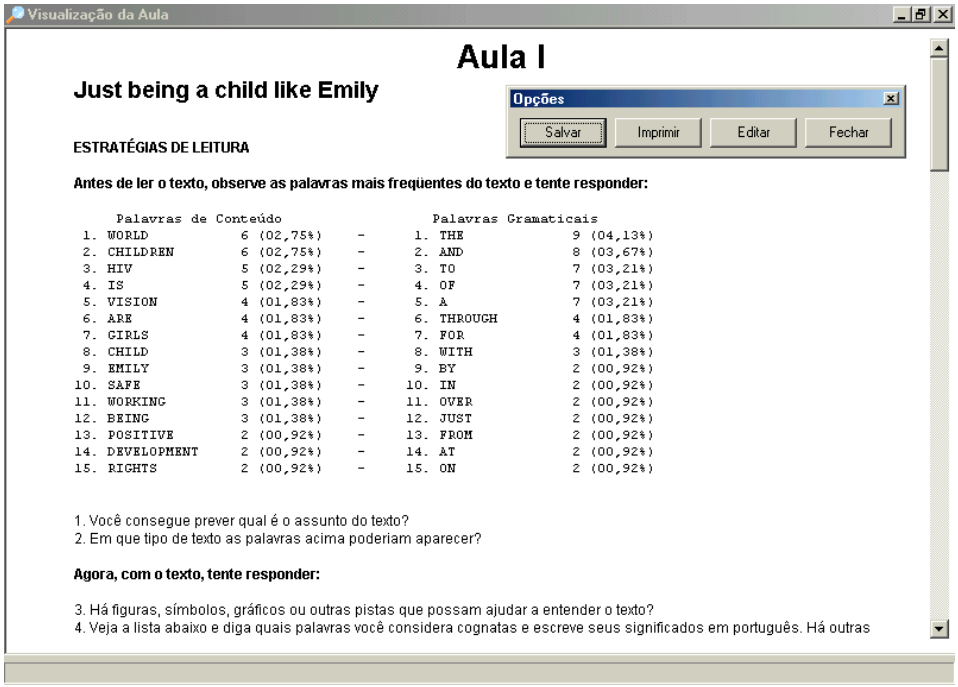


Figura 4.19 – Visualização da aula preparada pelo Assistente

Na barra suspensa Opções, é possível salvar o material criado pelo programa, imprimir o que é visualizado na tela ou editar utilizando o Editor de Aulas. Apresentamos abaixo um exemplo de atividade, preparada semi-automaticamente pelo *software*. Dividimos a atividade em duas partes para melhor visualização na página. Veja a primeira parte:

Aula I

Exemplo de atividade

ESTRATÉGIAS DE LEITURA

Antes de ler o texto, observe as palavras mais freqüentes do texto e tente responder:

Palavras de Conteúdo			Palavras Gramaticais	
1. WORLD	6 (02,75%)	-	1. THE	9 (04,13%)
2. CHILDREN	6 (02,75%)	-	2. AND	8 (03,67%)
3. HIV	5 (02,29%)	-	3. TO	7 (03,21%)
4. IS	5 (02,29%)	-	4. OF	7 (03,21%)
5. VISION	4 (01,83%)	-	5. A	7 (03,21%)
6. ARE	4 (01,83%)	-	6. THROUGH	4 (01,83%)
7. GIRLS	4 (01,83%)	-	7. FOR	4 (01,83%)
8. CHILD	3 (01,38%)	-	8. WITH	3 (01,38%)
9. EMILY	3 (01,38%)	-	9. BY	2 (00,92%)
10. SAFE	3 (01,38%)	-	10. IN	2 (00,92%)
11. WORKING	3 (01,38%)	-	11. OVER	2 (00,92%)
12. BEING	3 (01,38%)	-	12. JUST	2 (00,92%)
13. POSITIVE	2 (00,92%)	-	13. FROM	2 (00,92%)
14. DEVELOPMENT	2 (00,92%)	-	14. AT	2 (00,92%)
15. RIGHTS	2 (00,92%)	-	15. ON	2 (00,92%)

1. Você consegue prever qual é o assunto do texto?
2. Em que tipo de texto as palavras acima poderiam aparecer?

Agora, com o texto, tente responder:

3. Há figuras, símbolos, gráficos ou outras pistas que possam ajudar a entender o texto?
4. Veja a lista abaixo e diga quais palavras você considera cognatas e escreve seus significados em português. Há outras palavras cognatas no texto? Dê a tradução.

- VISION - AIDS - JUSTICE - PSEUDONYM - POSITIVE - VIRGINS - ASSISTS - SEXUALLY - TRANSMITTED - DISTURBING - SPONSORSHIP - DEVELOPMENT - ACTIVELY - EMERGING - RESTORE -

5. Essas são as palavras de mais destaque no texto. Quais palavras você conhece? Tente agrupá-las de acordo com algum tipo de padrão de significado.

- HIV - VISION - CHILDREN - GIRLS - WORLD - EMILY - SAFE - AIDS - CHILD - ORG - WORKING - RELIEF - JUSTICE - PSEUDONYM - POSITIVE -

6. Que tipo de texto é este?
7. Qual é o layout do texto e como se refere ao tipo de texto?
8. Qual é o objetivo do texto?
9. Qual é o público-alvo deste tipo de texto? Quem estaria interessado em ler este tipo de texto?
10. Qual é a idéia geral do texto? Confronte com sua idéia inicial na pergunta 3.
11. Quais são as principais idéias do texto? Em que parte do texto estão?
12. O computador acha que o texto tem 57,8% de palavras repetidas, 30,95% palavras cognatas e classifica o texto como razoavelmente fácil. Você concorda? Justifique.

Figura 4.20 – Primeira parte da atividade

Veja a segunda parte da atividade:

LÉXICO-GRAMÁTICA

1. Identifique se as palavras abaixo possuem prefixos ou sufixos e quais são.

SPONSORSHIP RESTORE EXTREME DEVELOPMENT EXPLOITATION

2. Qual é a função das seguintes palavras no texto. Dê a classe a classe gramatical.

	Significado	Classe gramatical
THAT	.....	.....
THAN	.....	.....
AND	.....	.....
AS	.....	.....

3. Dê a referência contextual das palavras abaixo:

WHO	.....
YOU	.....

4. Veja os exemplos abaixo e tente descobrir o significado de cada palavra centralizada. O significado das palavras nos exemplos é o mesmo que no texto?

1. manded an end to the "sexploitation" of children through commercial marketing.
2. ly targeting their campaigns to exploit children's 'pester power'." The
3. escribed as the "sexploitation of young children" - such as Tesco marketing a
4. selling pink and black lace lingerie to children and Next being forced to
5. ions introduced this week will mean all children's TV channels having to ban
6. be extended to other channels covering children's programmes by January 2008.
7. ternet, email and publications aimed at children". Asda said it had stopped
8. , and when I changed his nappy my other children would have to hold his arms
9. Published: 06 April 2007 Young Muslim children attending after-school
10. ish values could begin with the 100,000 children, aged five to 14, who go to
11. part of the national curriculum for all children from the age of seven.
12. to bring in compulsory lessons for all children from the age of seven but
13. ts in the UK spend less time with their children than those in any other
14. to a "greater institutionalisation" of children, the Association of Teachers
15. ssioned by the Government revealed that children in full-time nursery settings

1. ing forced to remove T-shirts for young girls with the slogan "so many young
2. ith The Young Ones, Three Of A Kind and Girls On Top among his producing
3. ing out with one half of the the Cheeky Girls, he's faced regular drubbings at
4. ality TV show called Killing the Cheeky Girls. The article, which appeared in
5. fellow countrywomen: "I love the Valley girls. They're so liberated."
6. aisle that was lined with young Chechen girls in white wedding dresses on one

LEITURA CRÍTICA

1. Qual é sua opinião sobre o texto? Você concorda com o que está escrito?

Figura 4.21 – Segunda parte da atividade

O exemplo de atividade apresentado foi preparado semi-automaticamente pelo *software*. Apenas selecionamos um texto, um título para atividade e o corpus para extrair os exemplos para o exercício com linhas de concordância. As únicas alterações feitas ao exemplo foram dividi-lo em duas partes e diminuir o número de linhas de concordância para caber na página.

O texto utilizado na preparação foi um retirado da revista Newsweek, uma das fontes de textos que costumo utilizar para preparação de atividades de leitura. O texto foi passado para o formato eletrônico e salvo com a extensão.txt, procedimento necessário para utilização de textos com o *software*. Se o usuário desejar utilizar o texto com sua formatação original, deve salvar uma cópia. O texto é apresentado logo abaixo:

Just being a child like Emily*

Imagine a world where children are safe from HIV/AIDS…

The impact of HIV/AIDS on children is overwhelming. Girls like Emily are at extreme risk, due to sexual abuse and exploitation, and also through a lack of knowledge about safe practices.

Most disturbing is the emerging trend of younger girls being raped by men who are HIV positive. Virgins are being mythologised as a "cure" for HIV/AIDS.

In just one World Vision centre with 200 girls, 58% had sexually transmitted diseases and 18% were HIV positive. The average age of these girls was 14 years, 2 months.

World Vision is working to keep children safe with experience gained from over 2000 projects that assist children around the world.

Join a World Vision for children

World Vision is a Christian relief and development partnership which assists more than 75 million people in over 90 countries, working to restore hope and to bring justice.

Through child sponsorship. Through relief and development, working with communities to build food security and to fight poverty.

By actively building awareness of children's rights, seeking justice and fighting for the protection of children. Through support for the United Nations Convention on the Rights of the Child.

Find out ways you can help at www.wvi.org

*Emily is a pseudonym to protect the child's identity.

Quadro 4.1 – Texto utilizado para preparar a aula

A atividade padrão inclui uma série de exercícios relacionados ao texto escolhido pelo usuário na preparação. Após a preparação da atividade feita pelo Assistente, o usuário pode decidir se imprime a atividade como o *software* a apresenta ou faz a edição da atividade de acordo com sua vontade.

O usuário não é obrigado a utilizar todos os exercícios do modelo. Uma das preocupações no processo de desenvolvimento do *software* era caracterizá-lo como uma ferramenta flexível para o usuário. Portanto, o usuário é livre para editar e selecionar dentre uma série, apenas os exercícios desejados, dependendo do objetivo em mente. É importante salientar que o termo ‘aula’, empregado no *software* pode ser entendido como material, que pode ser usado em uma ou mais aulas, e também não precisa ser o único material usado em qualquer parte da aula.

Tendo em vista também que grande parte da preparação da atividade é semi-automática, é possível que em algumas ocasiões o programa gere exercícios com erros, por exemplo, incluir na lista de palavras gramaticais uma palavra de conteúdo. Nesses casos, o usuário pode fazer a correção na atividade ou deixar para que o próprio aluno identifique o erro, considerando-o como parte do exercício.

O objetivo principal do desenvolvimento do *software* é que possa ser usado como uma ferramenta pelo usuário. Embora o *software* possa trazer a atividade praticamente pronta, outros passos e etapas devem ser levados em consideração para que seja uma ferramenta efetiva. Da mesma forma que utilizamos a vara para pescar o peixe, é preciso também saber, e escolher, a melhor maneira de limpá-lo, prepará-lo e, por fim, servi-lo.

Neste capítulo, foi apresentado o *software* construído, bem como a demonstração de sua utilização. No próximo capítulo, fazemos uma análise das principais funções desempenhadas pelo *software* na preparação de uma atividade criada pelo programa e apresentamos as primeiras impressões de seu uso por professores.

Capítulo 5

Teste e avaliação do *software*

Neste capítulo, são apresentados os resultados da análise das principais funções desempenhadas pelo *software*. Em seguida, são apresentadas as primeiras impressões do público para o qual o *software* foi criado. Por fim, é feita uma discussão crítica dos resultados como um todo.

5.1 Análise dos resultados fornecidos pelo *software*

Nesta seção, fazemos uma avaliação das principais informações geradas pelo *software*. A avaliação pretende verificar a performance das funções de análise lingüística executadas automaticamente pelo *software*. Para isso, fazemos o contraste dos resultados gerados automaticamente pelo *software* com os resultados provenientes de uma análise manual, conforme descrito na metodologia.

Conforme dissemos no capítulo 2, trabalhamos com a seguinte escala de valores para avaliar o desempenho de cada função em ambos os critérios de abrangência e precisão:

Valores	**Avaliação do desempenho**
0% a 59%	ruim
60% a 69%	regular
70% a 79%	bom
80% a 89%	muito bom
90% a 99%	excelente
100%	perfeito

Quadro 5.1 – Critérios de avaliação do desempenho do *software*

5.1.1 Contagem do número de palavras no texto

O quadro 5.2 mostra o resultado da contagem de palavras feita pelo *software*.

1 THE	9	04,13	43 YOUNGER	1	00,46	85 OUT	1	00,46
2 AND	8	03,67	44 CENTRE	1	00,46	86 WAYS	1	00,46
3 A	7	03,21	45 AVERAGE	1	00,46	87 YOU	1	00,46
4 TO	7	03,21	46 IMAGINE	1	00,46	88 CAN	1	00,46
5 OF	7	03,21	47 WHERE	1	00,46	89 HELP	1	00,46
6 WORLD	6	02,75	48 AIDS…	1	00,46	90 WWW	1	00,46
7 CHILDREN	6	02,75	49 IMPACT	1	00,46	91 WVI	1	00,46
8 IS	5	02,29	50 OVERWHELMING	1	00,46	92 ORG	1	00,46
9 HIV	5	02,29	51 EXTREME	1	00,46	93 PSEUDONYM	1	00,46
10 THROUGH	4	01,83	52 RISK	1	00,46	94 PROTECT	1	00,46
11 FOR	4	01,83	53 DUE	1	00,46	95 CHILD'S	1	00,46
12 GIRLS	4	01,83	54 ALSO	1	00,46	96 SUPPORT	1	00,46
13 VISION	4	01,83	55 ABUSE	1	00,46	97 AROUND	1	00,46
14 ARE	4	01,83	56 TREND	1	00,46	98 TRANSMITTED	1	00,46
15 WORKING	3	01,38	57 EXPLOITATION	1	00,46	99 WERE	1	00,46
16 SAFE	3	01,38	58 HAD	1	00,46	100 IDENTITY	1	00,46
17 WITH	3	01,38	59 LACK	1	00,46	101 AGE	1	00,46
18 BEING	3	01,38	60 KNOWLEDGE	1	00,46	102 THESE	1	00,46
19 CHILD	3	01,38	61 ABOUT	1	00,46	103 WAS	1	00,46
20 EMILY	3	01,38	62 PRACTICES	1	00,46	104 YEARS	1	00,46
21 RELIEF	2	00,92	63 MOST	1	00,46	105 MONTHS	1	00,46
22 BY	2	00,92	64 DISTURBING	1	00,46	106 KEEP	1	00,46
23 POSITIVE	2	00,92	65 EMERGING	1	00,46	107 EXPERIENCE	1	00,46
24 OVER	2	00,92	66 SEXUAL	1	00,46	108 GAINED	1	00,46
25 LIKE	2	00,92	67 NATIONS	1	00,46	109 PROJECTS	1	00,46
26 FROM	2	00,92	68 DISEASES	1	00,46	110 BUILD	1	00,46
27 JUSTICE	2	00,92	69 FOOD	1	00,46	111 ASSIST	1	00,46
28 AIDS	2	00,92	70 SECURITY	1	00,46	112 SEXUALLY	1	00,46
29 RIGHTS	2	00,92	71 FIGHT	1	00,46	113 JOIN	1	00,46
30 IN	2	00,92	72 POVERTY	1	00,46	114 CHRISTIAN	1	00,46
31 JUST	2	00,92	73 ACTIVELY	1	00,46	115 PARTNERSHIP	1	00,46
32 DEVELOPMENT	2	00,92	74 BUILDING	1	00,46	116 WHICH	1	00,46
33 AT	2	00,92	75 AWARENESS	1	00,46	117 ASSISTS	1	00,46
34 ON	2	00,92	76 CHILDREN'S	1	00,46	118 MORE	1	00,46
35 AS	1	00,46	77 SEEKING	1	00,46	119 THAN	1	00,46
36 MYTHOLOGISED	1	00,46	78 FIGHTING	1	00,46	120 MILLION	1	00,46
37 VIRGINS	1	00,46	79 PROTECTION	1	00,46	121 PEOPLE	1	00,46
38 ONE	1	00,46	80 COMMUNITIES	1	00,46	122 COUNTRIES	1	00,46
39 "CURE"	1	00,46	81 UNITED	1	00,46	123 RESTORE	1	00,46
40 WHO	1	00,46	82 SPONSORSHIP	1	00,46	124 HOPE	1	00,46
41 MEN	1	00,46	83 CONVENTION	1	00,46	125 BRING	1	00,46
42 RAPED	1	00,46	84 FIND	1	00,46	126 THAT	1	00,46

Quadro 5.2 – Lista de freqüência das palavras do texto gerada pelo *software*

Para a contagem, estamos considerando o número total de palavras do texto, isto é, o número de *tokens*.

Cálculo do valor de precisão:

Número de palavras da contagem manual: 218
Número de palavras da contagem automática: 218
Número de itens irrelevantes: 0
Precisão = (218/(218+0))*100
Precisão = 100% (perfeito)

Cálculo do valor de abrangência:

Número de palavras da contagem automática: 218
Número de itens relevantes não identificados: 0
Abrangência = (218/(218+0))*100
Abrangência = 100% (perfeito)

Segundo nossa escala de avaliação (quadro 5.1), os resultados para o texto indicam um desempenho perfeito em ambos os valores de precisão e abrangência.

5.1.2 Contagem do número de parágrafos

Cálculo do valor de precisão:

Número de parágrafos da contagem manual: 7
Número de parágrafos da contagem automática: 8
Número de itens irrelevantes: 1
Precisão = (7/(8+1))*100
Precisão = 78% (bom)

Cálculo do valor de abrangência:

Número de parágrafos da contagem automática: 8
Número de itens relevantes não identificados: 0
Abrangência = (8/(8+0))*100
Abrangência = 100% (perfeito)

Segundo nossa escala de avaliação, os resultados indicam um desempenho bom em relação ao valor de precisão e perfeito em relação ao valor de abrangência.

5.1.3 Contagem do número de frases

A contagem manual do número de frases foi realizada de acordo com a função criada.

Cálculo do valor de precisão:

Número de frases da contagem manual: 11
Número de frases da contagem automática: 10
Número de itens irrelevantes: 0
Precisão = (10/(10+0))*100
Precisão = 100% (perfeito)

Cálculo do valor de abrangência:

Número de frases da contagem automática: 10
Número de itens relevantes não identificados: 1
Abrangência = (10/(10+1))*100
Abrangência = 91% (excelente)

Segundo nossa escala de avaliação, os resultados indicam um desempenho perfeito em relação ao valor de precisão e excelente em relação ao valor de abrangência.

5.1.4 Contagem do número de sílabas

A contagem manual do número de sílabas foi feita levando-se em consideração o tipo de contagem realizada pela função criada para fazer a estimativa do número de sílabas, isto é, uma contagem baseada em caracteres.

Cálculo do valor de precisão:

Número de sílabas identificadas manualmente: 364
Número de sílabas identificadas automaticamente: 344
Número de itens irrelevantes: 0 (desconhecido)
Precisão = (344/(344+0))*100
Precisão = 100% (perfeito)

Cálculo do valor de abrangência:

Número de sílabas identificadas manualmente: 364
Número de sílabas identificadas automaticamente: 344
Número de itens relevantes não identificados: 20
Abrangência = (344/(344+20))*100
Abrangência = 95% (excelente)

Segundo nossa escala de avaliação, os resultados indicam um desempenho perfeito em relação ao valor de precisão e excelente em relação ao valor de abrangência.

5.1.5 Palavras cognatas identificadas

Palavras cognatas identificadas manualmente pelo pesquisador:

1 ABUSE	13 EXPERIENCE	25 PROJECTS
2 ACTIVELY	14 EXPLOITATION	26 PROTECT
3 AIDS	15 EXTREME	27 PROTECTION
4 ASSIST	16 HIV	28 PSEUDONYM
5 ASSISTS	17 IDENTITY	29 RISK
6 CENTRE	18 IMAGINE	30 SEXUAL
7 COMMUNITIES	19 IMPACT	31 SEXUALLY
8 CONVENTION	20 JUSTICE	32 SUPPORT
9 CHRISTIAN	21 MILLION	33 TRANSMITTED
10 CURE	22 NATIONS	34 VIRGINS
11 DISTURBING	23 POSITIVE	35 VISION
12 EMERGING	24 PRACTICES	

Quadro 5.3 – Palavras cognatas identificadas manualmente

Palavras cognatas identificadas automaticamente:

1 ABUSE	14 EXPERIENCE	27 PROTECTION
2 ACTIVELY	15 EXPLOITATION	28 PSEUDONYM
3 AIDS	16 EXTREME	29 RESTORE
4 ASSIST	17 IDENTITY	30 SECURITY
5 ASSISTS	18 IMAGINE	31 SEXUAL
6 CENTRE	19 IMPACT	32 SEXUALLY
7 CHRISTIAN	20 JUSTICE	33 SPONSORSHIP
8 COMMUNITIES	21 MILLION	34 SUPPORT
9 CONVENTION	22 NATIONS	35 THESE
10 COUNTRIES	23 POSITIVE	36 TRANSMITTED
11 DEVELOPMENT	24 PRACTICES	37 UNITED
12 DISTURBING	25 PROJECTS	38 VIRGINS
13 EMERGING	26 PROTECT	39 VISION

Quadro 5.4 – Palavras cognatas identificadas automaticamente

Na contagem feita manualmente, identificamos 35 possíveis palavras cognatas no texto e na análise automática, foram identificadas 39 palavras cognatas. A análise automática, embora tenha identificado um número maior de palavras, não identificou todas as palavras identificadas pela análise manual. As palavras CURE, HIV e RISK não foram identificadas. Talvez uma

explicação para isso seja o fato de que a comparação é feita com palavras que possuam cinco ou mais caracteres. A análise automática também identificou 6 palavras que não estão na lista da análise manual: COUNTRIES, DEVELOPMENT, RESTORE, SECURITY, SPONSORSHIP, UNITED e THESE.

