Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica e de Computadores

Rui Pedro de Azevedo Venâncio

Geração de Pseudopalavras para Avaliação Linguística

Coimbra, Fevereiro 2018

UNIVERSIDADE DE COIMBRA

Geração de Pseudopalavras para Avaliação Linguística

Orientador:

Prof. Doutor Fernando Manuel dos Santos Perdigão

Júri:

Prof. Doutora Teresa Martinez dos Santos Gomes

Prof. Doutora Carla Alexandra Calado Lopes

Prof. Doutor Fernando Manuel dos Santos Perdigão

Coimbra, Fevereiro 2018

Agradecimentos

Quero agradecer, em primeiro lugar, ao Prof. Doutor Fernando Perdigão, pelos conhecimentos transmitidos, o apoio incansável e o empenho total para o sucesso deste projeto.

Ao Jorge Proença pela disponibilidade, ajuda e à-vontade que demonstrou durante a realização deste trabalho.

A toda a minha família, em especial, ao meu pai, mãe e irmã, aos meus tios e aos meus avós, pela educação que me deram e por todo o apoio prestado, ao longo de todos estes anos.

Gostaria de agradecer também aos eternos LedZener, pelas memórias e aventuras nesta cidade.

A todos os meus amigos de infância e aos que conheci nesta cidade, também foram fulcrais direta ou indiretamente.

À memória dos meus avós paternos, por tudo o que me ensinaram e ajudaram, nunca serão esquecidos.

Resumo

A capacidade de leitura é um aspeto importante durante a aprendizagem da língua e é adquirida, geralmente, em crianças com idade pré-escolar. A avaliação do desempenho da leitura pode ser feita através de diferentes formas, tanto na leitura de texto como na leitura de pseudopalavras.

Pseudopalavras são palavras que não existem no léxico, mas que são pronunciáveis, uma vez que seguem as regras fonotáticas de uma determinada língua.

A leitura de pseudopalavras permite avaliar se as regras de conversão de texto para fala (consciência fonológica) estão bem assimiladas, já que o leitor não tem qualquer tipo de familiaridade com as pseudopalavras que está a ler. Assim é possível avaliar o desempenho na leitura, de modo a, por exemplo, prevenir futuros défices fonológicos. Assim, a criação de um sistema que seja capaz de gerar pseudopalavras segundo determinados critérios e especificações é importante em diferentes áreas da linguística.

Este trabalho aborda o problema da geração de pseudopalavras, propondo algoritmos para a sua concretização. Os algoritmos são baseados em concatenação de sílabas, com a condição de todos os pares de sílabas das pseudopalavras existirem na língua, neste caso, o Português Europeu.

Este projeto também pressupõe a criação de um corpus lexical e um *software* fácil de utilizar e capaz de mostrar as pseudopalavras geradas e medidas adicionais relacionadas com proximidade lexical. Os algoritmos e o consequente interface com o utilizador foram desenvolvidos em *MATLAB*.

Abstract

Reading ability is an important aspect of a language learning and is generally acquired in preschool children. The evaluation of reading performance can be evaluated through the reading of texts or by reading pseudowords.

Pseudowords are words that respect the phonotactic restrictions of a language, in this case European Portuguese, and can be read but don't exist in lexicon.

When reading pseudowords it's possible to evaluate if the rules of conversion from text to speech (phonological awareness) are well assimilated because the reader doesn't have any kind of familiarity with it. Thus it's possible to evaluate reading performance in order to, for example, prevent future phonological deficits. So it is important to have a system that can be able to generate pseudowords, according to certain criterias and specifications, is important in different areas of linguistics.

This thesis describes the difficulty of generating pseudowords and proposes algorithms. The algorithms are based on concatenation of syllables, with the restriction that all pairs of syllables of pseudowords exist in the language, in this case European Portuguese.

This project also presupposes the creation of a lexical corpora and an easy-to-use software capable of presenting the generated pseudowords and additional measures related to lexical proximity. The algorithms and the user interface were developed in MATLAB.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

IPF	Índice de probabilidade funológica
OLD20	Ortographic Levenshtein Distance 20
Ned1	Neighbors at edit distance 1
pt-PT	Português Europeu
Npp	Número de pseudopalavras
PP	Pseudopalavras
Dist1sub	Vizinhos de distância 1 só por substituição
Dist1	Vizinhos de distância 1
Dist2	Vizinhos de distância 2
Dist3	Vizinhos de distância 3
MEX	MATLAB executable
Dists	Vizinhos de diferentes distãncias
Lists	Lista dos 20 vizinhos mais próximos
HTML	HyperText Markup Language

Capítulo 1

Introdução

As crianças, antes de entrarem para o primeiro ciclo, já são capazes de diferenciar e manipular sílabas, no entanto a sua consciência fonológica só é melhorada com a entrada das crianças no 1º ciclo [3]. Contudo é importante que as crianças aprendam a ler e a escrever antes da entrada no primeiro ano de escolaridade, de modo a prevenir possíveis défices no futuro, na associação grafema-fonema (leitura) e fonema-grafema (escrita). [1]

Na idade escolar as crianças aprendem os valores fonológicos das letras, ou seja, os sons que as mesmas representam, uma vez que a letra pode ter diferentes sons, ajudando deste modo na identificação das letras e na leitura das palavras. É através do som das palavras que a criança aprende a identificar a semelhança e a diferença entre as mesmas.