Cálculo do valor de precisão:

Número de palavras cognatas identificadas manualmente: 35
Número de palavras cognatas da contagem automática: 39
Número de itens relevantes identificados: 32
Número de itens irrelevantes: 7
Precisão = (32/(32+7))*100
Precisão = 82% (muito bom)

Cálculo do valor de abrangência:

Número de palavras cognatas da contagem automática: 39
Número de itens relevantes não identificados: 3
Abrangência = (39/(39+3))*100
Abrangência = 93% (excelente)

Segundo nossa escala de avaliação, os resultados indicam um desempenho muito bom em relação ao valor de precisão e excelente em relação ao valor de abrangência.

5.1.6 Etiquetagem da lista de palavras

Para verificar os resultados da etiquetagem automática foi realizada uma contagem manual pelo pesquisador. O quadro abaixo exibe a etiquetagem automática feita pelo *software*.

N	Palavra	Etiq.	N	Palavra	Etiq.	N	Palavra	Etiq.
1	EXTREME	aj0	43	CONVENTION	nn1	85	OF	prf
2	SEXUAL	aj0	44	IMPACT	nn1	86	ABOUT	prp
3	CHRISTIAN	aj0	45	SECURITY	nn1	87	BY	prp
4	POSITIVE	aj0	46	AIDS	nn1	88	LIKE	prp
5	OVERWHELMING	aj0	47	PSEUDONYM	nn1	89	FROM	prp
6	AVERAGE	aj0	48	CHILD	nn1	90	ON	prp
7	DUE	aj0	49	WORLD	nn1	91	AT	prp
8	SAFE	aj0	50	KNOWLEDGE	nn1	92	THROUGH	prp
9	YOUNGER	ajc	51	TREND	nn1	93	FOR	prp
10	A	at0	52	AIDS…	nn1	94	IN	prp
11	THE	at0	53	RISK	nn1	95	WITH	prp
12	SEXUALLY	av0	54	LACK	nn1	96	OVER	prp
13	ACTIVELY	av0	55	FOOD	nn1	97	AROUND	prp
14	MOST	av0	56	POVERTY	nn1	98	TO	to0
15	ALSO	av0	57	BUILDING	nn1	99	ARE	vbb
16	MORE	av0	58	AWARENESS	nn1	100	WERE	vbd
17	JUST	av0	59	CHILDREN'S	nn1	101	WAS	vbd
18	OUT	avp	60	ORG	nn1	102	BEING	vbg
19	WHERE	avq-cjs	61	CHILD'S	nn1	103	IS	vbz
20	AND	cjc	62	"CURE"	nn1	104	HAD	vhd
21	THAN	cjs	63	AGE	nn1	105	CAN	vm0
22	AS	cjs	64	RELIEF	nn1	106	HOPE	vvb
23	THAT	cjt	65	PARTNERSHIP	nn1	107	DISTURBING	vvg
24	MILLION	crd	66	PROJECTS	nn2	108	EMERGING	vvg
25	ONE	crd	67	COUNTRIES	nn2	109	FIGHTING	vvg
26	THESE	dt0	68	VIRGINS	nn2	110	SEEKING	vvg
27	WHICH	dtq	69	COMMUNITIES	nn2	111	WORKING	vvg
28	PEOPLE	nn0	70	PRACTICES	nn2	112	RESTORE	vvi
29	WWW	nn1	71	NATIONS	nn2	113	ASSIST	vvi
30	WVI	nn1	72	MEN	nn2	114	PROTECT	vvi
31	MYTHOLOGISED	nn1	73	GIRLS	nn2	115	IMAGINE	vvi
32	SPONSORSHIP	nn1	74	CHILDREN	nn2	116	BRING	vvi
33	JUSTICE	nn1	75	DISEASES	nn2	117	BUILD	vvi
34	DEVELOPMENT	nn1	76	WAYS	nn2	118	FIGHT	vvi
35	ABUSE	nn1	77	RIGHTS	nn2	119	FIND	vvi
36	EXPLOITATION	nn1	78	MONTHS	nn2	120	HELP	vvi
37	EXPERIENCE	nn1	79	YEARS	nn2	121	KEEP	vvi
38	IDENTITY	nn1	80	UNITED	np0	122	JOIN	vvi
39	CENTRE	nn1	81	EMILY	np0	123	TRANSMITTED	vvn
40	VISION	nn1	82	HIV	np0	124	RAPED	vvn
41	PROTECTION	nn1	83	YOU	pnp	125	GAINED	vvn
42	SUPPORT	nn1	84	WHO	pnq	126	ASSISTS	vvz

Quadro 5.5 – Lista de Palavras etiquetadas

Cálculo do valor de precisão:
Número de palavras da lista etiquetadas: 126
Número de palavras etiquetadas corretamente: 119
Número de palavras etiquetadas incorretamente: 7
Precisão = (119/(119+7))*100
Precisão = 94% (excelente)

Cálculo do valor de abrangência:
Número de palavras da lista etiquetadas: 126
Número de palavras etiquetadas corretamente: 119
Número de palavras etiquetadas incorretamente: 7
Abrangência = (126/(126+7))*100
Abrangência = 95% (excelente)

Segundo nossa escala de avaliação, os resultados indicam um desempenho excelente em relação ao valor de precisão e também em relação ao valor de abrangência.

5.1.7 Resultado geral

O quadro abaixo mostra os valores de precisão e abrangência de todas as funções avaliadas anteriormente.

Função	Precisão	Abrangência
Contagem do número de palavras	100% (perfeito)	100% (perfeito)
Contagem do número de parágrafos	78% (bom)	100% (perfeito)
Contagem do número de frases	100% (perfeito)	91% (muito bom)
Contagem do número de sílabas	100% (perfeito)	95% (excelente)
Identificação de palavras cognatas	82% (muito bom)	93% (excelente)
Etiquetagem das palavras	94% (excelente)	95% (excelente)
Média	**92% (excelente)**	**95% (excelente)**

Quadro 5.6 – Média dos valores de precisão e abrangência

A média dos resultados indica que o desempenho no processamento do texto utilizado para a preparação da aula é excelente nas medidas de precisão e abrangência. O *software* não teve nenhuma avaliação ruim ou regular. O resultado é um bom indicativo de que o *software* pode ser utilizado para os fins pelos quais foi criado.

Um ponto importante a considerar, em relação à avaliação realizada, é o fato de que as análises feitas pelo *software* não são perfeitas, e o professor deve ter esta consciência. Como qualquer outro *software* de análise de corpus, o objetivo principal não é atingir a 'perfeição', porque sempre há uma margem de erro para análises automáticas.

5.2 Primeiras impressões do uso do *software*

Ao término da criação do *software*, em uma primeira oportunidade, o produto final foi apresentado para duas professoras: uma professora do Ensino Médio da escola pública e uma professora universitária, colega do grupo de orientandos do Professor Dr.Tony Berber Sardinha, que se ofereceu voluntariamente para utilizar o *software*. As duas professoras utilizaram o programa e enviaram por e-mail suas primeiras impressões.

É importante fazer uma observação sobre como foi obtido o relato da professora de inglês do Ensino Médio. O *software* foi apresentado à professora, juntamente com outras três professoras do Ensino Médio, no final de uma orientação técnica sobre um dos projetos da Oficina Pedagógica da Diretoria de Ensino da Região Leste 3. A princípio, o objetivo era gravar o relato das professoras sobre o *software* por meio de um gravador de voz, mas devido a um problema no aparelho e aos poucos minutos que tinha, não foi possível gravá-lo. Assim, para que não fosse perdida a oportunidade, pedi para que as professoras enviassem por e-mail suas primeiras impressões sobre o uso do *software*, ao fazê-lo mencionei oralmente algumas questões para que as professoras tivessem uma idéia alguns pontos para avaliar o *software*, as quais não registrei naquele momento.

A seguir, mostramos abaixo o relato da única professora a enviar a tarefa por e-mail.

Muito bem elaborado, que ajudará muito as aulas de Inglês.
Gostaria de ter um programa deste em meu computador, pois posso criar e recriar exercícios com a maior facilidade, porque ele facilitará a minha vida, quando eu elaboro as minhas aulas, e muito simples de usar.
Sim, deste que você tenha trabalhado os pré-requisitos, ou seja, a gramática e o que será trabalhado em seus textos.
Maravilhoso, porque na disciplina de Inglês não temos nenhum recurso que facilite.
É claro, você adaptará de acordo com suas necessidades.

Quadro 5.7 – Relato de uma professora de inglês da escola pública

Embora o programa tenha como público-alvo alunos do Ensino Médio, também pode ser utilizado em outros contextos de ensino. Apresento aqui as considerações feitas pela professora universitária que preparou a aula com o *software* e aplicou a atividade com seus alunos. Segue abaixo o seu relato:

Nas questões

5. Essas são as palavras de mais destaque no texto. Quais palavras você conhece? Tente agrupá-las de acordo com algum tipo de padrão de significado.

Tive que dar um exemplo de padrão de significado...os alunos não entenderam o enunciado (normal rrss).

12. O computador acha que o texto tem 48,86% de palavras repetidas, 6,2% palavras cognatas e classifica o texto como razoavelmente fácil. Você concorda? Justifique.

Meus alunos estranharam os números e ...digamos...como são de humanas quiseram contar as palavras, fazer contas e reclamaram um pouquinho. Aí entrei como 'lingüista de corpus' para explicar o conceito. Será que um professor sem este background que temos não ficaria confuso?

São apenas dúvidas...mas a aula ficou muito fácil de ser preparada. Com o editor o prof.poderá modificar o que achar necessário.

Quadro 5.8 – Relato de uma professora universitária

As percepções das professoras são importantes no sentido de tentar aprimorar o programa, o que não pôde ser feito neste projeto devido ao tempo limitado disponível para o mestrado.

Pretendemos continuar desenvolvendo o *software* e tentar levar em conta as necessidades e opiniões dos professores.

O ponto levantado pela professora universitária, que além de utilizar o *software* aplicou a atividade com seus alunos, revela um aspecto do uso de *software* na sala de aula de ensino de línguas que não pode ser subestimado: o impacto que o *software* causa, por ser novidade. Não podemos esquecer que utilizar programa de análise de corpus em aula de língua ainda é algo muito raro, e isso causa estranhamento.

Esse estranhamento não ocorre mais com outras tecnologias que povoam o contexto escolar, como o giz, o quadro negro, o retroprojetor e o gravador de áudio. Cremos que à medida que a presença de *software* de análise de corpus se torne mais comum, o estranhamento deva diminuir. Colabora para isso o fato de o computador estar cada vez mais presente na vida das pessoas, o que diminui a aura de novidade que envolve um programa como o nosso.

O segundo ponto levantado pela mesma professora diz respeito à interação dos alunos com os dados apresentados pelo programa, no sentido de que eles passaram a contar e a conferir os resultados da analise automática. Creio que esse seja um desdobramento benéfico, pois o programa não é infalível e precisa ser checado; além disso, à medida que os alunos verificarem que o *output* do programa é confiável (tanto na abrangência quanto na precisão), passarão a ter menos suspeita.

Neste capítulo, foram apresentados os resultados da análise das principais funções desempenhadas pelo *software* e as primeiras impressões de professores em um primeiro contato. O capítulo a seguir, “Considerações Finais”, faz seu fechamento.

CONSIDERAÇÕES FINAIS

Este capítulo retoma, primeiramente, os passos da pesquisa ora relatada. Em seguida, aponta os resultados encontrados para os objetivos propostos. E, finalmente, apresenta ponderações críticas sobre a pesquisa e sugestões para seu aproveitamento em outros momentos.

Esta pesquisa teve como objetivo principal geral o desenvolvimento e avaliação de um *software* para auxiliar o professor de inglês na elaboração de atividades didáticas de leitura com corpora. Especificamente, a pesquisa objetivou o desenvolvimento de uma ferramenta computacional para o professor de inglês do Ensino Médio elaborar atividades de leitura com corpora.

Para tanto, foram levantadas as características e recursos que fariam parte do *software* de acordo com o público-alvo. Em seguida, o *software* foi criado por meio de programação. Por fim, o *software* foi testado, a fim de verificar sua eficiência, avaliando-o de acordo com os objetivos a que se propõem.

A codificação do *software* foi feita na linguagem do Microsoft Visual Basic 6. Os corpora utilizados foram dois: um corpus de treinamento composto por diversos gêneros, e o BNC (*British National Corpus*) como corpus de referência.

O teste do *software* foi realizado por meio da preparação de uma atividade, em que foram registradas informações processadas das principais funções de análise lingüística executadas pelo *software* na preparação de uma atividade. As informações foram contrastadas com informações provenientes de uma análise manual e calculados os valores de precisão (*precision*) e abrangência (*recall*) para cada função executada.

A seguir, são apresentados os resultados encontrados de acordo com os objetivos da pesquisa, os quais podem ser sintetizados da seguinte maneira:

Primeiramente, tivemos que levantar as características da ferramenta de acordo com o público-alvo. Consideramos algumas variáveis importantes: o acesso do professor ao

computador, o grau de conhecimento digital e o fato de que os pressupostos da Lingüística de Corpus ainda sejam desconhecidos pelos professores.

É importante ressaltar que a busca pelas características da ferramenta foi contínua durante a pesquisa. Houve a necessidade de se refazer várias vezes os possíveis componentes, interfaces e modelos de atividades do *software*, que foram apresentados nos seminários de pesquisa, onde recebia sugestões e críticas feitas por colegas, também professores.

O produto final foi um *software* que auxilia o professor na preparação de atividades de leitura com dados de análise de corpora. O *software* construído prepara as atividades de forma semi-automática, por meio de um assistente. Assim, não são necessários conhecimentos avançados em informática ou em Lingüística de Corpus para poder utilizá-lo.

A intenção de criação do *software* não é substituir o livro didático ou desmerecer o seu papel, mas oferecer ao professor de inglês uma opção a mais para expor o aluno à língua em uso, tornando mais rápida e eficiente a preparação de materiais de qualidade para o ensino de leitura em inglês.

Em segundo lugar, partimos para a criação do *software* por meio de programação. Para a codificação do *software* foi necessário descobrir como criar uma série de funções para a manipulação de textos. As principais mencionadas no trabalho são:

- Fazer lista de palavras e concordâncias;
- Extrair palavras-chave de um texto;
- Identificar palavras cognatas;
- Etiquetar palavras de um texto;
- Descobrir a densidade lexical de um texto;
- Definir a dificuldade de um texto.

Para tanto, foi necessário desenvolver habilidades de programação. Uma das dificuldades encontradas foi a falta de materiais de referência para o desenvolvimento de aplicações para análise lingüística na linguagem de programação utilizada.

Terceiro, passamos para o teste das principais funções de análise lingüística desempenhadas pelo *software*. Os resultados indicaram um desempenho satisfatório do funcionamento do *software*.

Também apresentamos as primeiras impressões do uso do *software* por professores. Embora os relatos ainda não sejam representativos, mostram um posicionamento positivo em relação ao *software* construído.

Como futuro encaminhamento, pretende-se a utilização do *software* por professores da escola pública de São Paulo. O objetivo é cada vez mais desenvolver e melhorar as funções executadas pelo *software.*

Os primeiros passos para isso estão sendo dados. Foi criada uma página para a divulgação do *software* na Internet, que teve algumas modificações, como a inclusão de ferramentas de pesquisa para análise de corpora, no sítio www.corpuslg.org/software/rcb. Também se iniciou o planejamento de um curso sobre a ferramenta para professores da escola pública. Inicialmente, trabalhando na Oficina Pedagógica da Diretoria de Ensino da Região Leste 3, auxiliando nos projetos de língua portuguesa e inglesa pelo Programa Bolsa Mestrado, abrimos inscrições pelo sítio da própria diretoria, www.deleste3.com.br, fornecendo um pequeno resumo sobre a ferramenta e um *link* para sua página, para um curso livre sobre o *Reading Class Builder*.

Devido à necessidade de aprovação do curso como um projeto, o curso está sendo oferecido fora do horário de trabalho do professor e sem certificação. Mesmo assim, obtivemos um número de 14 professores inscritos, interessados no curso, em duas semanas de divulgação. A intenção é, posteriormente, apresentá-lo como um projeto à CENP (Coordenadoria de Estudos e Normas Pedagógicas).

Como todo e qualquer trabalho de pesquisa, este também possui algumas limitações. As limitações devem-se ao tempo disponível para realização da pesquisa de mestrado. A primeira delas é que poderia ser realizado um número maior de testes das funções do *software* construído,

especialmente algumas que consideramos ainda necessitar de uma melhor avaliação, se possível com outros instrumentos, como a verificação da dificuldade estrutural do texto, por meio da fórmula de Flesch, e a identificação de possíveis palavras cognatas. A segunda é a pouca oportunidade para verificação do uso do *software* por professores. A terceira está relacionada à possibilidade de investigar como a aprendizagem por meio das atividades preparadas pelo *software* ocorre.

Contudo, a pesquisa pretende ter contribuído para a Lingüística de Corpus, mais especificamente para a área de Lingüística de Corpus e Ensino, ao mostrar resultados que indicaram o funcionamento efetivo de um *software* para preparação semi-automática de aulas com corpora. Até o momento, não havia estudos do tipo nesta área e, portanto, a pesquisa espera ter feita uma contribuição original para a área.

Espera-se que a ferramenta desenvolvida nesta pesquisa possa ser realmente útil não só para os professores do Ensino Médio, público-alvo inicial, mas também para professores de diversos contextos educacionais.

Referências bibliográficas

Alonso, M. C. G. (2006). Corpus lingüístico e a aquisição de falsos cognatos em espanhol como língua estrangeira. Dissertação de Mestrado. LAEL, PUC, São Paulo.

Amarante, R. P. (2005). Uma contribuição da lingüística de corpus para a fonologia: Um estudo de colocações e aspectos segmentais das vogais da língua inglesa. Dissertação de Mestrado. LAEL, PUC, São Paulo.

Barbosa, M. E. de C. (2004). Material didático para ensino de inglês instrumental on-line: uma abordagem experiencial baseada em corpus, gênero e tarefa. Dissertação de Mestrado. LAEL, PUC, São Paulo.

Beeferman, D., Berger, A., & Lafferty, J. (1997). Text segmentation using exponential models. In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing.

Berber Sardinha, T. (2004). Lingüística de corpus. São Paulo: Manole.

Berber Sardinha, T. (2006a). The book is on the table. Manuscrito inédito.

Berber Sardinha, T. (2006b). Chunk Counter, disponível em http://www2.lael.pucsp.br/corpora/Chunk Counter /index.html.

Bértoli Dutra, P. (2002). Explorando a lingüística de corpus e letras de música na produção de atividades pedagógicas. Dissertação de Mestrado. LAEL, PUC, São Paulo.

Biber, D., S. Conrad & R. Reppen. (1998). *Corpus linguistics* (Investigating Language Structure and Use). Cambridge: Cambridge University Press.

BRASIL, (2002). Ministério da Educação. Secretaria de Educação Média e Tecnológica. PCN + Ensino Médio: Orientações Educacionais complementares aos parâmetros curriculares nacionais. Ciências da natureza, matemática e suas tecnologias. Brasília: MEC, SEMTEC.

Souza, R. C. (2005). Dois corpora, uma tarefa. O percurso de coleta, análise e utilização de corpora eletrônicos na elaboração de uma tarefa para ensino de inglês como Língua Estrangeira. Dissertação de Mestrado. LAEL, PUC, São Paulo.

Ferrari, J. (2004). ESP, Lingüística de corpus e sócio-interacionismo na elaboração de uma unidade de material didático para comércio exterior. Dissertação de Mestrado. LAEL, PUC, São Paulo.

Flowerdew, J. (1993). Concordancing as a tool in course design. System, 21(2).231-244.

Fox, G. (1998). “Using corpus data in the classroom”. In: Tomlinson, B. (org.) *Materials development in language teaching*. Cambridge, Cambridge University Press. P25-43.

Fulcher, G. (1997). Text difficulty and accessibility: Reading formulae and expert judgement. System, 25(4), 497-513.

Gilmore, A. (2004). A comparison of textbooks and authentic interactions. ELT Journal Volume 58/4 October, 1-12.

Guariento W., Morley J. (2001). Text and task authenticity in the EFL classroom. ELT Journal Volume 55/4 October ,1-7

Grellet, F. (1981). *Developing Reading Skills*. Cambridge, U.K.: Cambridge University Press.

Hoey, M. (1993). “Introduction”. In: HEY, M. (org.). *Data, description, discourse: papers on the English language in honour of John McH. Sinclair on his sixtieth birthday*. Londres, Harper-Collins, 1993.

Holmes, J. (1998). Doubt and certainty in ESL textbooks, Applied Linguistics, 9, 21-44.

Hunston, S. (2002). *Corpora in Applied Linguistics*. Cambridge : Cambridge University Press.

Jacobi, C. C. B. D. (2001). Lingüística de Corpus e ensino de espanhol a brasileiros: descrição de padrões e preparação de atividades didáticas (decir/hablar; mismo; mientras / em cuanto/ aunque). Dissertação de Mestrado. LAEL, PUC, São Paulo.

Johns, T. (1986). Microconcord: A language learner's research tool. System: 14(2): 151-162.

Johns, T. (1991). From printout to handout: grammar and vocabulary teaching in the context of data-driven learning. In T. Johns & P. King (eds.) *Classroom concordancing*, ELR Journal, vol. 4. Birmingham: Birmingham University Press, 1-16.

Kennedy, G. D. (1998). *An introduction to corpus linguistics*. Nova York, Longman.

Kennedy, G. D. (2000). There is nothing as practical as a good theory. In: Michael Lewis (Ed.) *Teaching collocation (further developments in the lexical approach)* London: LTP. 10-27.

Kennedy, G. D. (1987a). Expressing temporal frequency in academic English, TESOL Quarterly, 21, 69-86.

Kennedy, G. D. (1987b). Quantification and the use of English: a case study of one aspect of the learners' task, Applied Linguistics, 8, 264-86.

Lewis, M. (1993). *The lexical approach: the state of ELT and a way forward*. Hove, LTP.

Lewis, M. (1997). *Implementing the lexical approach: putting theory into practice*. Hove, LPT.

Lewis, M. (2000). There is nothing as practical as a good theory. In: Michael Lewis (Ed.) *Teaching collocation* (further developments in the lexical approach) (cap. 1, 10-27) London: LTP.

Ljung, M. (1990). *A Study of TEFL Vocabulary*, Stockholm: Almqvist and Wiksell.

McEnery, T. & A. Wilson. (1996). *Corpus linguistics*. Edinburgh: Edinburgh University Press.

Mindt, D. (1992). *Zeitbezung im Englischen: eine didaktische Grammatik des englischen Futurs*. Tübingen: Gunter Narr.

Nuttal, C. (1982). *Reading Skills in a Foreign Language. Cambridge* : Cambridge University Press.

Scott, M., Carioni, L., Zanaffa, M., Bayer, E., & Quintanilha, T. (1984). Using a 'standard exercise' teaching reading comprehension. ELT Journal. 38, 114-20.

Sinclair, J. (1991). *Corpus, concordance, collocation*. Oxford: Oxford University Press.

Sinclair, J. (1987). Collocation. A progress report. In: Ross Steele / Terry Threadgold (Eds.): *Language Topics*. Essays in honour of Michael Halliday. (Amsterdam/Philadelphia). 2, 319-331

Sommerville, I. (2001). *Software Engineering*. Pearson Education.

Thurstun, J.; Candlin, C. (1998) .Concordancing and the teaching of the vocabulary of Academic English. *English for Specific Purposes*, 17(3): 267-280.

Tribble, C. & G. Jones (1990). *Concordances in the classroom*: a resource book for teachers. London: Longman.

Vicentini, G. P. M. (2006). A Lingüística de Corpus e o seriado Friends como base para o ensino de chunks em sala de aula de Língua Inglesa. Dissertação de mestrado. PUC-SP.

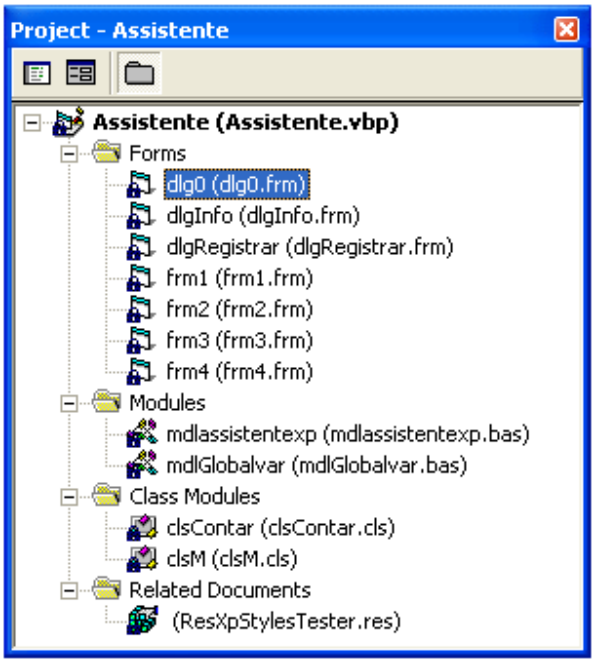
Willis, D. (1990). The Lexical Syllabus – A new approach to language teaching. London/Glasgow: Collins ELT.

Willis, J. (1998). Concordances in the classroom without a computer: assembling and exploiting concordances of common words In: Tomlinson, B. (org.) *Materials development in language teaching*. Cambridge, Cambridge University Press. p25-43.

ANEXOS

A seguir, apresentamos a estrutra do projeto e o código fonte em Visual Basic da primeira versão do *software* e as telas da versão final.

ANEXO 1 – Estrutura do Projeto em Visual Basic



ANEXO 2 – Código do módulo mdlassistentexp

```
Option Explicit

Public Type tagInitCommonControlsEx
   lngSize As Long
   lngICC As Long
End Type
Public Declare Function InitCommonControlsEx Lib "comctl32.dll" (iccex As
tagInitCommonControlsEx) As Boolean
Public Const ICC_USEREX_CLASSES = &H200

Public Sub Main()

   ' we need to call InitCommonControls before we
   ' can use XP visual styles.  Here I'm using
   ' InitCommonControlsEx, which is the extended
   ' version provided in v4.72 upwards (you need
   ' v6.00 or higher to get XP styles)
   On Error Resume Next
   ' this will fail if Comctl not available
   '  - unlikely now though!
   Dim iccex As tagInitCommonControlsEx
   With iccex
       .lngSize = LenB(iccex)
       .lngICC = ICC_USEREX_CLASSES
   End With
   InitCommonControlsEx iccex

   ' now start the application
   On Error GoTo 0
```

```
   frm1.Show

End Sub
```

ANEXO 3 – Código do módulo mdlGlobalvar

```
Global intTipo As Integer


'OBJETO TEXTO
Global strCaminho As String
Global strTexto As String
Global lngSilabas As Long
Global lngPalavras As Long
Global lngFrases As Long
Global lngParagrafos As Long
Global lngTitulos As Long
Global lngFormas As Long
Global lngFlesh As Long
Global strDificuldade As String
Global dblRazaofi As Double
Global dblDensidadelexical As Double
Global lngNgramaticais As Long
Global lngNlexicais As Long
Global dblNcognatos As Double
'Lista
Global dicLista As New Dictionary
'Caminhos
Global Biblio() As String


'AFIXOS
Global strAfixos As String
'TIPOS DE AULA
Global nEx1 As Integer
Global nEx2 As Integer
Global nEx3 As Integer
Global nEx4 As Integer
Global nEx5 As Integer
Global nEx6 As Integer
Global nEx7 As Integer

'VISUALIZAR
Global varCor As Variant
Global varFlag As Variant
'QUAL MODELO UTILIZAR
Global strModelo As String
Global strUnidade As String
Global strTitulo As String
```

ANEXO 4 – Código da classe clsContar

```
' VB5 -> msvbvm50.dll
Private Declare Function VarPtrArray& Lib "msvbvm60.dll" Alias "VarPtr" (ptr() As Any)
Private Declare Sub RtlMoveMemory Lib "kernel32" (dst As Any, src As Any, ByVal
nBytes&)
Private Declare Sub RtlZeroMemory Lib "kernel32" (dst As Any, ByVal nBytes&)
Private Type SAFEARRAY1D
```

```
    cDims           As Integer
    fFeatures       As Integer
    cbElements      As Long
    cLocks          As Long
    pvData          As Long
    cElements       As Long
    lLbound         As Long
End Type
Public y As Long


Public Function Contar_Palavras&(Expression$, D As Dictionary, _
                                 Optional IncludeEmpty As Boolean)
Dim i As Variant
Dim j As Long

 Const Delimiters = " "
    DoEvents
    Const ARR_CHUNK& = 1024

    Dim cExp&, ubExpr&
    Dim cDel&, ubDelim&
    Dim aExpr%(), aDelim%()
    Dim sa1 As SAFEARRAY1D, sa2 As SAFEARRAY1D
    Dim cTokens&, iPos&

    ubExpr = Len(Expression$)
    ubDelim = Len(Delimiters)

    sa1.cbElements = 2:     sa1.cElements = ubExpr
    sa1.cDims = 1:          sa1.pvData = StrPtr(Expression$)
    RtlMoveMemory ByVal VarPtrArray(aExpr), VarPtr(sa1), 4

    sa2.cbElements = 2:     sa2.cElements = ubDelim
    sa2.cDims = 1:          sa2.pvData = StrPtr(Delimiters)
    RtlMoveMemory ByVal VarPtrArray(aDelim), VarPtr(sa2), 4

    'If IncludeEmpty Then
     '   ReDim Preserve resulttokens(ubExpr)
    'Else
        'ReDim Preserve resulttokens(ubExpr \ 2)
    'End If

    ubDelim = ubDelim - 1
    For cExp = 0 To ubExpr - 1
        For cDel = 0 To ubDelim
            If aExpr(cExp) = aDelim(cDel) Then
                If cExp > iPos Then
                  Contar_ocorrencias Mid$(Expression$, iPos + 1, cExp - iPos), D
                    cTokens = cTokens + 1
                ElseIf IncludeEmpty Then
                    Contar_ocorrencias vbNullString, D
                    cTokens = cTokens + 1
                End If
                iPos = cExp + 1
                Exit For
            End If
        Next cDel
    Next cExp

    '/ remainder
    If (cExp > iPos) Or IncludeEmpty Then
       Contar_ocorrencias Mid$(Expression$, iPos + 1), D
```

```
        cTokens = cTokens + 1
    End If


    '/ return ubound
    Contar_Palavras = cTokens - 1

    '/ tidy up
    RtlZeroMemory ByVal VarPtrArray(aExpr), 4
    RtlZeroMemory ByVal VarPtrArray(aDelim), 4
DoEvents
End Function

Private Sub Contar_ocorrencias(strPalavra As String, dicLista As Dictionary)
DoEvents
If dicLista.Exists(strPalavra$) = True Then
DoEvents
dicLista.Item(strPalavra$) = dicLista.Item(strPalavra$) + 1
y = y + 1
Else
DoEvents
dicLista.Add strPalavra$, 1
y = y + 1
End If
DoEvents
End Sub



Public Function Contar_paragrafos(Texto As String) As Long
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim f As Long

r.Global = True

r.Pattern = "\." & vbNewLine & "|" & "\?" & vbNewLine & "|" & "!" & vbNewLine

Set c = r.Execute(Texto)
For Each M In c

f = f + 1

Next
Contar_paragrafos = f
End Function

Public Function Contar_frases(Texto As String) As Long
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim f As Long

r.Global = True

r.Pattern = "[\.;\?!]\b|[\.;\?!]" & vbNewLine
Set c = r.Execute(Texto)
For Each M In c

f = f + 1
```

```
Next
Contar_frases = f
End Function

Public Function Contar_silabas(Texto As String) As Long
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim f As Long

r.IgnoreCase = True
r.Global = True

r.Pattern = " a | e | i | o | u | y " & _
            "|qua|que|qui|quo" & _
            "|gua|gue|gui|guo" & _
            "|b[aeiouAEIOUyY]+" & _
            "|c[aeiouAEIOUyY]+" & _
            "|d[aeiouAEIOUyY]+" & _
            "|f[aeiouAEIOUyY]+" & _
            "|g[aeiouAEIOUyY]+" & _
            "|h[aeiouAEIOUyY]+" & _
            "|j[aeiouAEIOUyY]+" & _
            "|k[aeiouAEIOUyY]+" & _
            "|l[aeiouAEIOUyY]+" & _
            "|m[aeiouAEIOUyY]+" & _
            "|n[aeiouAEIOUyY]+" & _
            "|p[aeiouAEIOUyY]+" & _
            "|q[aeiouAEIOUyY]+" & _
            "|r[aeiouAEIOUyY]+" & _
            "|s[aeiouAEIOUyY]+" & _
            "|t[aeiouAEIOUyY]+" & _
            "|v[aeiouAEIOUyY]+" & _
            "|x[aeiouAEIOUyY]+" & _
            "|w[aeiouAEIOUyY]+" & _
            "|y[aeiouAEIOUyY]+" & _
            "|z[aeiouAEIOUyY]+" & _
            "|\ba\w|\be\w|\bi\w|\bo\w|\bu\w|\wa\b|\we\b|\wi\b|\wo\b|\wu\b" &
"|\ba\w+|\be\w+|\bi\w+|\bo\w+|\bu\w+"



Set c = r.Execute(Texto)
For Each M In c

f = f + 1

Next
Contar_silabas = f
End Function
Public Function Contar_titulos(Texto As String) As Long
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim f As Long

r.Global = True
r.Pattern = " " & vbNewLine & "|\w" & vbNewLine


Set c = r.Execute(Texto)
For Each M In c
```

```
f = f + 1

Next

End Function


Public Function Limpar_texto(Texto As String) As String
Dim r As New RegExp

r.Pattern = "\||\\|<|,|>|\.|\.\.\.|:|;|\?|/|\^|~|\}|\]|\{|\[|\+|=|_|\)" & _
            "|\*|\&|¨|%|\$|#|@|\(|\!|[0-9]|´|`|\s+|" & Chr(34) & _
            "|" & vbCr & "|\s+|--+|''+|''|\W-+|\b-\b|- " & vbNewLine & _
            "|' " & vbNewLine
r.Global = True
r.IgnoreCase = True
Texto = Replace(Replace(Texto, "...", " "), Chr(34), " ")
Texto = UCase(r.Replace(Texto, " " & vbNewLine))
Limpar_texto = Trim(r.Replace(Texto, " "))

End Function
Public Function Flesh_formula(nfrases As Long, nsilabas As Long, nItens As Long)
Dim x As Long
Dim y As Long
Dim t As Long



If nfrases = 0 Then nfrases = 1

x = (1.015 * (nItens / nfrases))
y = (0.846 * (nsilabas / nItens))
t = (x + y) + 100
Flesh_formula = 206.835 - t



End Function
Public Function Definir_dificuldade(flesch As Long) As String
Dim nivel As String

If Format(flesch, "00") = 90 Then nivel = "muito fácil"
If Format(flesch, "00") = 91 Then nivel = "muito fácil"
If Format(flesch, "00") = 92 Then nivel = "muito fácil"
If Format(flesch, "00") = 93 Then nivel = "muito fácil"
If Format(flesch, "00") = 94 Then nivel = "muito fácil"
If Format(flesch, "00") = 95 Then nivel = "muito fácil"
If Format(flesch, "00") = 96 Then nivel = "muito fácil"
If Format(flesch, "00") = 97 Then nivel = "muito fácil"
If Format(flesch, "00") = 98 Then nivel = "muito fácil"
If Format(flesch, "00") = 99 Then nivel = "muito fácil"
If Format(flesch, "00") = 100 Then nivel = "muito fácil"
If Format(flesch, "00") = 80 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 81 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 82 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 83 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 84 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 85 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 86 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 87 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 88 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 89 Then nivel = "razoavelmente fácil"
```

```
If Format(flesch, "00") = 70 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 71 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 72 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 73 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 74 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 75 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 76 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 77 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 78 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 79 Then nivel = "razoavelmente fácil"
If Format(flesch, "00") = 60 Then nivel = "padrão"
If Format(flesch, "00") = 61 Then nivel = "padrão"
If Format(flesch, "00") = 62 Then nivel = "padrão"
If Format(flesch, "00") = 63 Then nivel = "padrão"
If Format(flesch, "00") = 64 Then nivel = "padrão"
If Format(flesch, "00") = 65 Then nivel = "padrão"
If Format(flesch, "00") = 66 Then nivel = "padrão"
If Format(flesch, "00") = 67 Then nivel = "padrão"
If Format(flesch, "00") = 68 Then nivel = "padrão"
If Format(flesch, "00") = 69 Then nivel = "padrão"
If Format(flesch, "00") = 50 Then nivel = "razoavelmente difícil"
If Format(flesch, "00") = 51 Then nivel = "razoavelmente difícil"
If Format(flesch, "00") = 52 Then nivel = "razoavelmente difícil"
If Format(flesch, "00") = 53 Then nivel = "razoavelmente difícil"
If Format(flesch, "00") = 54 Then nivel = "razoavelmente difícil"
If Format(flesch, "00") = 55 Then nivel = "razoavelmente difícil"
If Format(flesch, "00") = 56 Then nivel = "razoavelmente difícil"
If Format(flesch, "00") = 57 Then nivel = "razoavelmente difícil"
If Format(flesch, "00") = 58 Then nivel = "razoavelmente difícil"
If Format(flesch, "00") = 59 Then nivel = "razoavelmente difícil"
If Format(flesch, "00") = 30 Then nivel = "difícil"
If Format(flesch, "00") = 31 Then nivel = "difícil"
If Format(flesch, "00") = 32 Then nivel = "difícil"
If Format(flesch, "00") = 33 Then nivel = "difícil"
If Format(flesch, "00") = 34 Then nivel = "difícil"
If Format(flesch, "00") = 35 Then nivel = "difícil"
If Format(flesch, "00") = 36 Then nivel = "difícil"
If Format(flesch, "00") = 37 Then nivel = "difícil"
If Format(flesch, "00") = 38 Then nivel = "difícil"
If Format(flesch, "00") = 39 Then nivel = "difícil"
If Format(flesch, "00") = 40 Then nivel = "difícil"
If Format(flesch, "00") = 41 Then nivel = "difícil"
If Format(flesch, "00") = 42 Then nivel = "difícil"
If Format(flesch, "00") = 43 Then nivel = "difícil"
If Format(flesch, "00") = 44 Then nivel = "difícil"
If Format(flesch, "00") = 45 Then nivel = "difícil"
If Format(flesch, "00") = 46 Then nivel = "difícil"
If Format(flesch, "00") = 47 Then nivel = "difícil"
If Format(flesch, "00") = 48 Then nivel = "difícil"
If Format(flesch, "00") = 49 Then nivel = "difícil"
If Format(flesch, "00") = 0 Then nivel = "muito difícil"
If Format(flesch, "00") = 1 Then nivel = "muito difícil"
If Format(flesch, "00") = 2 Then nivel = "muito difícil"
If Format(flesch, "00") = 3 Then nivel = "muito difícil"
If Format(flesch, "00") = 4 Then nivel = "muito difícil"
If Format(flesch, "00") = 5 Then nivel = "muito difícil"
If Format(flesch, "00") = 6 Then nivel = "muito difícil"
If Format(flesch, "00") = 7 Then nivel = "muito difícil"
If Format(flesch, "00") = 8 Then nivel = "muito difícil"
If Format(flesch, "00") = 9 Then nivel = "muito difícil"
If Format(flesch, "00") = 10 Then nivel = "muito difícil"
If Format(flesch, "00") = 11 Then nivel = "muito difícil"
If Format(flesch, "00") = 12 Then nivel = "muito difícil"
```

```
If Format(flesch, "00") = 13 Then nivel = "muito difícil"
If Format(flesch, "00") = 14 Then nivel = "muito difícil"
If Format(flesch, "00") = 15 Then nivel = "muito difícil"
If Format(flesch, "00") = 16 Then nivel = "muito difícil"
If Format(flesch, "00") = 17 Then nivel = "muito difícil"
If Format(flesch, "00") = 18 Then nivel = "muito difícil"
If Format(flesch, "00") = 19 Then nivel = "muito difícil"
If Format(flesch, "00") = 20 Then nivel = "muito difícil"
If Format(flesch, "00") = 21 Then nivel = "muito difícil"
If Format(flesch, "00") = 22 Then nivel = "muito difícil"
If Format(flesch, "00") = 23 Then nivel = "muito difícil"
If Format(flesch, "00") = 24 Then nivel = "muito difícil"
If Format(flesch, "00") = 25 Then nivel = "muito difícil"
If Format(flesch, "00") = 26 Then nivel = "muito difícil"
If Format(flesch, "00") = 27 Then nivel = "muito difícil"
If Format(flesch, "00") = 28 Then nivel = "muito difícil"
If Format(flesch, "00") = 29 Then nivel = "muito difícil"
If flesch <= 0 Then nivel = "muito difícil"
If flesch >= 100 Then nivel = "muito fácil"
Definir_dificuldade = nivel
End Function
```