Esta dissertação é muito pertinente, já que com a criação de pseudopalavras, seguindo as regras fonotáticas da língua portuguesa, é possível que um professor, por exemplo, possa avaliar o desempenho das crianças através da leitura das mesmas.

Pseudopalavras são palavras que não existem no léxico da língua e não têm qualquer significado, mas são pronunciáveis mais ou menos sem ambiguidade, segundo as regras fonotáticas.

O propósito desta dissertação é ter inicialmente um corpus lexical suficientemente grande e tratado de maneira a que posteriormente seja criado um *software* eficiente e facilmente utilizável, que gera pseudopalavras e que apresenta cálculos lexicais e/ou métricas das mesmas, consoante o interesse do utilizador.

Este projeto vem preencher um vazio, que existe até então, em relação ao português europeu, na medida em que existe a necessidade de ferramentas que permitam a geração de pseudopalavras, já que neste momento só existem (e poucos) sistemas geradores de pseudopalavras noutras línguas, que não o Português Europeu.

1.1 Motivação

Uma das motivações para este projeto provém da implementação das Metas Curriculares de Português do Ensino Básico, que definem diferentes objetivos para diferentes anos de escolaridade, sendo um deles a avaliação da leitura, nomeadamente, a leitura de pseudo-

palavras.

Outra motivação deste projeto recaiu sobre a necessidade de um sistema gerador de pseudopalavras em Português Europeu (pt-PT) facilmente utilizável e adaptado à nossa língua materna, de modo a que qualquer investigador, professor ou utilizador comum, sinta a necessidade de utilizar pseudopalavras para o seu estudo.

Até à data da dissertação não existe nenhum gerador de pseudopalavras a nível do Português Europeu e muito poucos a nível mundial, devido à sua elevada complexidade.

1.2 Objetivos

Tendo sempre em mente a motivação deste trabalho, os objetivos passaram pela criação de um corpus lexical e de algoritmos para flexão de palavras, divisão silábica, pronunciação, e extração de características de proximidade lexical e fonológica.

Criação de um sistema gerador de pseudopalavras em português Europeu, o que não existe até ao momento, fácil de utilizar por parte do utilizador, com diferentes especificações e diferentes métodos de geração através da combinação de sílabas, tendo em conta os devidos pesos a nível da frequência de ocorrência de cada par de sílabas na língua.

1.3 Estrutura da dissertação

Esta dissertação está devidamente organizada e dividida em seis capítulos. O capítulo 1 dá a conhecer e a introduzir o trabalho desenvolvido durante a realização desta dissertação e as diferentes áreas que esta dissertação abrange. Indica também quais os objetivos e a motivação para o desenvolvimento da mesma.

O capítulo 2 descreve que sistemas geradores de pseudopalavras existem, com uma breve descrição dos mesmos e que estudos existem até ao momento da realização desta dissertação, o que equivale ao estado da arte deste projeto.

No capítulo 3 é descrito como obtivemos o nosso corpus lexical e de que projetos e de que formas e/ou técnicas utilizamos para a extração do mesmo.

O capítulo 4 faz uma abordagem detalhada aos algoritmos que tivemos de fazer de maneira a termos as tão desejadas pseudopalavras e os seu cálculos lexicais.

No capítulo 5 são visualizados os resultados obtidos com os nossos algoritmos, em termos de pseudopalavras, tempos de processamento, entre outras coisas.

Por fim no capítulo 6 temos as conclusões da dissertação, as falhas e os possíveis melhoramentos.

Capítulo 2

Trabalhos relacionados

Neste capítulo serão abordados, primeiramente estudos que analisam o formato das pseudopalavras e a sua importância para prevenir défices fonológicos e de seguida que softwares foram explorados na realização desta dissertação.

2.1 Estudos sobre Pseudopalavras

Um estudo elaborado, no âmbito da dissertação de mestrado em Ciências da Linguagem, por [**?**] demonstra que "as dificuldades no processamento fonológico em crianças com dislexia tendem a ser reproduzidas de forma