ANEXO 5 – Código da classe clsM

```
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (pDest As Any,
pSource As Any, ByVal dwLength As Long)
Private Declare Function LocalAlloc Lib "kernel32" (ByVal uFlags As Long, ByVal uBytes
As Long) As Long
Private Declare Function LocalFree Lib "kernel32" (ByVal hMem As Long) As Long
Private Const LPTR = (&H0 Or &H40)
Private StrLen As Long
Private Buffer As String
Public BufferSize As Long
Public Registros As Long


Public Property Get Value() As String

   Value = Left$(Buffer, StrLen)

End Property

Public Property Let Value(s As String)

   Buffer = ""
   StrLen = 0
   Append s

End Property

Public Property Get Length() As Long

   Length = StrLen

End Property


Public Sub Append(s As String)
   Dim l As Long

   l = StrLen + Len(s)
```

```
   While l > Len(Buffer)
      Buffer = Buffer & Space$(BufferSize)
   Wend

   CopyMemory ByVal StrPtr(Buffer) + (StrLen * 2), ByVal StrPtr(s), LenB(s)
   StrLen = StrLen + Len(s)

End Sub

Public Sub AppendLine(s As String)

   Append s
   Append vbCrLf

End Sub

Public Sub Prepend(s As String)

   Dim v As String
   Dim l As Long

   l = StrLen + Len(s)

   While l > Len(Buffer)
      Buffer = Buffer & Space$(BufferSize)
   Wend

   v = Value

   CopyMemory ByVal StrPtr(Buffer) + LenB(s), ByVal StrPtr(v), LenB(v)
   CopyMemory ByVal StrPtr(Buffer), ByVal StrPtr(s), LenB(s)
   StrLen = StrLen + Len(s)

End Sub

Public Function FromLeft(Length As Long) As String

   If StrLen > Length Then
      FromLeft = Left$(Buffer, Length)
   Else
      FromLeft = Left$(Buffer, StrLen)
   End If

End Function

Public Function FromRight(Length As Long) As String

   If Length > StrLen Then
      FromRight = Mid$(Buffer, 1, StrLen)
   Else
      FromRight = Mid$(Buffer, (StrLen - Length) + 1, Length)
   End If

End Function

Public Function CharacterAt(Pos As Long) As String

   If Pos < StrLen + 1 And Pos > 0 Then
      CharacterAt = Mid$(Buffer, Pos, 1)
   End If

End Function
```

```
Public Sub UpperCase()

   Buffer = StrConv(Buffer, vbUpperCase)

End Sub

Public Sub LowerCase()

   Buffer = StrConv(Buffer, vbLowerCase)

End Sub

Private Sub Class_Initialize()

   BufferSize = 100000
   StrLen = 0

End Sub

Public Function malloc(Strin As String) As Long
    Dim PointerA As Long, lSize As Long

    lSize = LenB(Strin) 'Length of string in bytes.

    'Allocate the memory needed and returns a pointer to that memory
    PointerA = LocalAlloc(LPTR, lSize + 4)
    If PointerA <> 0 Then
        'Final allocation
        CopyMemory ByVal PointerA, lSize, 4
        If lSize > 0 Then
            'copy the string to that allocated memory.
            CopyMemory ByVal PointerA + 4, ByVal StrPtr(Strin), lSize
        End If
    End If
    'return the pointer to the string stored memory
    malloc = PointerA
End Function

Public Function RetMemory(PointerA As Long) As String
    Dim lSize As Long, sThis As String
    If PointerA = 0 Then
        GetMemory = ""
    Else
        'get the size of the string stored at pointer "PointerA"
        CopyMemory lSize, ByVal PointerA, 4
        If lSize > 0 Then
            'buffer a varible
            sThis = String(lSize \ 2, 0)
            'retrive the data at the address of "PointerA"
            CopyMemory ByVal StrPtr(sThis), ByVal PointerA + 4, lSize
            'return the buffer
            RetMemory = sThis
        End If
    End If
End Function

Public Sub FreeMemory(PointerA As Long)
    'frees up the memory at the address of "PointerA"
    LocalFree PointerA
End Sub
```

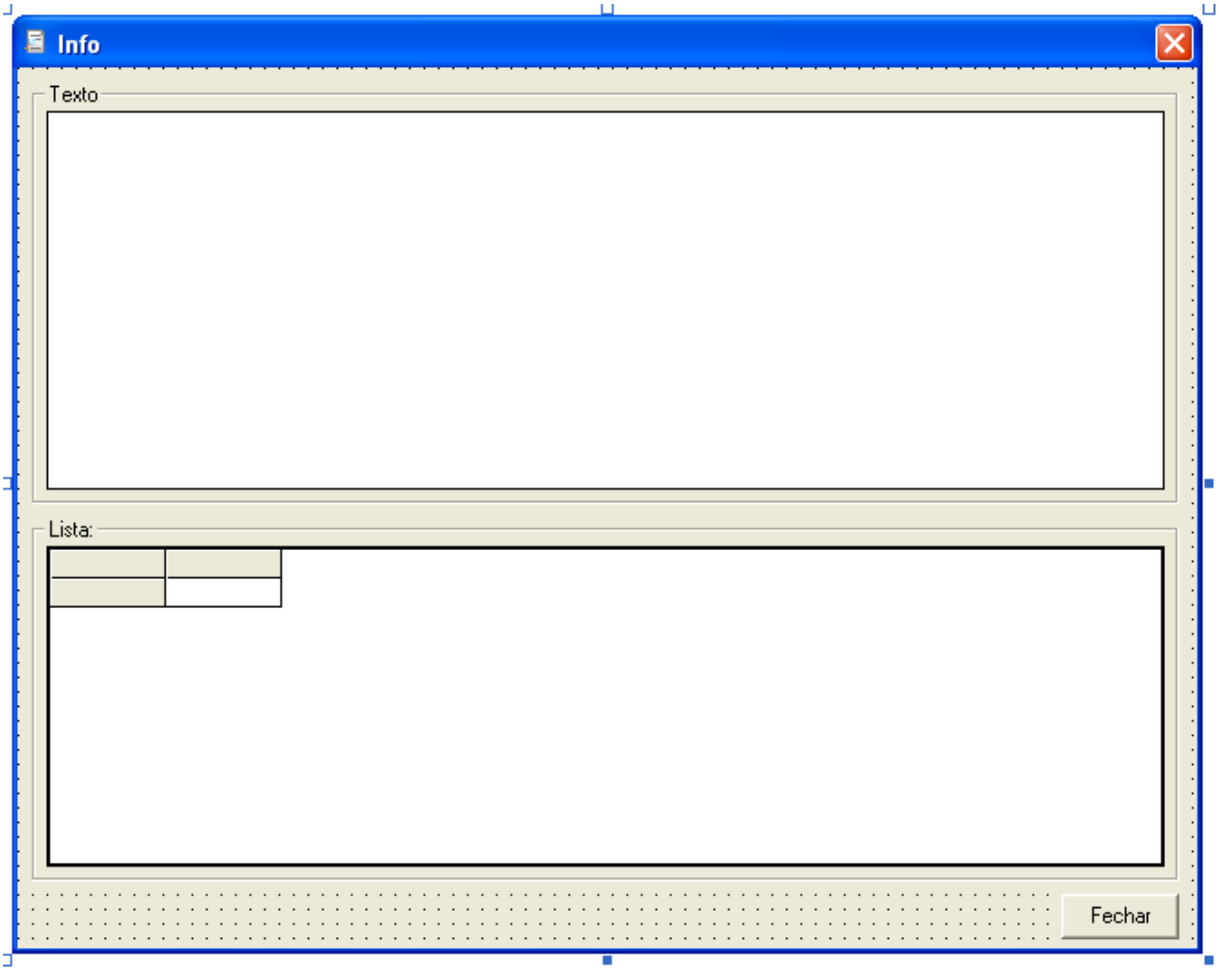
ANEXO 6 – Formulário dlg0

ANEXO 7 – Código do formulário dlg0

```
Option Explicit
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal
lpClassName As Any, ByVal lpWindowName As Any) As Long
   Private Declare Function PostMessage Lib "user32" Alias "PostMessageA" (ByVal hwnd
As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
   Private Const WM_QUIT = &H12

Private Sub cmdEditar_Click()

Fechar_utilitario "Nova Aula - Editor de Aulas"
frmFinal.rtb1.SaveFile (App.Path & "\aula.als")
Shell App.Path & "\Editor.exe", vbNormalFocus
Unload frmFinal
Unload dlg0
Set frmFinal = Nothing
Set dlg0 = Nothing

End Sub

Private Sub cmdFechar_Click()
Unload frmFinal
Unload dlg0
Set frmFinal = Nothing
Set dlg0 = Nothing

End Sub

Private Sub cmdImprimir_Click()
Imprimir
End Sub

Private Sub cmdSalvar_Click()
frmFinal.Salvar
End Sub

Private Sub Form_Load()
dlg0.Top = dlg0.Top + frmFinal.Top / 2
dlg0.Left = ((frmFinal.Left + frmFinal.Width) - dlg0.Width) - 200
End Sub

Private Sub Form_Unload(Cancel As Integer)
Unload dlg0

End Sub
Public Function Fechar_utilitario(strTitulo As String)
Dim sTitle As String
     Dim iHwnd As Long
     Dim ihTask As Long
     Dim iReturn As Long
```

```
      sTitle = strTitulo
      iHwnd = FindWindow(0&, sTitle)
      iReturn = PostMessage(iHwnd, WM_QUIT, 0&, 0&)

End Function

Public Function Imprimir()


frmFinal.cdg1.CancelError = True
On Error GoTo tratarerro
frmFinal.cdg1.ShowPrinter
Printer.Copies = frmFinal.cdg1.Copies
frmFinal.rtb1.SelPrint Printer.hDC

tratarerro:
Exit Function
End Function
```

ANEXO 8 – Formulário dlgInfo

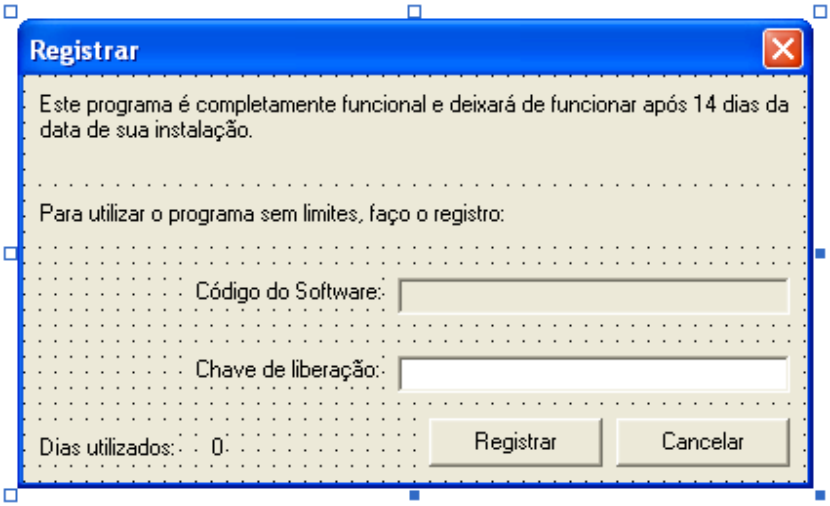


ANEXO 9 – Código do formulário dlgInfo

```
Option Explicit
Dim strAnterior As String

Private Sub cmdFechar_Click()
Unload dlgInfo
Set dlgInfo = Nothing
End Sub

Private Sub List1_Click()
Desmarcar vbBlue, True
Destacar msf1.TextMatrix(msf1.Row, 1)
End Sub

Public Function Destacar(strPalavra)
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection

r.Pattern = "\b" & strPalavra & "\b" & "|" & "\b" & strPalavra & ",\b" & "|" & "\b" &
strPalavra & "\.\b"
r.IgnoreCase = True
r.Global = True

strAnterior = "\b" & strPalavra & "\b" & "|" & "\b" & strPalavra & ",\b" & "|" & "\b"
& strPalavra & "\.\b"

Set c = r.Execute(rtb1.Text)
With rtb1
For Each M In c
.SelStart = M.FirstIndex
        .SelLength = M.Length
        .SelColor = vbRed
        .SelBold = True
Next
End With

End Function
Public Function Desmarcar(cor As Variant, flag As Variant)
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection

r.Pattern = strAnterior
r.IgnoreCase = True
r.Global = True

Set c = r.Execute(rtb1.Text)
With rtb1
For Each M In c
.SelStart = M.FirstIndex
        .SelLength = M.Length
        .SelColor = cor
        .SelBold = flag
Next
End With

End Function

Private Sub Form_Load()

End Sub
```

```
Private Sub msf1_Click()
Desmarcar varCor, varFlag
Destacar msf1.TextMatrix(msf1.Row, 1)
End Sub
```

ANEXO 10 – Formulário dlgRegistrar

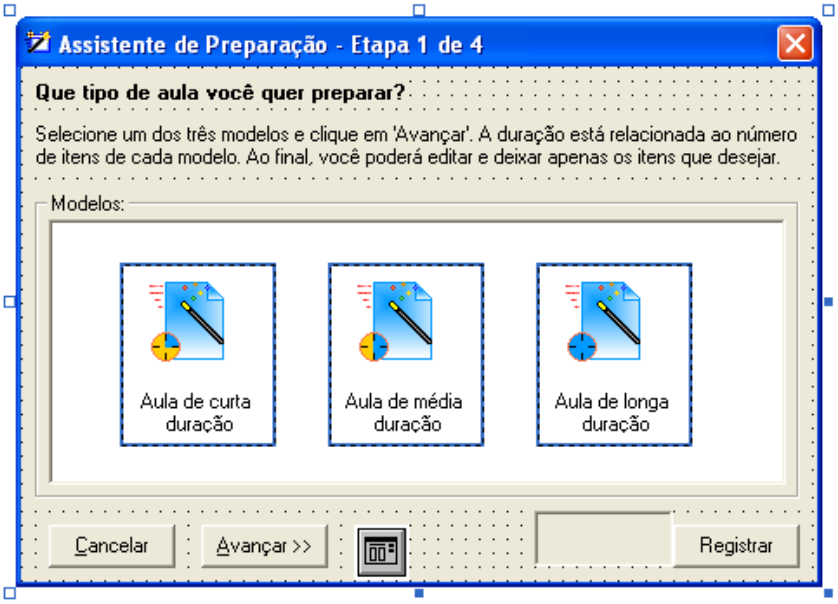


ANEXO 11 – Código do formulário dlgRegistrar

```
Option Explicit

Private Sub cmdCancelar_Click()
Unload dlgRegistrar
frm1.Show
End Sub

Private Sub cmdRegistrar_Click()
If txtcodigo.Text = "" Then
   txtcodigo.SetFocus
   Exit Sub
End If
frm1.alock.LiberationKey = txtcodigo.Text
If Not frm1.alock.RegisteredUser Then
  MsgBox "Chave de LIBERAÇÃO INCORRETA", vbOKOnly + vbCritical, "Chave Liberação
Incorreta"
   txtcodigo.SetFocus
Else
  MsgBox "REGISTRO EFETUADO COM SUCESSO !", vbExclamation, "Registro OK"
  frm1.cmdRegistrar.Visible = False
  'frmMenu.Caption = "VERSÃO REGISTRADA"
  'frmMenu.lblregistro(2).Enabled = False
  Unload dlgRegistrar
  frm1.Show
End If
```

```
End Sub

Private Sub Form_Load()
If frm1.alock.LastRunDate > Now Then

    MsgBox "Ocorreu uma alteração na data do sistema operacional " _
    & vbCrLf & " O programa será encerrado.", vbOKOnly + vbCritical, "Erro de sistema"
    End
End If

 txtcodigo.Text = frm1.alock.SoftwareCode
 lbldias.Caption = frm1.alock.UsedDays

End Sub
```

ANEXO 12 – Formulário frm1



ANEXO 13 – Código do formulário frm1

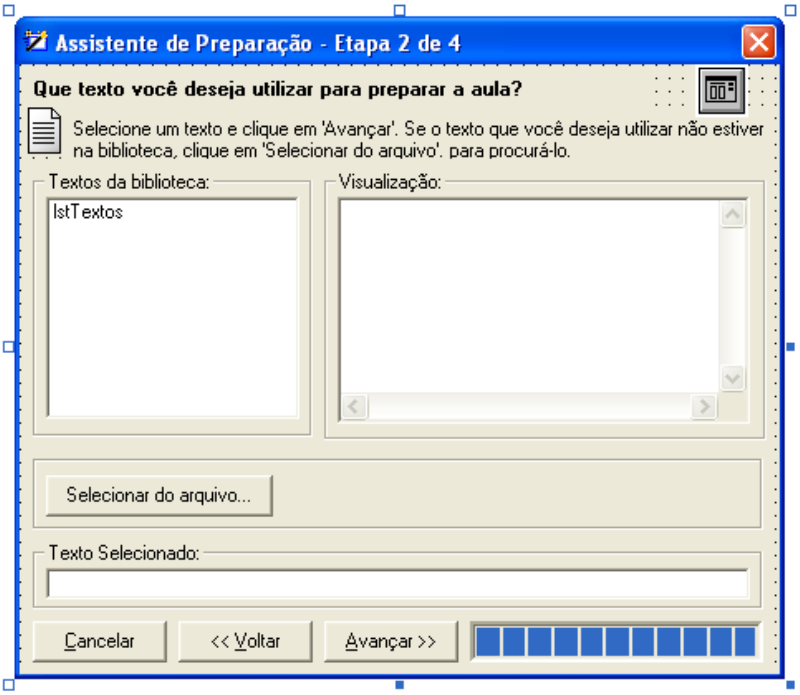
```
Private Declare Function InitCommonControls Lib "comctl32.dll" () As Long

Private Sub cmdRegistrar_Click()
Load dlgRegistrar
dlgRegistrar.Show 0, frm1
End Sub

Private Sub Form_Initialize()
Call InitCommonControls
Ajuda

End Sub
Private Sub cmdAvancar_Click()
Load frm2
frm1.Hide
frm2.Show
Definir_tipo_de_aula
```

```
End Sub

Private Sub cmdCancelar_Click()
Unload frm1
Set frm1 = Nothing
End Sub

Public Function Tirar_selecao()
Dim i As Long
For i = 0 To shp1.Count - 1
shp1(i).Visible = False
Next i
cmdAvancar.Enabled = False
End Function

Private Sub Form_Load()
Tirar_selecao
App.HelpFile = App.Path & "\ajuda.HLP"


If alock.RegisteredUser = True Then
cmdRegistrar.Visible = False
cmdAvancar.Visible = True
Exit Sub
Else

If alock.UsedDays < 14 Then
cmdAvancar.Visible = True
Else
cmdAvancar.Visible = False
MsgBox "Você já usou esta versão Demo " & alock.Counter & "dias." & vbNewLine _
& "Faça o registro!"
End If
End If

If alock.LastRunDate > Now Then

    MsgBox "Ocorreu uma alteração na data do sistema operacional " _
    & vbCrLf & " O programa será encerrado.", vbOKOnly + vbCritical, "Erro de sistema"
    End
End If

End Sub

Private Sub Form_Unload(Cancel As Integer)
Unload frm1
End Sub

Private Sub img1_Click(index As Integer)
Selecionar index
End Sub

Private Sub lbl1_Click(index As Integer)
Selecionar index
End Sub

Private Sub Picture1_Click()
Tirar_selecao
End Sub

Private Sub tipo_Click(index As Integer)
Selecionar index
End Sub
```

```
Public Function Selecionar(index As Integer)
Tirar_selecao
shp1(index).Visible = True
intTipo = index
cmdAvancar.Enabled = True

End Function

Public Function Definir_tipo_de_aula()

Dim i As Long
Dim p() As String
Dim conf As String
Dim x As Integer

Select Case intTipo
Case 0

Open App.Path & "\crt.ini" For Input As #1
conf = Input(LOF(1), #1)
Close #1

p = Split(conf, vbNewLine)
For i = LBound(p) To UBound(p)
x = x + 1
If x = 8 Then Exit For
Next i



nEx1 = CInt(p(0))
nEx2 = CInt(p(1))
nEx3 = CInt(p(2))
nEx4 = CInt(p(3))
nEx5 = CInt(p(4))
nEx6 = CInt(p(5))
nEx7 = CInt(p(6))

Case 1
Open App.Path & "\mda.ini" For Input As #1
conf = Input(LOF(1), #1)
Close #1

p = Split(conf, vbNewLine)
For i = LBound(p) To UBound(p)
x = x + 1
If x = 8 Then Exit For
Next i



nEx1 = CInt(p(0))
nEx2 = CInt(p(1))
nEx3 = CInt(p(2))
nEx4 = CInt(p(3))
nEx5 = CInt(p(4))
nEx6 = CInt(p(5))
nEx7 = CInt(p(6))

Case 2
Open App.Path & "\lng.ini" For Input As #1
conf = Input(LOF(1), #1)
```

```
Close #1

p = Split(conf, vbNewLine)
For i = LBound(p) To UBound(p)
x = x + 1
If x = 8 Then Exit For
Next i



nEx1 = CInt(p(0))
nEx2 = CInt(p(1))
nEx3 = CInt(p(2))
nEx4 = CInt(p(3))
nEx5 = CInt(p(4))
nEx6 = CInt(p(5))
nEx7 = CInt(p(6))
Case Else
End Select



End Function


Public Function Ajuda()
On Error GoTo tratarerro
cdg1.CancelError = True

With cdg1
        .HelpFile = App.HelpFile

        If IndexID = 0 Then
            .HelpCommand = cdlHelpContents
        Else
            .HelpContext = IndexID
            .HelpCommand = cdlHelpContext
        End If

        .ShowHelp
End With



Exit Function
tratarerro:

End Function
```

ANEXO 14 – Formulário frm2



ANEXO 15 – Código do Formulário frm2

```
Private Sub cmdAvancar_Click()
pgb1.Visible = True
FileCopy App.Path & "\objTexto.mtz", App.Path & "\objtexto1.mtz"
Abrir_Texto
Contar
Load frm3
frm2.Hide
frm3.Show
End Sub

Private Sub cmdCancelar_Click()
Unload frm1
Unload frm2
Set frm1 = Nothing
Set frm2 = Nothing
End Sub

Private Sub CmdSelecionardoArquivo_Click()
On Error GoTo Cancelar
cdg1.Filter = "Somente texto (*.txt)|*.txt|"
cdg1.ShowOpen
txtTexto.Text = cdg1.FileName
Visualizar_texto cdg1.FileName
strCaminho = cdg1.FileName
Exit Sub
Cancelar:
cdg1.CancelError = True
End Sub

Public Function Visualizar_texto(caminho As String)
```

```
Open caminho For Input As #1
txtVisualizacao.Text = Input(LOF(1), #1)
Close #1
End Function

Private Sub cmdVoltar_Click()
Unload frm2
frm1.Show
End Sub

Public Function Abrir_Texto()
Open strCaminho For Input As #1
strTexto = Input(LOF(1), #1)
Close #1
End Function

Public Function Contar()
Dim c As New clsContar
Dim i As Variant
With pgb1
.Min = 1
.Max = 9
.Visible = True
.Value = 1
lngSilabas = c.Contar_silabas(strTexto)
lngFrases = c.Contar_frases(strTexto)
lngParagrafos = c.Contar_paragrafos(strTexto)
c.Contar_Palavras c.Limpar_texto(strTexto), dicLista, False
.Value = 2
lngPalavras = c.y
lngFormas = dicLista.Count
dblRazaofi = Format((lngFormas / lngPalavras) * 100, "00.00")
lngFlesh = c.Flesh_formula(lngFrases, lngSilabas, lngPalavras)
strDificuldade = c.Definir_dificuldade(lngFlesh)
.Value = 3
Armazenar_frequencias
.Value = 4
Etiquetar
.Value = 5
Densidade_lexical
.Value = 6
Fazer_palavras_chave
.Value = 7
cognatos
.Value = 8
Armazenar_estatisticas
.Value = 9
.Visible = False
End With

Set c = Nothing


End Function

Public Function Armazenar_estatisticas()
Dim bd As DAO.Database
Dim tb As DAO.Recordset

Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("estatisticas", dbOpenTable)

tb.AddNew
```

```
tb("paragrafos") = lngParagrafos
tb("frases") = lngFrases
tb("silabas") = lngSilabas
tb("itens") = lngPalavras
tb("formas") = lngFormas
tb("razaofi") = dblRazaofi
tb("densidadelexical") = dblDensidadelexical
tb("cognatos") = dblNcognatos
tb("flesh") = lngFlesh
tb("dificuldade") = strDificuldade
tb.Update


tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing
End Function

Public Function Armazenar_frequencias()
Dim bd As DAO.Database
Dim tb As DAO.Recordset

Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("formas", dbOpenTable)
DoEvents
For Each x In dicLista.Keys
DoEvents
tb.AddNew
tb("palavra") = x
tb("frequencia") = dicLista.Item(x)
tb("porcentagem") = Format((dicLista.Item(x) / lngPalavras) * 100, "00.00")
tb.Update
Next

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing

dicLista.RemoveAll
Set dicLista = Nothing
DoEvents
End Function

Public Function Etiquetar()
Dim bd1 As DAO.Database
Dim tb1 As DAO.Recordset
Dim bd2 As DAO.Database
Dim tb2 As DAO.Recordset
Dim M As New clsM
Dim p1 As Long


Set bd1 = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb1 = bd1.OpenRecordset("formas", dbOpenTable)
Set bd2 = DBEngine.OpenDatabase(App.Path & "\etiq.ref")
DoEvents
tb1.MoveFirst
Do Until tb1.EOF
DoEvents

p1 = M.malloc(Replace(tb1("palavra"), "'", "´"))
```

```
Set tb2 = bd2.OpenRecordset("SELECT [tag] FROM [etiquetadas] WHERE [palavra] = " & "'"
& M.RetMemory(p1) & "'" & " ")
M.FreeMemory (p1)
If tb2.RecordCount <> 0 Then
tb1.Edit
tb1("tag") = tb2("tag")
tb1.Update
Else
tb1.Edit
tb1("tag") = "nn1"
tb1.Update
End If
tb1.MoveNext
Loop

tb1.Close
bd1.Close
tb2.Close
bd2.Close

Set tb1 = Nothing
Set bd1 = Nothing
Set tb2 = Nothing
Set bd2 = Nothing
Set M = Nothing
DoEvents
End Function

Public Function Densidade_lexical()
Dim bd1 As DAO.Database
Dim tb1 As DAO.Recordset
Dim palavra As String
Dim r As New RegExp

r.Pattern =
"AJ0|AJC|AJS|NN0|NN1|NN2|NP0|VBB|VBD|VBG|VBI|VBN|VBZ|VDB|VVG|VVD|VVI|VVB|VV"
r.Global = True
r.IgnoreCase = True

Set bd1 = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb1 = bd1.OpenRecordset("formas", dbOpenTable)

DoEvents
tb1.MoveFirst
Do Until tb1.EOF
DoEvents
If r.Test(tb1("tag")) = True Then
lngNlexicais = lngNlexicais + tb1("frequencia")
tb1.Edit
tb1("categoria") = "1"
tb1.Update
Else
lngNgramaticais = lngNgramaticais + tb1("frequencia")
tb1.Edit
tb1("categoria") = "0"
tb1.Update
End If
tb1.MoveNext
Loop

dblDensidadelexical = Format((100 * (lngNlexicais / lngPalavras)), "00.00")

tb1.Close
```

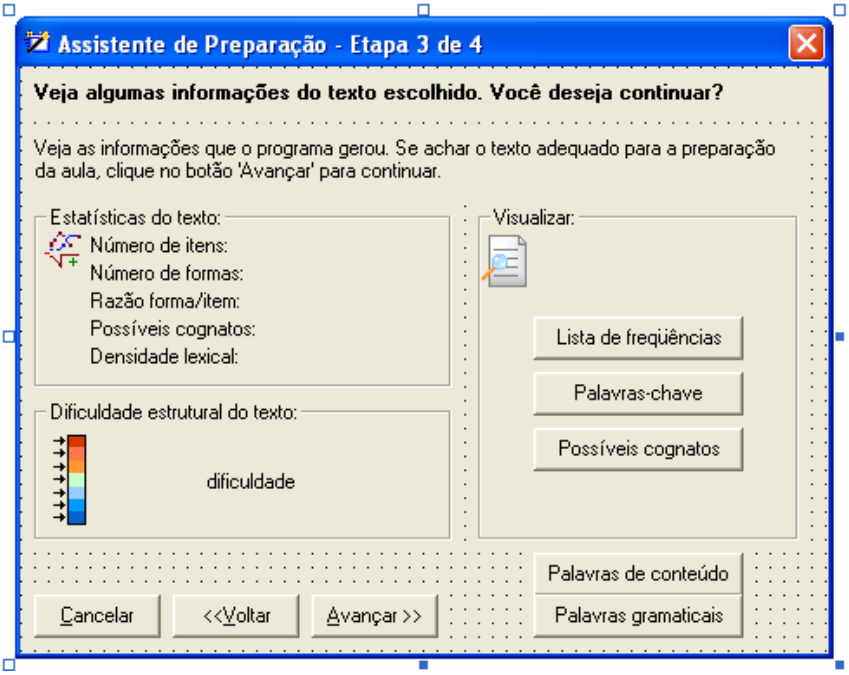
```
bd1.Close
Set tb1 = Nothing
Set bd1 = Nothing
DoEvents
End Function

Public Function Calcular_chavicidade(fpc1 As String, fpc2 As Long, _
ntpc1 As Long, ntpc2 As Long, Significancia As Long) As String

'ntpc2 era string

'Dim a, b, c, d As Long
Dim E1, E2 As Double
Dim LL As Double
'Dim SobreUso As String
'Dim SubUso As String

'SobreUso = "+"
'SubUso = "-"

'a = fpc1
'b = fpc2
'c = ntpc1
'd = ntpc2

E1 = ntpc1 * (fpc1 + fpc2) / (ntpc1 + ntpc2)
E2 = ntpc2 * (fpc1 + fpc2) / (ntpc1 + ntpc2)
LL = 2 * ((fpc1 * Log(fpc1 / E1)) + (fpc2 * Log(fpc2 / E2)))

If LL <= Significancia Then
Calcular_chavicidade = Format(LL, "00.00") & "-" 'SubUso
Else
Calcular_chavicidade = Format(LL, "00.00") & "+" 'SobreUso
End If

End Function
Public Function Fazer_palavras_chave()
Dim bd1 As Database
Dim bd2 As Database
Dim tb1 As Recordset
Dim tb2 As Recordset
Dim p1 As Long
Dim M As New clsM
Dim l As Long
Dim i As Long
Set bd1 = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb1 = bd1.OpenRecordset("SELECT * FROM [formas] WHERE [palavra] > '1'ORDER BY
[palavra] ASC")

DoEvents
tb1.MoveFirst
Do Until tb1.EOF
DoEvents
p1 = M.malloc(Replace(tb1("palavra"), "'", "´"))

Set bd2 = OpenDatabase(App.Path & "\ingl.ref")
Set tb2 = bd2.OpenRecordset("SELECT [palavra],[frequencia] FROM [eng] WHERE [palavra]
=" & "'" & M.RetMemory(p1) & "'" & " ")
M.FreeMemory (p1)
If tb2.RecordCount = 0 Then
tb1.Edit
tb1("chavicidade") = 0
tb1.Update
```

```
Else
tb1.Edit
tb1("chavicidade") = Calcular_chavicidade(tb1("frequencia"), tb2("frequencia"), _
lngPalavras, 102467488, 0.1)
tb1.Update
End If

tb1.MoveNext
'If l = 500 Then Exit Do
Loop


tb1.Close
tb2.Close
bd1.Close
bd2.Close
Set tb1 = Nothing
Set tb2 = Nothing
Set bd1 = Nothing
Set bd2 = Nothing
Set M = Nothing

DoEvents
End Function

Public Function cognatos()
Dim bd1 As DAO.Database
Dim tb1 As DAO.Recordset
Dim bd2 As DAO.Database
Dim tb2 As DAO.Recordset
Dim cog As Long
Dim p1 As Long
Dim M As New clsM


Set bd1 = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb1 = bd1.OpenRecordset("formas", dbOpenTable)

Set bd2 = DBEngine.OpenDatabase(App.Path & "\radicais.ref")

DoEvents
tb1.MoveFirst
Do Until tb1.EOF
DoEvents

p1 = M.malloc(Replace(Mid$(tb1("palavra"), 1, 5), "'", "´"))

Set tb2 = bd2.OpenRecordset("SELECT [radical] FROM  [rad] WHERE [radical] = " & "'" &
M.RetMemory(p1) & "'" & " ")
M.FreeMemory (p1)

If tb2.RecordCount <> 0 Then
tb1.Edit
tb1("cognato") = "1"
tb1.Update
Else
End If

tb1.MoveNext
Loop


    DoEvents
```

```
 tb1.MoveFirst
 Do Until tb1.EOF
 DoEvents
 If tb1("cognato") <> 0 Then
 cog = cog + 1
 Else
 tb1.Edit
 tb1("cognato") = 0
 tb1.Update
 End If
 tb1.MoveNext
 Loop

dblNcognatos = Format((100 * (cog / lngFormas)), "00.00")

tb1.Close
tb2.Close
bd1.Close
bd2.Close

Set tb1 = Nothing
Set tb2 = Nothing
Set bd1 = Nothing
Set bd2 = Nothing
Set M = Nothing
DoEvents
End Function

Private Sub Form_Load()
carregar_da_biblioteca
End Sub

Private Sub Form_Unload(Cancel As Integer)
Unload frm1
Unload frm2
End Sub

Private Sub lstTextos_Click()
Selecionar_da_biblioteca lstTextos.ListIndex
End Sub

Private Sub txtTexto_Change()
cmdAvancar.Enabled = True
End Sub

Public Function carregar_da_biblioteca()
Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long

Set bd = DBEngine.OpenDatabase(App.Path & "\bibliotxts.mdb")
Set tb = bd.OpenRecordset("textos", dbOpenTable)
On Error Resume Next
tb.MoveFirst
Do Until tb.EOF

lstTextos.AddItem tb("texto")

tb.MoveNext
Loop


tb.Close
```

```
bd.Close
Set tb = Nothing
Set bd = Nothing
End Function

Public Function Selecionar_da_biblioteca(indice As Integer)
Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long

Set bd = DBEngine.OpenDatabase(App.Path & "\bibliotxts.mdb")
Set tb = bd.OpenRecordset("textos", dbOpenTable)
On Error Resume Next
tb.MoveFirst
Do Until tb.EOF

If tb("texto") = lstTextos.List(indice) Then

txtTexto.Text = tb("caminho")
Visualizar_texto tb("caminho")
strCaminho = tb("caminho")
Else

End If


tb.MoveNext
Loop


tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing

End Function
```

ANEXO 16 –Formulário frm3



ANEXO 17 – Código do formulário frm3

```
Private Sub cmdAvançar_Click()
Load frm4
frm3.Hide
frm4.Show
End Sub

Private Sub cmdCancelar_Click()
Unload frm1
Unload frm2
Unload frm3
Set frm1 = Nothing
Set frm2 = Nothing
Set frm3 = Nothing
End Sub

Private Sub cmdfrequencia_Click()
Load dlgInfo
dlgInfo.rtb1.LoadFile strCaminho
dlgInfo.Caption = "Lista de Freqüências - [Ordem: Freqüencial]"
varCor = vbBlack
varFlag = False


Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long

Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
```

```
Set tb = bd.OpenRecordset("SELECT [palavra],[frequencia],[porcentagem] FROM [formas]
ORDER BY [frequencia] DESC")

With dlgInfo.msf1
.Rows = tb.RecordCount + 1
.Cols = 4
tb.MoveFirst
Do Until tb.EOF
i = i + 1
.TextMatrix(i, 0) = i
.TextMatrix(i, 1) = tb("palavra")
.TextMatrix(i, 2) = tb("frequencia")
.TextMatrix(i, 3) = tb("porcentagem")

tb.MoveNext
Loop
.TextMatrix(0, 0) = "Ordem"
.TextMatrix(0, 1) = "Palavra"
.TextMatrix(0, 2) = "Freqüência"
.TextMatrix(0, 3) = "Em %"
.ColAlignment(1) = flexAlignRightCenter
.ColAlignment(2) = flexAlignCenterCenter
.ColAlignment(3) = flexAlignCenterCenter
.ColWidth(0) = 800
.ColWidth(1) = 2500
.ColWidth(2) = 1200
.ColWidth(3) = 1000
.Col = 0
.Row = 0
.CellAlignment = flexAlignCenterCenter
.Col = 1
.Row = 0
.CellAlignment = flexAlignCenterCenter


tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing


dlgInfo.Show 1
End With
End Sub

Private Sub cmdVoltar_Click()
Unload frm3
frm2.Show
End Sub

Private Sub Command1_Click()
Load dlgInfo
dlgInfo.rtb1.LoadFile strCaminho
dlgInfo.Caption = "Palavras-chave - [Ordem: Chavicidade]"

varCor = vbBlue
varFlag = True

Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long
Dim matches As MatchCollection
Dim a_match As Match
```

```
With dlgInfo.rtb1

'On Error Resume Next
Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("SELECT [palavra],[chavicidade] FROM [formas] WHERE
[chavicidade] > '1' ORDER BY [chavicidade] DESC")


dlgInfo.msf1.Rows = tb.RecordCount + 1
dlgInfo.msf1.Cols = 3


tb.MoveFirst
Do Until tb.EOF


   i = i + 1
With dlgInfo.msf1
.TextMatrix(i, 0) = i
.TextMatrix(i, 1) = tb("palavra")
.TextMatrix(i, 2) = tb("chavicidade")
End With


    Dim reg_exp As New RegExp
    reg_exp.Pattern = "\b" & tb("palavra") & "\b"
    reg_exp.IgnoreCase = True
    reg_exp.Global = True


    ' Color the matched text.
    Set matches = reg_exp.Execute(.Text)
    For Each a_match In matches
    .SelStart = a_match.FirstIndex
        .SelLength = a_match.Length
        .SelColor = vbBlue
        .SelBold = True
    Next a_match

    .SelStart = 0
    .SelLength = 0

tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing
End With

With dlgInfo.msf1
.ColWidth(0) = 800
.ColWidth(1) = 2500
.ColWidth(2) = 1200
.TextMatrix(0, 0) = "Ordem"
.TextMatrix(0, 1) = "Palavra"
.TextMatrix(0, 2) = "Chavicidade"
.ColAlignment(1) = flexAlignRightCenter
.ColAlignment(2) = flexAlignCenterCenter
.Col = 0
.Row = 0
```

```
.CellAlignment = flexAlignCenterCenter
.Col = 1
.Row = 0
.CellAlignment = flexAlignCenterCenter


End With

dlgInfo.Show 1
End Sub

Private Sub Command2_Click()
Load dlgInfo

dlgInfo.rtb1.LoadFile strCaminho
dlgInfo.Caption = "Possíveis cognatos - [Ordem: Alfabética]"
varCor = vbBlue
varFlag = True


Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long

Dim matches As MatchCollection
Dim a_match As Match
With dlgInfo.rtb1
    On Error Resume Next

Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("SELECT [palavra], [frequencia] FROM [formas] WHERE
[cognato] > '0' ORDER BY [palavra] ASC")

dlgInfo.msf1.Rows = tb.RecordCount + 1
dlgInfo.msf1.Cols = 3


tb.MoveFirst
Do Until tb.EOF

    Dim reg_exp As New RegExp
    reg_exp.Pattern = "\b" & tb("palavra") & "\b"
    reg_exp.IgnoreCase = True
    reg_exp.Global = True

   i = i + 1
With dlgInfo.msf1
.TextMatrix(i, 0) = i
.TextMatrix(i, 1) = tb("palavra")
.TextMatrix(i, 2) = tb("frequencia")
End With


    ' Color the matched text.
    Set matches = reg_exp.Execute(.Text)
    For Each a_match In matches
    .SelStart = a_match.FirstIndex
        .SelLength = a_match.Length
        .SelColor = vbBlue
        .SelBold = True
    Next a_match

    .SelStart = 0
```

```
    .SelLength = 0

tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing
End With

With dlgInfo.msf1
.ColWidth(0) = 800
.ColWidth(1) = 2500
.ColWidth(2) = 1200
.TextMatrix(0, 0) = "Ordem"
.TextMatrix(0, 1) = "Palavra"
.TextMatrix(0, 2) = "Freqüência"
.ColAlignment(1) = flexAlignRightCenter
.ColAlignment(2) = flexAlignCenterCenter
.Col = 0
.Row = 0
.CellAlignment = flexAlignCenterCenter
.Col = 1
.Row = 0
.CellAlignment = flexAlignCenterCenter
End With



dlgInfo.Show
End Sub

Private Sub Command3_Click()
Load dlgInfo

dlgInfo.rtb1.LoadFile strCaminho
dlgInfo.Caption = "Palvras de conteúdo - [Ordem: Alfabética]"
varCor = vbBlue
varFlag = True


Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long
Dim matches As MatchCollection
Dim a_match As Match
With dlgInfo.rtb1
    On Error Resume Next

Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("SELECT [palavra], [frequencia] FROM [formas] WHERE
[categoria] = '1'ORDER BY [palavra] ASC")

dlgInfo.msf1.Rows = tb.RecordCount + 1
dlgInfo.msf1.Cols = 3


tb.MoveFirst
Do Until tb.EOF
i = i + 1
    Dim reg_exp As New RegExp
    reg_exp.Pattern = "\b" & tb("palavra") & "\b"
```

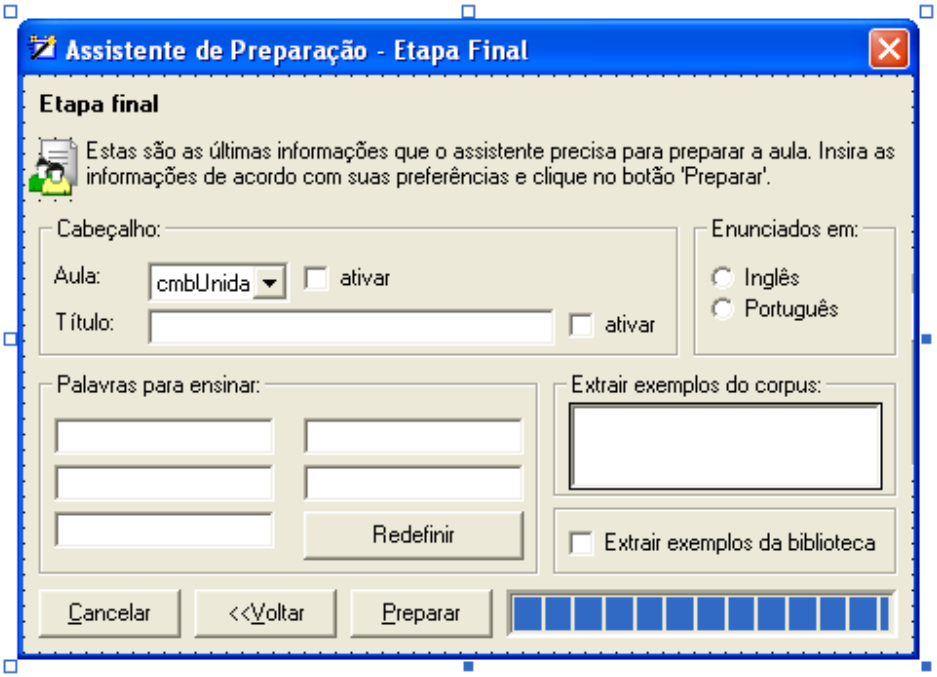
```
    reg_exp.IgnoreCase = True
    reg_exp.Global = True

With dlgInfo.msf1
.TextMatrix(i, 0) = i
.TextMatrix(i, 1) = tb("palavra")
.TextMatrix(i, 2) = tb("frequencia")
End With


    ' Color the matched text.
    Set matches = reg_exp.Execute(.Text)
    For Each a_match In matches
    .SelStart = a_match.FirstIndex
        .SelLength = a_match.Length
        .SelColor = vbBlue
        .SelBold = True
    Next a_match

    .SelStart = 0
    .SelLength = 0

tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing
End With

With dlgInfo.msf1
.ColWidth(0) = 800
.ColWidth(1) = 2500
.ColWidth(2) = 1200
.TextMatrix(0, 0) = "Ordem"
.TextMatrix(0, 1) = "Palavra"
.TextMatrix(0, 2) = "Freqüência"
.ColAlignment(1) = flexAlignRightCenter
.ColAlignment(2) = flexAlignCenterCenter
.Col = 0
.Row = 0
.CellAlignment = flexAlignCenterCenter
.Col = 1
.Row = 0
.CellAlignment = flexAlignCenterCenter
End With



dlgInfo.Show
End Sub

Private Sub Command4_Click()
Load dlgInfo

dlgInfo.rtb1.LoadFile strCaminho
dlgInfo.Caption = "Palavras gramaticais - [Ordem: Alfabética]"
varCor = vbBlue
varFlag = True


Dim bd As DAO.Database
```

```
Dim tb As DAO.Recordset
Dim i As Long
Dim matches As MatchCollection
Dim a_match As Match
With dlgInfo.rtb1
    On Error Resume Next

Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("SELECT [palavra], [frequencia] FROM [formas] WHERE
[categoria] = '0'ORDER BY [palavra] ASC")

dlgInfo.msf1.Rows = tb.RecordCount + 1
dlgInfo.msf1.Cols = 3

tb.MoveFirst
Do Until tb.EOF
i = i + 1
    Dim reg_exp As New RegExp
    reg_exp.Pattern = "\b" & tb("palavra") & "\b"
    reg_exp.IgnoreCase = True
    reg_exp.Global = True

With dlgInfo.msf1
.TextMatrix(i, 0) = i
.TextMatrix(i, 1) = tb("palavra")
.TextMatrix(i, 2) = tb("frequencia")
End With

    ' Color the matched text.
    Set matches = reg_exp.Execute(.Text)
    For Each a_match In matches
    .SelStart = a_match.FirstIndex
        .SelLength = a_match.Length
        .SelColor = vbBlue
        .SelBold = True
    Next a_match

    .SelStart = 0
    .SelLength = 0

tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing

End With

With dlgInfo.msf1
.ColWidth(0) = 800
.ColWidth(1) = 2500
.ColWidth(2) = 1200
.TextMatrix(0, 0) = "Ordem"
.TextMatrix(0, 1) = "Palavra"
.TextMatrix(0, 2) = "Freqüência"
.ColAlignment(1) = flexAlignRightCenter
.ColAlignment(2) = flexAlignCenterCenter
.Col = 0
.Row = 0
.CellAlignment = flexAlignCenterCenter
.Col = 1
```

```
.Row = 0
.CellAlignment = flexAlignCenterCenter
End With



dlgInfo.Show
End Sub


Private Sub Form_Load()
Definir_dificuldade
Definir_estatisticas
End Sub


Public Function Definir_dificuldade()
Dim i As Long

For i = 0 To 6
Image3(i).Visible = False
Next i

lbldificuldade.Caption = UCase(strDificuldade)

Select Case lbldificuldade.Caption
Case "MUITO DIFÍCIL"
Image3(6).Visible = True
Case "DIFÍCIL"
Image3(5).Visible = True
Case "RAZOAVELMENTE DIFÍCIL"
Image3(4).Visible = True
Case "PADRÃO"
Image3(3).Visible = True
Case "RAZOAVELMENTE FÁCIL"
Image3(2).Visible = True
Case "FÁCIL"
Image3(1).Visible = True
Case "MUITO FÁCIL"
Image3(0).Visible = True
Case Else

End Select



End Function

Public Function Definir_estatisticas()
lblrot(7).Caption = "Número de itens: " & lngPalavras
lblrot(8).Caption = "Número de formas: " & lngFormas
lblrot(9).Caption = "Razão forma/item: " & dblRazaofi & "%"
lblrot(10).Caption = "Possíveis cognatos: " & dblNcognatos & "%"
lblrot(11).Caption = "Densidade lexical: " & dblDensidadelexical & "%"


End Function

Private Sub Label1_Click()

End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
Unload frm1
Unload frm2
Unload frm3

End Sub
```

ANEXO 18 – Formulário frm4



ANEXO 19 – Código do formulário frm4

```
'O QUE ENSINAR
Dim Ex1 As String
Dim Ex2 As String
Dim Ex3 As String
Dim Ex4 As String
Dim Ex5 As String
Dim Ex6 As String
'PALAVRAS PARA ENSINAR
Dim strP() As String

Dim strP1 As String
Dim strP2 As String
Dim strP3 As String
Dim strP4 As String
Dim strP5 As String

'LINHAS DE CONCORDÂNCIAS
Dim Conc1 As String
Dim Conc2 As String
Dim Conc3 As String
Dim Conc4 As String
```

```
Dim Conc5 As String

Private Sub chkBiblioteca_Click()
If chkBiblioteca.Value = 1 Then
Frame4.Enabled = False

Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long

ReDim Biblio(0)

Set bd = DBEngine.OpenDatabase(App.Path & "\bibliotxts.mdb")
Set tb = bd.OpenRecordset("textos", dbOpenTable)

tb.MoveFirst
Do Until tb.EOF
i = i + 1
ReDim Preserve Biblio(i)
Biblio(i) = tb("caminho")

tb.MoveNext
Loop


tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing
cmdAvancar.Enabled = True
Else
Frame4.Enabled = True
End If
End Sub

Private Sub chkScanning_Click()
If chkscanning.Value = 1 Then
txtScanning.Enabled = True
txtScanning.SetFocus
lstperguntas.Enabled = True
Else
txtScanning.Enabled = False
txtScanning.Text = ""
lstperguntas.Clear
lstperguntas.Enabled = False
End If
End Sub

Private Sub chkTitulo_Click()
If chkTitulo.Value = 1 Then
txtTitulo.Enabled = True
txtTitulo.SetFocus
Else
txtTitulo.Enabled = False
txtTitulo.Text = ""
End If
End Sub

Private Sub chkUnidade_Click()
If chkUnidade.Value = 1 Then
cmbUnidade.Enabled = True
cmbUnidade.SetFocus
Else
```

```
cmbUnidade.Enabled = False

End If
End Sub

Private Sub cmdAdicionar_Click()
If chkscanning.Value = 0 Then Exit Sub
lstperguntas.AddItem lstperguntas.ListCount + 1 & ". " & txtScanning.Text
txtScanning.Text = ""
txtScanning.SetFocus
End Sub

Private Sub cmdAvancar_Click()
strP1 = txtp1.Text
strP2 = txtp2.Text
strP3 = txtp3.Text
strP4 = txtp4.Text
strP5 = txtp5.Text

pgb1.Visible = True
pgb1.Min = 1
pgb1.Max = 7
pgb1.Value = 1
DoEvents
Concordancias1
pgb1.Value = 2
DoEvents
Concordancias2
pgb1.Value = 3
DoEvents
Concordancias3
pgb1.Value = 4
DoEvents
Concordancias4
pgb1.Value = 5
DoEvents
Concordancias5
pgb1.Value = 6
DoEvents
Preparar_aula
pgb1.Value = 7

Unload frm1
Unload frm2
Unload frm3
Unload frm4

End Sub

Private Sub cmdCancelar_Click()
Unload frm1
Unload frm2
Unload frm3
Unload frm4
Set frm1 = Nothing
Set frm2 = Nothing
Set frm3 = Nothing
Set frm4 = Nothing
End Sub

Private Sub cmdLimpar_Click()
If chkscanning.Value = 0 Then Exit Sub
lstperguntas.Clear
```

```
End Sub

Private Sub cmdVoltar_Click()
Unload frm4
frm3.Show
End Sub

Private Sub Command1_Click()
txtp1.Enabled = True
txtp2.Enabled = True
txtp3.Enabled = True
txtp4.Enabled = True
txtp5.Enabled = True
txtp1.SetFocus
End Sub

Private Sub File1_Click()
If Len(File1.FileName) = 0 Then Exit Sub
Definir_corpus File1.FileName
cmdAvancar.Enabled = True
End Sub

Private Sub Form_Load()
File1.Path = App.Path & "\corpora\"
strAfixos =
"^IN\w+\b|^UN\w+\b|^ILL\w+\b|^NON\w+\b|^IRR\w+\b|^MIS\w+\b|^SUPER\w+\b|^OUT\w+\b|
^SUB\w+\b|^OVER\w+\b|^UNDER\w+\b|" & _

"^ULTRA\w+\b|^MINI\w+\b|^HYPER\w+\b|^INTER\w+\b|^TRANS\w+\b|^FORE\w+\b|^PRE\w+\b|^POST\w+\b|
^EX\w+\b|^RE\w+\b|" & _

"^AUTO\w+\b|^NEO\w+\b|^PROTO\w+\b|^SEMI\w+\b|^VICE\w+\b|ER\b|OR$|ANT$|ENT$|ATION$|" & _
            "TION$|ION$|MENT$|NESS$|SHIP$|HOOD$|IST$|ISM$|IFY$|IZE$|ISE$|" & _
            "LY$|WARDS$|ABLE$|IBLE$|ISH$|FUL$|LESS$|OUS$|EOUS$|IOUS$|IC$|" & _
            "IAL$|ICAL$|IVE$|ATIVE$|ITIVE$"

Palavras_mais_frequentes
cognatos
Palavras_chave
Conjuncoes
Pronomes
Afixos
Definir_Palavras_de_ensino

txtScanning.Enabled = True
Preencher_cmbUnidade

txtp1.Text = strP1
txtp2.Text = strP2
txtp3.Text = strP3
txtp4.Text = strP4
txtp5.Text = strP5

txtScanning.Enabled = False
lstperguntas.Enabled = False


txtp1.Enabled = False
txtp2.Enabled = False
txtp3.Enabled = False
txtp4.Enabled = False
txtp5.Enabled = False
```

```
cmbUnidade.Enabled = False
txtTitulo.Enabled = False
optPortugues.Value = True
End Sub

Public Function Palavras_mais_frequentes()
Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long
Dim n As String
Dim p As String
Dim G() As String

Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")

Set tb = bd.OpenRecordset("SELECT [palavra], [frequencia], [porcentagem] FROM [formas] WHERE
[categoria] = '0' ORDER BY [frequencia] DESC")

i = 0
n = ""
tb.MoveFirst
Do Until tb.EOF
i = i + 1
n = CStr(i)
If Len(CStr(i)) = 1 Then n = " " & CStr(i)
p = tb("palavra")
If Len(p) < 15 Then p = p & Space(15 - Len(p))
ReDim Preserve G(i)
G(i) = n & ". " & p & " " & tb("frequencia") & " (" & tb("porcentagem") & "%) "
If i = nEx1 Then Exit Do
tb.MoveNext
Loop

ReDim Preserve G(i + 1)

Set tb = bd.OpenRecordset("SELECT [palavra], [frequencia], [porcentagem] FROM [formas] WHERE
[categoria] = '1' ORDER BY [frequencia] DESC")

i = 0
tb.MoveFirst
Do Until tb.EOF
i = i + 1
n = CStr(i)
If Len(CStr(i)) = 1 Then n = " " & CStr(i)
p = tb("palavra")
If Len(p) < 15 Then p = p & Space(15 - Len(p))
Ex1 = Ex1 & n & ". " & p & " " & tb("frequencia") & " (" & tb("porcentagem") & "%) " & vbTab
& " - " & vbTab & G(i) & vbNewLine
If i = nEx1 Then Exit Do
tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing
End Function

Public Function cognatos()
Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long
```

```
Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("SELECT [palavra]FROM [formas] WHERE [cognato] = '1' ORDER BY
[chavicidade] DESC")

Ex2 = "- "
tb.MoveFirst
Do Until tb.EOF
i = i + 1
Ex2 = Ex2 & tb("palavra") & " - "
If i = nEx2 Then Exit Do
tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing
End Function


Public Function Palavras_chave()
Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long

Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("SELECT [palavra]FROM [formas] WHERE [categoria] = '1' ORDER BY
[chavicidade] DESC")

Ex3 = "- "
tb.MoveFirst
Do Until tb.EOF
i = i + 1
ReDim Preserve strP(i)
strP(i) = tb("palavra")
Ex3 = Ex3 & tb("palavra") & " - "
If i = nEx3 Then Exit Do
tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing
End Function

Public Function Preparar_aula()
Load frmFinal
Definir_lingua
Dim i As Long
Dim strScanning As String
Dim strAdicionais As String

Dim l1 As String
Dim l2 As String

If optIngles.Value = True Then
l1 = "Lexical Words" & Space$(7)
l2 = "Grammatical Words"
Else
l1 = "Palavras de Conteúdo"
l2 = "Palavras Gramaticais"
End If
```

```
Ex1 = String(5, " ") & l1 & vbTab & "  " & vbTab & String(10, " ") & l2 & vbNewLine & Ex1

frmFinal.rtb1.LoadFile App.Path & strModelo
frmFinal.rtb1.Find ("<palavras mais freqüentes>")
frmFinal.rtb1.SelText = Ex1

If optIngles.Value = True Then strScanning = "SCANNING"
If optPortugues.Value = True Then strScanning = "PERGUNTAS ESPECÍFICAS"

frmFinal.rtb1.Find ("<p1>")
frmFinal.rtb1.SelText = "10"
frmFinal.rtb1.Find ("<p2>")
frmFinal.rtb1.SelText = "11"
'frmFinal.rtb1.Find ("<p3>")
'frmFinal.rtb1.SelText = "12"
frmFinal.rtb1.Find ("<p4>")
frmFinal.rtb1.SelText = "12"

If chkscanning.Value = 1 Then
For i = 0 To lstperguntas.ListCount - 1
strAdicionais = strAdicionais & lstperguntas.List(i) & vbNewLine
Next i
frmFinal.rtb1.Find ("<PERGUNTAS ESPECÍFICAS>")
frmFinal.rtb1.SelText = vbNewLine & strScanning & vbNewLine
frmFinal.rtb1.Find ("<perguntas adicionais>")
frmFinal.rtb1.SelText = vbNewLine & strAdicionais & vbNewLine
Else
frmFinal.rtb1.Find ("<PERGUNTAS ESPECÍFICAS>")
frmFinal.rtb1.SelText = ""
frmFinal.rtb1.Find ("<perguntas adicionais>")
frmFinal.rtb1.SelText = ""
End If

If chkUnidade.Value = 1 Then
frmFinal.rtb1.Find ("<unidade>")
frmFinal.rtb1.SelFontSize = 20
frmFinal.rtb1.SelText = strUnidade & " " & cmbUnidade.Text
Else
frmFinal.rtb1.Find ("<unidade>")
frmFinal.rtb1.SelFontSize = 1
frmFinal.rtb1.SelText = ""
End If

If chkTitulo.Value = 1 Then
frmFinal.rtb1.Find ("<título>")
frmFinal.rtb1.SelFontSize = 14
frmFinal.rtb1.SelText = txtTitulo.Text
Else
frmFinal.rtb1.Find ("<título>")
frmFinal.rtb1.SelFontSize = 1
frmFinal.rtb1.SelText = ""
End If

frmFinal.rtb1.Find ("<palavras cognatas>")
frmFinal.rtb1.SelText = Ex2

frmFinal.rtb1.Find ("<palavras-chave>")
frmFinal.rtb1.SelText = Ex3

frmFinal.rtb1.Find ("<razão fi>")
frmFinal.rtb1.SelText = dblRazaofi & "%"
```

```
frmFinal.rtb1.Find ("<ncognatas>")
frmFinal.rtb1.SelText = dblNcognatos & "%"

If optIngles.Value = True Then

If strDificuldade = "muito difícil" Then strDificuldade = "very hard"
If strDificuldade = "difícil" Then strDificuldade = "hard"
If strDificuldade = "razoavelmente difícil" Then strDificuldade = "fairly hard"

If strDificuldade = "muito fácil" Then strDificuldade = "very easy"
If strDificuldade = "fácil" Then strDificuldade = "easy"
If strDificuldade = "rozoavelmente fácil" Then strDificuldade = "fairly easy"

If strDificuldade = "padrão" Then strDificuldade = "standard"


Else
End If


frmFinal.rtb1.Find ("<dificuldade>")
frmFinal.rtb1.SelText = strDificuldade

frmFinal.rtb1.Find ("<conjunções>")
frmFinal.rtb1.SelText = Ex4

frmFinal.rtb1.Find ("<pronomes>")
frmFinal.rtb1.SelText = Ex5

frmFinal.rtb1.Find ("<palavras com afixos>")
frmFinal.rtb1.SelText = Ex6

frmFinal.rtb1.Find ("<conc1>")
frmFinal.rtb1.SelText = Conc1

frmFinal.rtb1.Find ("<conc2>")
frmFinal.rtb1.SelText = Conc2

frmFinal.rtb1.Find ("<conc3>")
frmFinal.rtb1.SelText = Conc3

frmFinal.rtb1.Find ("<conc4>")
frmFinal.rtb1.SelText = Conc4

frmFinal.rtb1.Find ("<conc5>")
frmFinal.rtb1.SelText = Conc5

frmFinal.rtb1.SelStart = 1
frmFinal.Show
End Function

Public Function Conjuncoes()
Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long
Dim p As String
Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("SELECT [palavra]FROM [formas] WHERE [tag] LIKE 'cj*' ORDER BY
[chavicidade] DESC")


tb.MoveFirst
```

```
Do Until tb.EOF
i = i + 1
p = tb("palavra")

If Len(p) < 15 Then p = p & Space(15 - Len(p))
Ex4 = Ex4 & p & String(25, ".") & String(5, " ") & String(20, ".") & vbNewLine
If i = nEx4 Then Exit Do
tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing

End Function

Public Function Pronomes()
Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long
Dim p As String
Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("SELECT [palavra]FROM [formas] WHERE [tag] LIKE 'pn*' ORDER BY
[chavicidade] DESC")

On Error Resume Next
tb.MoveFirst
Do Until tb.EOF
i = i + 1
p = tb("palavra")

If Len(p) < 15 Then p = p & Space(15 - Len(p))
Ex5 = Ex5 & p & String(25, ".") & vbNewLine
If i = nEx5 Then Exit Do
tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing

End Function

Public Function Afixos()
Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long
Dim p As String
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection


Set bd = DBEngine.OpenDatabase(App.Path & "\objtexto1.mtz")
Set tb = bd.OpenRecordset("SELECT [palavra]FROM [formas] ORDER BY [cognato] DESC")


tb.MoveFirst
Do Until tb.EOF

p = tb("palavra")
```

```
r.Pattern = strAfixos
r.Global = True
r.IgnoreCase = True

Set c = r.Execute(p)

For Each M In c

Ex6 = Ex6 & " " & p
i = i + 1

Next

If i = nEx6 Then Exit Do

tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing


End Function

Public Function Concordancias1()
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim t As Long
Dim i As Integer
Dim j As Long
Dim n As String
Dim Mem As New clsM
Dim l As Long


DoEvents
r.Pattern = "(.{40})\b" & strP1 & "\b(.{40})" '& "|" & "(.{40})\b" & strP1 & ",\b(.{40})" &
"|" & "(.{40})\b" & strP1 & "\.\b(.{40})"
r.Global = True
r.IgnoreCase = True

DoEvents
For j = 1 To UBound(Biblio)
l = l + 1
If l = 100 Then GoTo Sair
DoEvents
Open Biblio(j) For Input As #1
t = Mem.malloc(Input(LOF(1), #1))
Close #1

Set c = r.Execute(Replace(Mem.RetMemory(t), vbNewLine, ""))
DoEvents
For Each M In c
l = 0
i = i + 1
n = CStr(i)
If Len(CStr(i)) = 1 Then n = " " & CStr(i)
Conc1$ = Conc1$ & n & ". " & M.Value & vbNewLine
If i = nEx7 Then GoTo Sair
```

```
DoEvents
Next
DoEvents
Mem.FreeMemory (t)

Next j
DoEvents

Sair:
Set r = Nothing
Set M = Nothing
Set c = Nothing
Set Mem = Nothing
End Function
Public Function Concordancias2()
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim t As Long
Dim i As Integer
Dim j As Long
Dim n As String
Dim Mem As New clsM
Dim l As Long

DoEvents
r.Pattern = "(.{40})\b" & strP2 & "\b(.{40})" '& "|" & "(.{40})\b" & strP2 & ",\b(.{40})" &
"|" & "(.{40})\b" & strP2 & "\.\b(.{40})"
r.Global = True
r.IgnoreCase = True

DoEvents
For j = 1 To UBound(Biblio)
l = l + 1
If l = 100 Then GoTo Sair
DoEvents
Open Biblio(j) For Input As #1
t = Mem.malloc(Input(LOF(1), #1))
Close #1

Set c = r.Execute(Replace(Mem.RetMemory(t), vbNewLine, ""))
DoEvents
For Each M In c
l = 0
DoEvents
i = i + 1
n = CStr(i)
If Len(CStr(i)) = 1 Then n = " " & CStr(i)
Conc2$ = Conc2$ & n & ". " & M.Value & vbNewLine
If i = nEx7 Then GoTo Sair
Next
DoEvents
Mem.FreeMemory (t)
Next j
DoEvents

Sair:
Set r = Nothing
Set M = Nothing
Set c = Nothing
Set Mem = Nothing
End Function
Public Function Concordancias3()
```

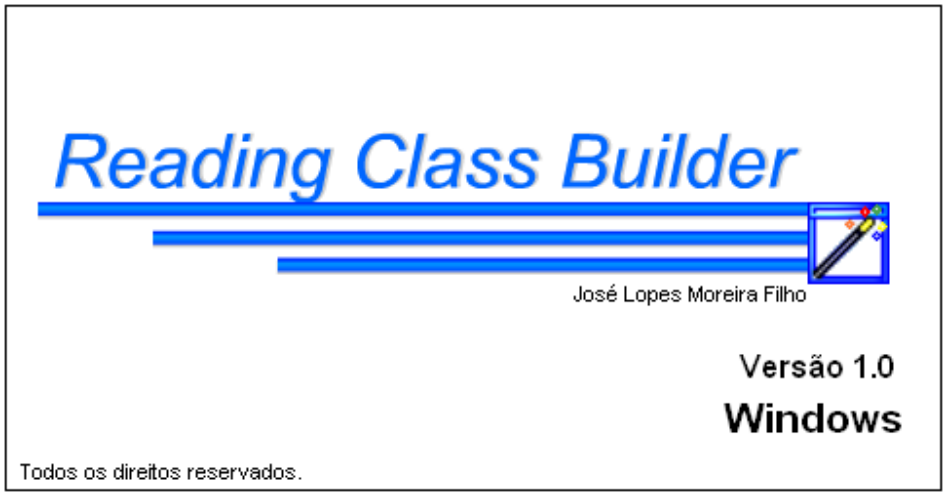
```
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim t As Long
Dim i As Integer
Dim j As Long
Dim n As String
Dim Mem As New clsM
Dim l As Long

DoEvents
r.Pattern = "(.{40})\b" & strP3 & "\b(.{40})" '& "|" & "(.{40})\b" & strP3 & ",\b(.{40})" &
"|" & "(.{40})\b" & strP3 & "\.\b(.{40})"
r.Global = True
r.IgnoreCase = True

DoEvents
For j = 1 To UBound(Biblio)
l = l + 1
If l = 100 Then GoTo Sair

DoEvents
Open Biblio(j) For Input As #1
t = Mem.malloc(Input(LOF(1), #1))
Close #1

Set c = r.Execute(Replace(Mem.RetMemory(t), vbNewLine, ""))
DoEvents
For Each M In c
l = 0
DoEvents
i = i + 1
n = CStr(i)
If Len(CStr(i)) = 1 Then n = " " & CStr(i)
Conc3$ = Conc3$ & n & ". " & M.Value & vbNewLine
If i = nEx7 Then GoTo Sair
DoEvents
Next

Mem.FreeMemory (t)
DoEvents
Next j
DoEvents
Sair:
Set r = Nothing
Set M = Nothing
Set c = Nothing
Set Mem = Nothing
End Function

Public Function Concordancias4()
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim t As Long
Dim i As Integer
Dim j As Long
Dim n As String
Dim Mem As New clsM
Dim l As Long

DoEvents
r.Pattern = "(.{40})\b" & strP4 & "\b(.{40})" '& "|" & "(.{40})\b" & strP4 & ",\b(.{40})" &
```

```
"|" & "(.{40})\b" & strP4 & "\.\b(.{40})"
r.Global = True
r.IgnoreCase = True

DoEvents
For j = 1 To UBound(Biblio)
l = l + 1
If l = 100 Then GoTo Sair
DoEvents
Open Biblio(j) For Input As #1
t = Mem.malloc(Input(LOF(1), #1))
Close #1

Set c = r.Execute(Replace(Mem.RetMemory(t), vbNewLine, ""))
DoEvents
For Each M In c
l = 0
DoEvents
i = i + 1
n = CStr(i)
If Len(CStr(i)) = 1 Then n = " " & CStr(i)
Conc4$ = Conc4$ & n & ". " & M.Value & vbNewLine
If i = nEx7 Then GoTo Sair
Next
DoEvents
Mem.FreeMemory (t)

Next j
DoEvents
Sair:
Set r = Nothing
Set M = Nothing
Set c = Nothing
Set Mem = Nothing
End Function
Public Function Concordancias5()
Dim r As New RegExp
Dim M As Match
Dim c As MatchCollection
Dim t As Long
Dim i As Integer
Dim j As Long
Dim n As String
Dim Mem As New clsM
Dim l As Long

DoEvents
r.Pattern = "(.{40})\b" & strP5 & "\b(.{40})" '& "|" & "(.{40})\b" & strP5 & ",\b(.{40})" &
"|" & "(.{40})\b" & strP5 & "\.\b(.{40})"
r.Global = True
r.IgnoreCase = True

DoEvents
For j = 1 To UBound(Biblio)
l = l + 1
If l = 100 Then GoTo Sair
DoEvents
Open Biblio(j) For Input As #1
t = Mem.malloc(Input(LOF(1), #1))
Close #1

Set c = r.Execute(Replace(Mem.RetMemory(t), vbNewLine, ""))
DoEvents
```

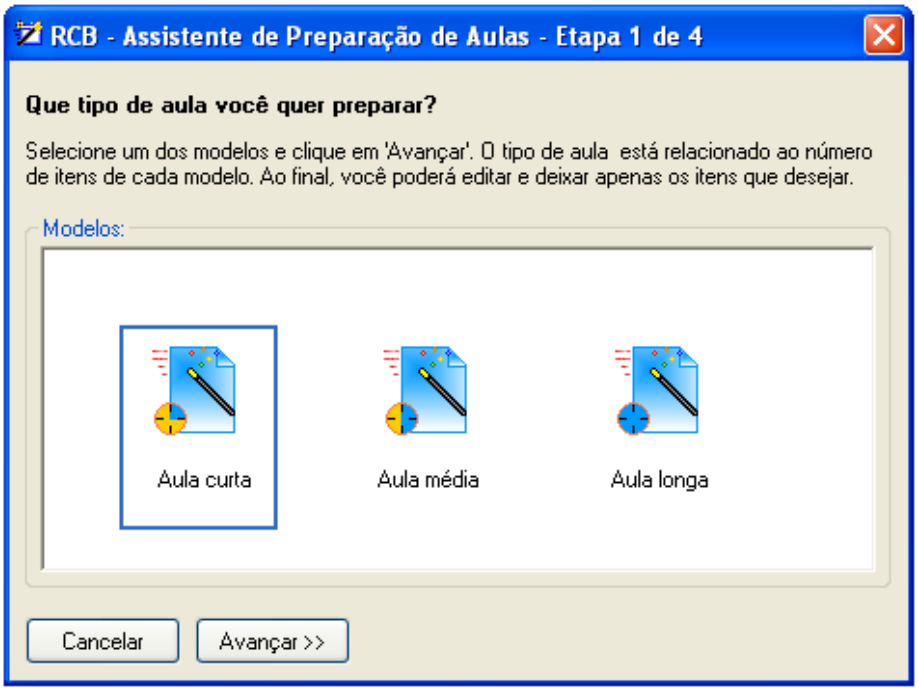
```
For Each M In c
l = 0
DoEvents
i = i + 1
n = CStr(i)
If Len(CStr(i)) = 1 Then n = " " & CStr(i)
Conc5$ = Conc5$ & n & ". " & M.Value & vbNewLine
If i = nEx7 Then GoTo Sair
DoEvents
Next
DoEvents
Mem.FreeMemory (t)
Next j
Sair:
Set r = Nothing
Set M = Nothing
Set c = Nothing
Set Mem = Nothing
End Function
Public Function Definir_Palavras_de_ensino()
strP1 = strP(1)
strP2 = strP(2)
strP3 = strP(3)
strP4 = strP(4)
strP5 = strP(5)

End Function
Public Function Definir_corpus(strCorpus As String)
Dim bd As DAO.Database
Dim tb As DAO.Recordset
Dim i As Long

Set bd = DBEngine.OpenDatabase(App.Path & "\corpora\" & strCorpus)
Set tb = bd.OpenRecordset("textos", dbOpenTable)
ReDim Biblio(0)
tb.MoveFirst
Do Until tb.EOF
i = i + 1
ReDim Preserve Biblio(i)
Biblio(i) = App.Path & "\corpora\" & tb("caminho") & ".dtc"

tb.MoveNext
Loop

tb.Close
bd.Close
Set tb = Nothing
Set bd = Nothing
End Function

Public Function Definir_cabecalho()

End Function

Public Function Definir_lingua()
If optIngles.Value = True Then
strModelo = "\mdl2.als"
strUnidade = "Lesson"

Else
End If

If optPortugues.Value = True Then
```

```
strModelo = "\mdl1.als"
strUnidade = "Aula"
Else
End If
End Function

Private Sub Form_Unload(Cancel As Integer)
Unload frm1
Unload frm2
Unload frm3
Unload frm4
End Sub

Private Sub txtp1_LostFocus()
txtp1.Text = UCase(txtp1.Text)
End Sub

Public Function Preencher_cmbUnidade()
cmbUnidade.AddItem "I"
cmbUnidade.AddItem "II"
cmbUnidade.AddItem "III"
cmbUnidade.AddItem "IV"
cmbUnidade.AddItem "V"
cmbUnidade.AddItem "VI"
cmbUnidade.AddItem "VII"
cmbUnidade.AddItem "VIII"
cmbUnidade.AddItem "IX"
cmbUnidade.AddItem "X"
cmbUnidade.AddItem "1"
cmbUnidade.AddItem "2"
cmbUnidade.AddItem "3"
cmbUnidade.AddItem "4"
cmbUnidade.AddItem "5"
cmbUnidade.AddItem "6"
cmbUnidade.AddItem "7"
cmbUnidade.AddItem "8"
cmbUnidade.AddItem "9"
cmbUnidade.AddItem "10"
cmbUnidade.AddItem "ONE"
cmbUnidade.AddItem "TWO"
cmbUnidade.AddItem "THREE"
cmbUnidade.AddItem "FOUR"
cmbUnidade.AddItem "FIVE"
cmbUnidade.AddItem "SIX"
cmbUnidade.AddItem "SEVEN"
cmbUnidade.AddItem "EIGHT"
cmbUnidade.AddItem "NINE"
cmbUnidade.AddItem "TEN"

End Function

Private Sub txtScanning_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
If chkscanning.Value = 0 Then Exit Sub
lstperguntas.AddItem lstperguntas.ListCount + 1 & ". " & txtScanning.Text
txtScanning.Text = ""
txtScanning.SetFocus
Else
End If
End Sub
```

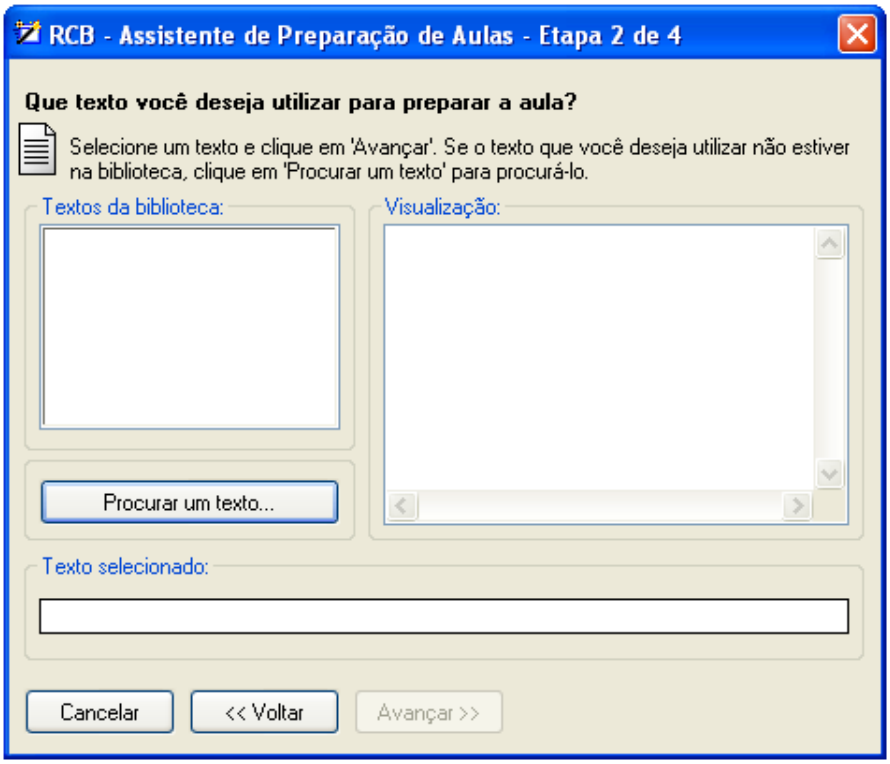
ANEXO 20 – *Splash screen* do produto final



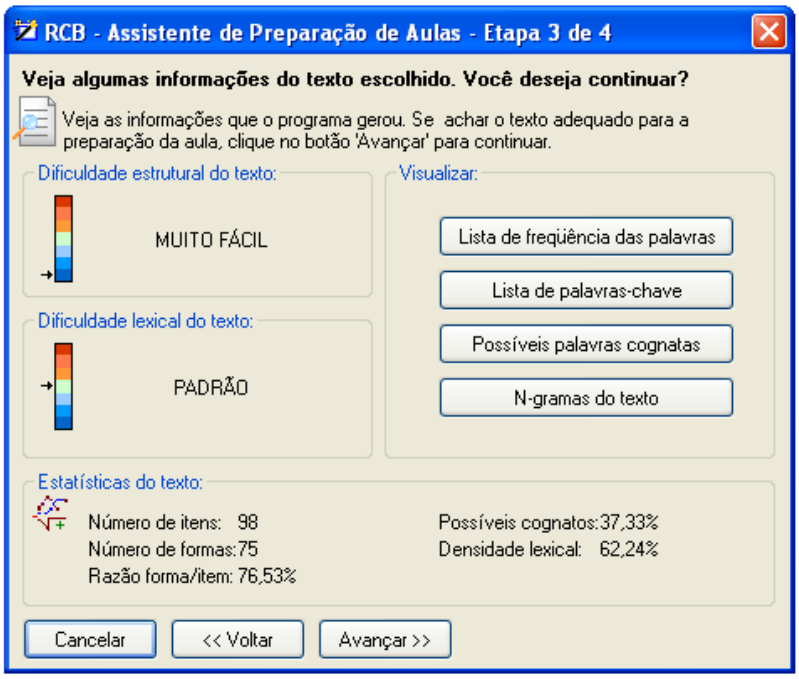
ANEXO 21 – Etapa 1 do Assistente de Preparação de Aulas



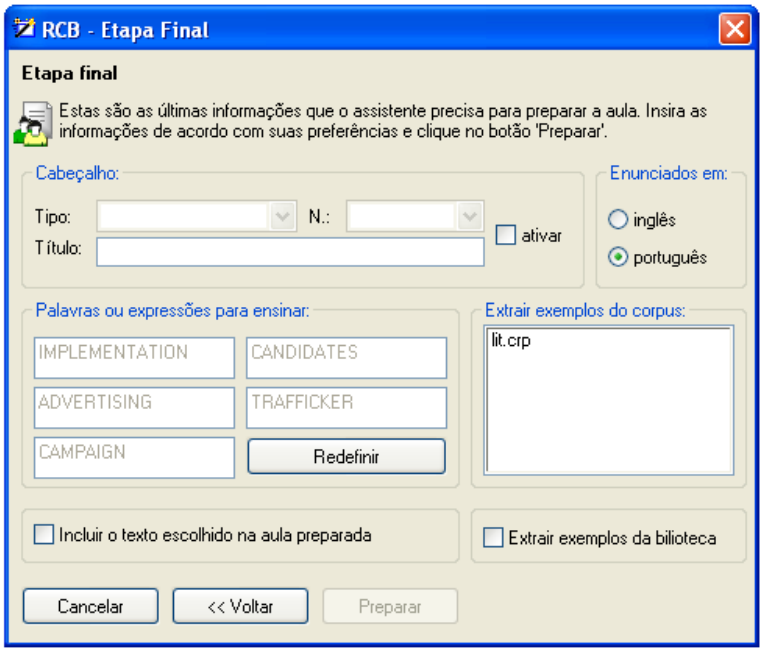
ANEXO 22 – Etapa 2 do Assistente de Preparação de Aulas



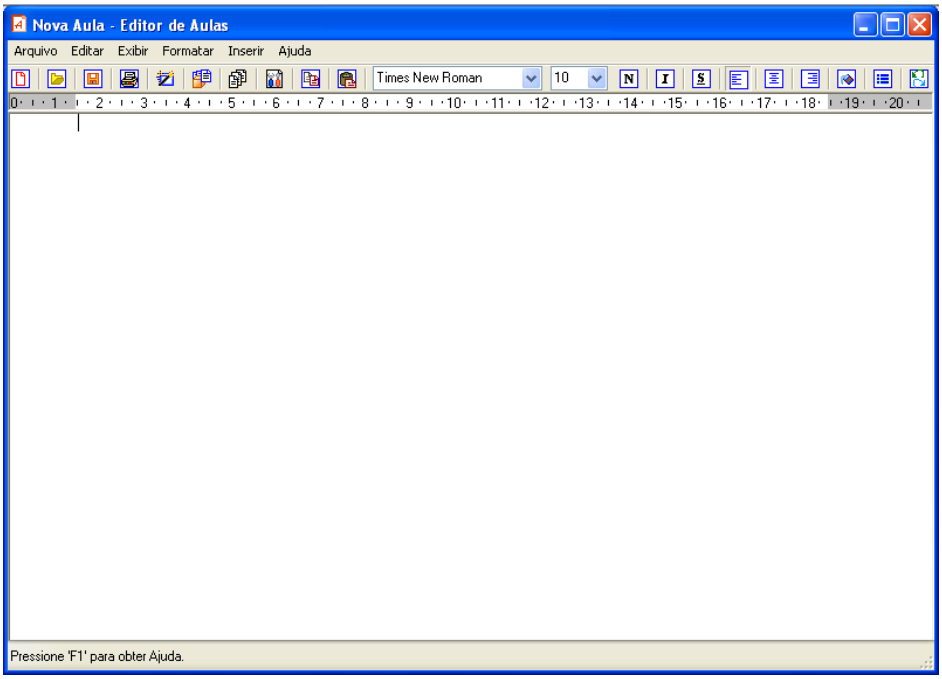
ANEXO 23 – Etapa 3 do Assistente de Preparação de Aulas



ANEXO 24 – Etapa 4 do Assistente de Preparação de Aulas

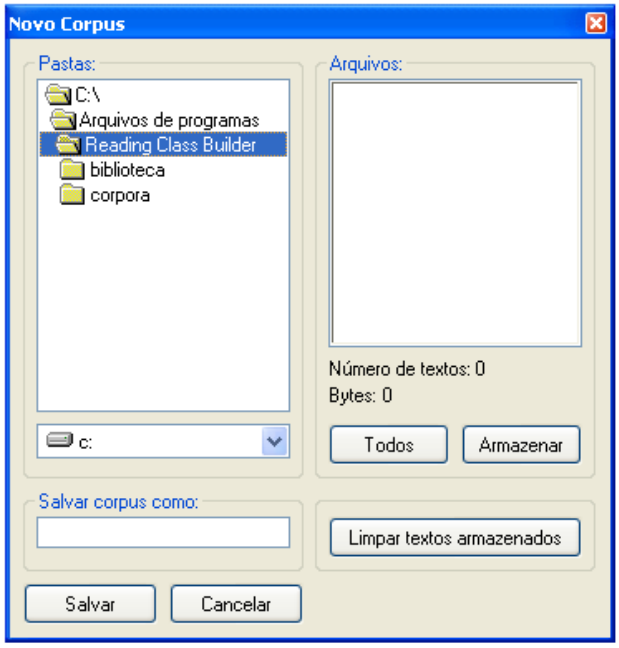


ANEXO 25 – Editor de Aulas

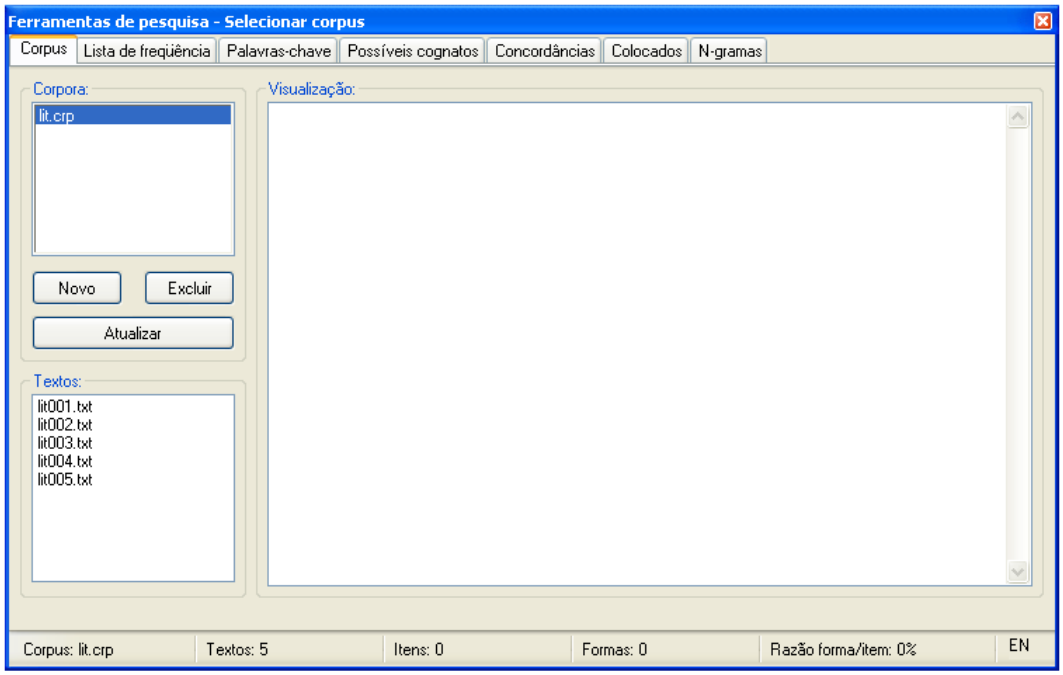


ANEXO 26 – Biblioteca

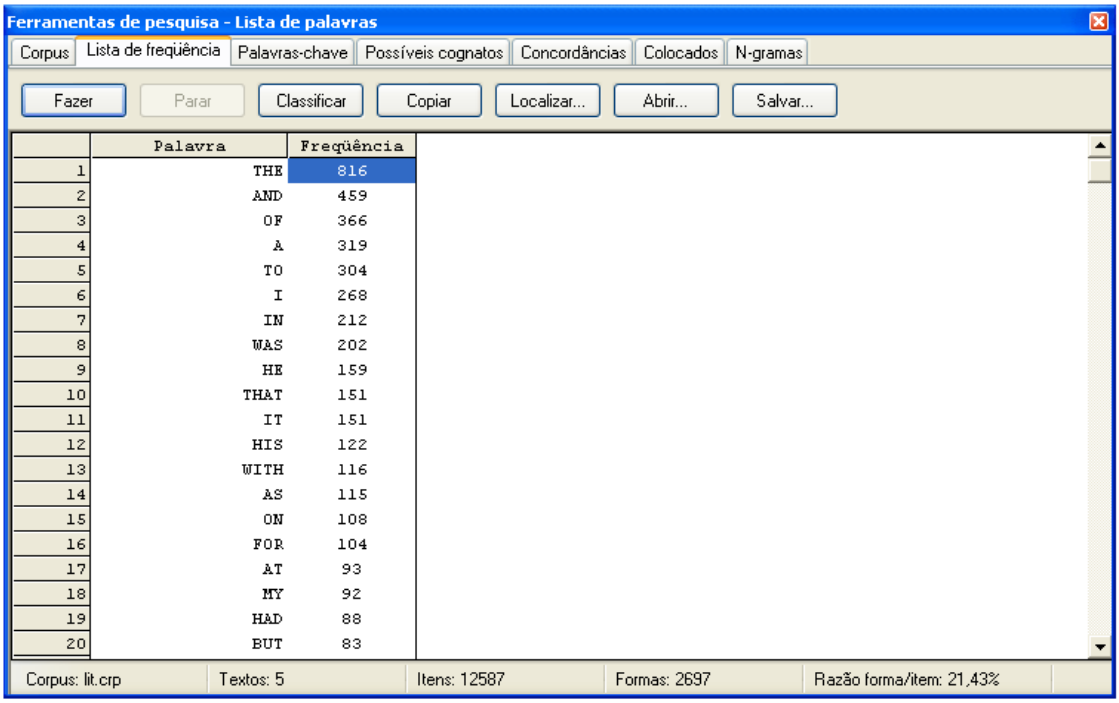
ANEXO 27 – Compilador (Novo Corpus)



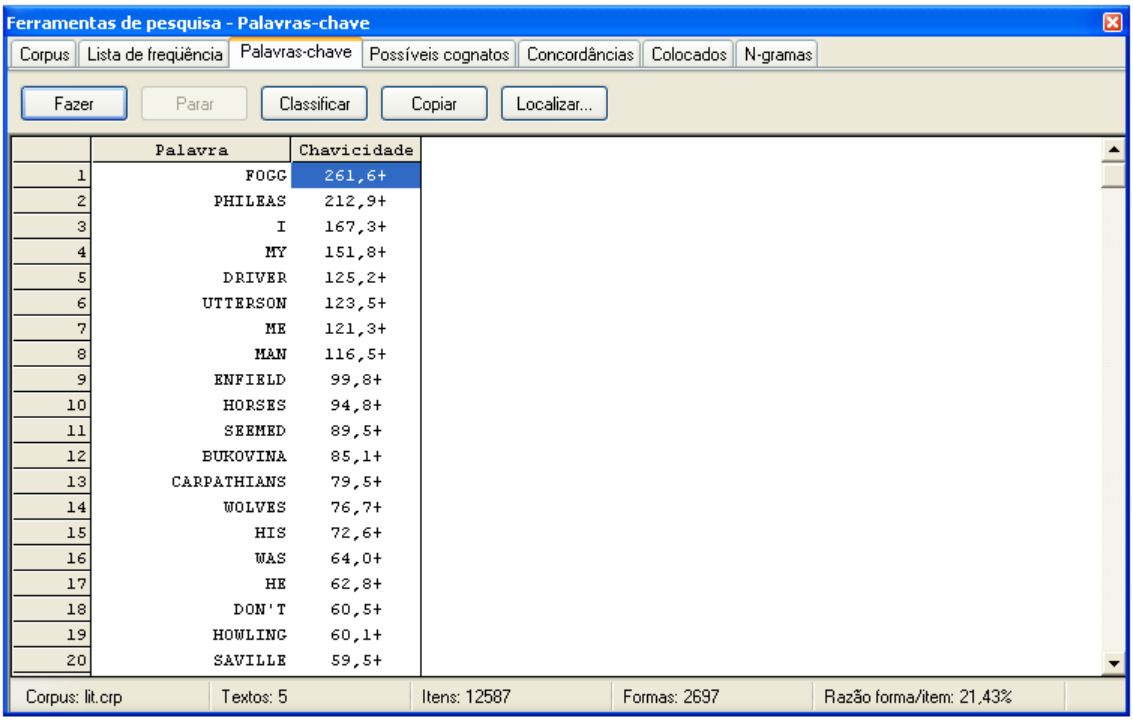
ANEXO 28 – Ferramentas de pesquisa – Selecionar Corpus



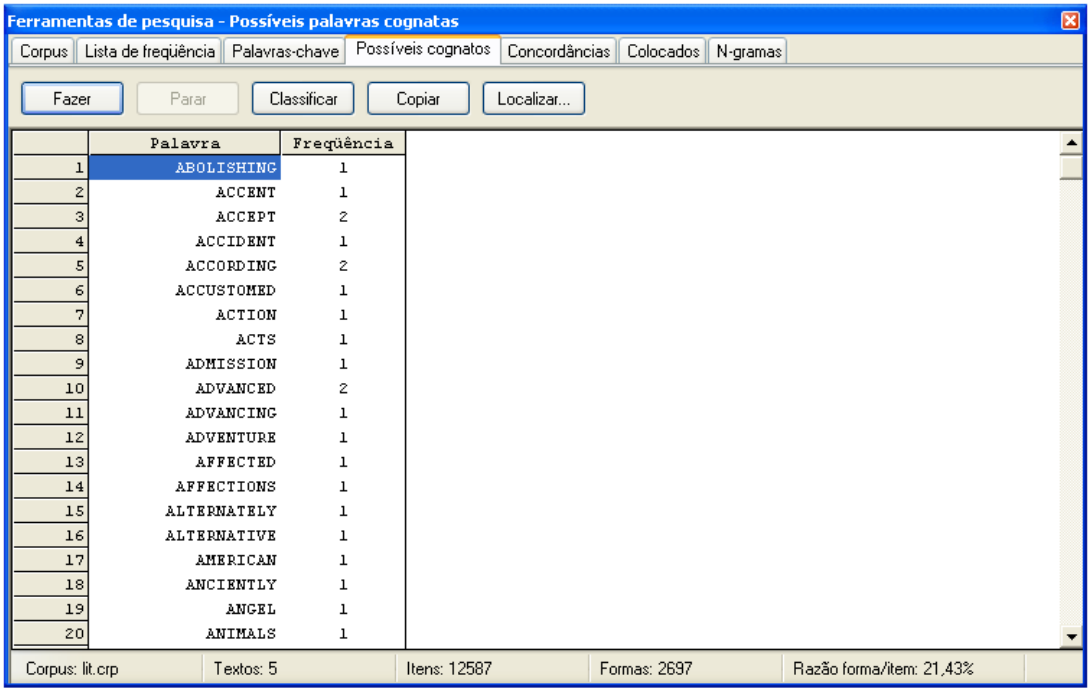
ANEXO 29 – Ferramentas de pesquisa – Lista de Palavras



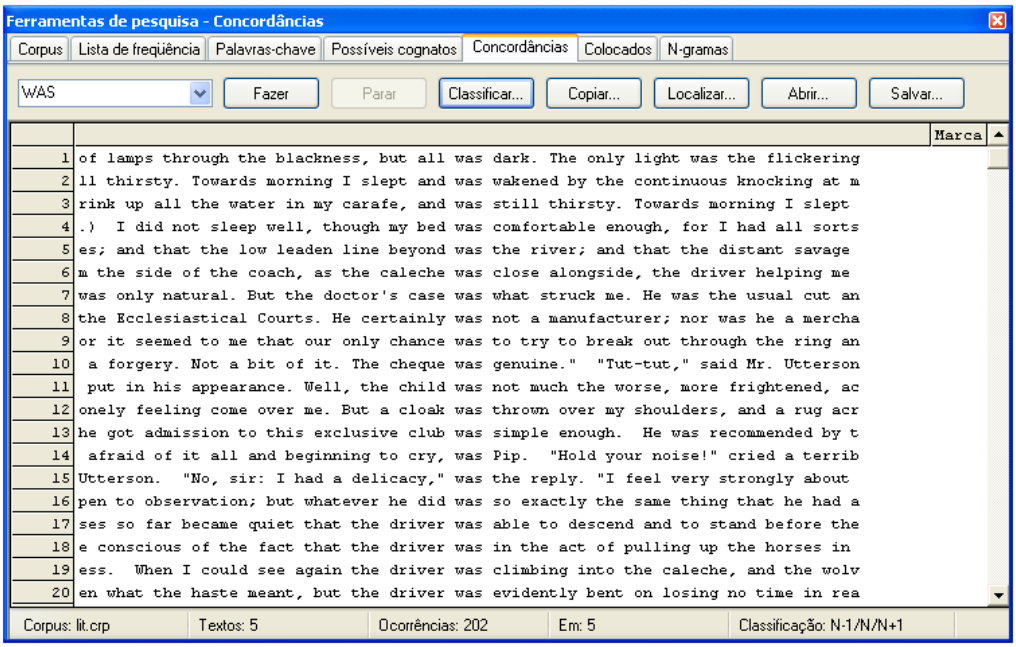
ANEXO 30 – Ferramentas de pesquisa – Palavras-chave



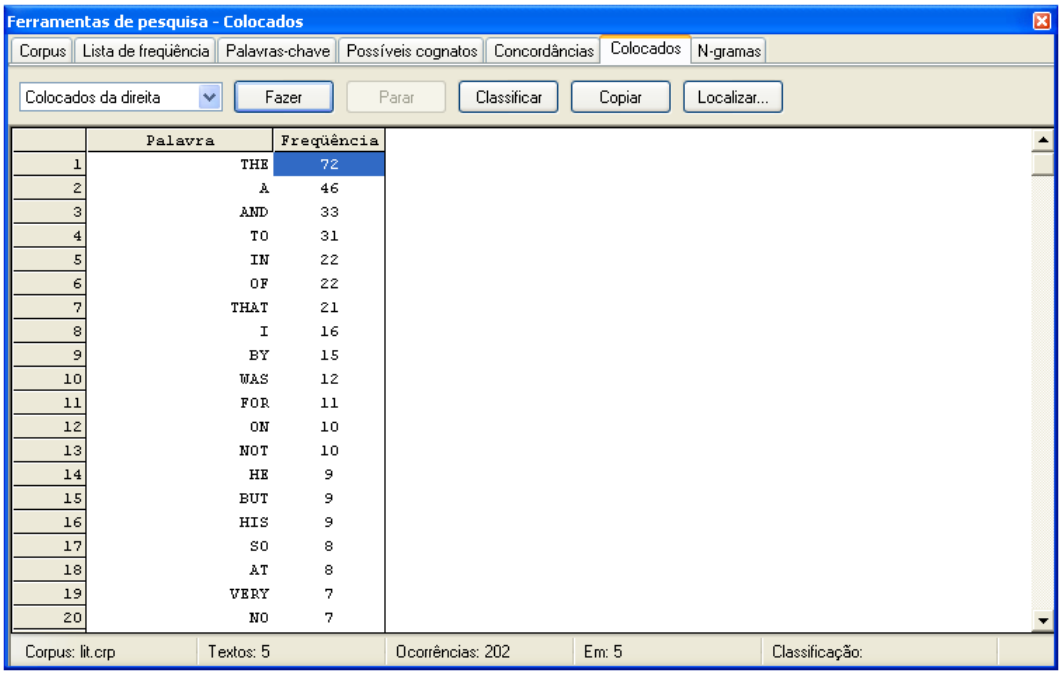
ANEXO 31 – Ferramentas de pesquisa – Possíveis cognatos



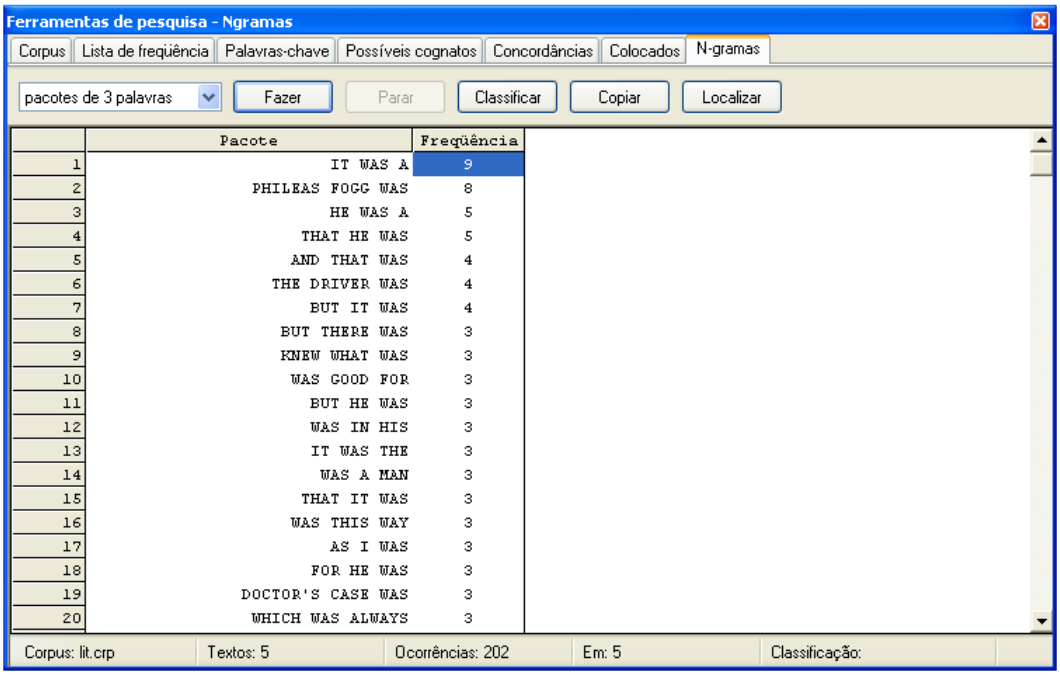
ANEXO 32 – Ferramentas de pesquisa – Concordâncias



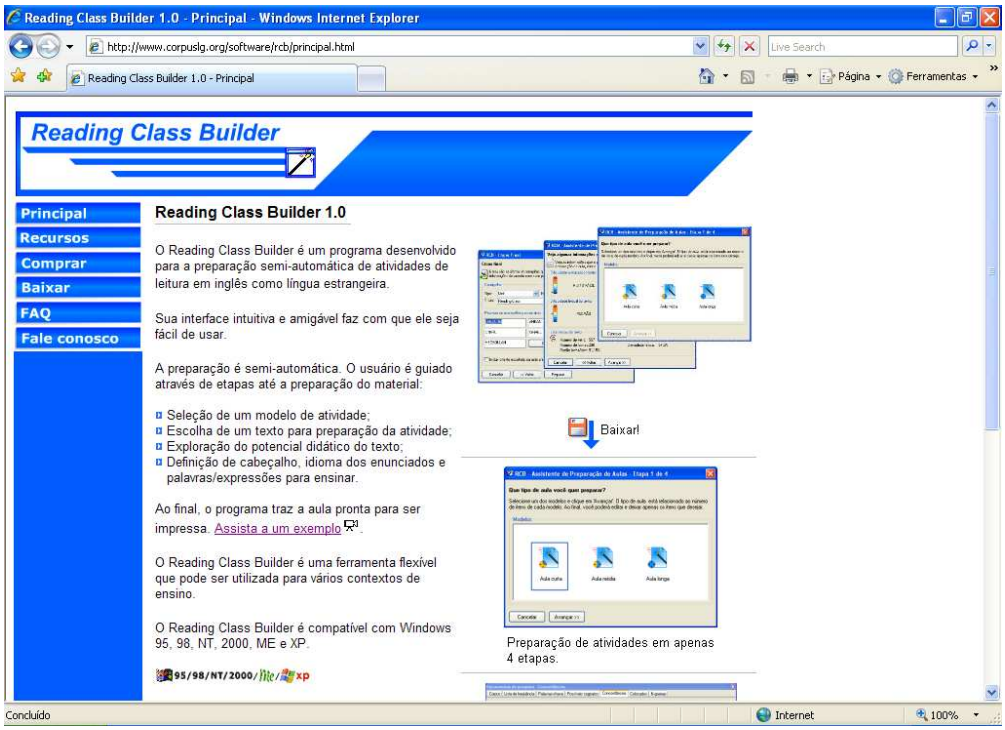
ANEXO 33 – Ferramentas de pesquisa – Colocados



ANEXO 34 – Ferramentas de pesquisa – N-gramas



ANEXO 35 – Página inicial do site criado para divulgação do *software*



Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo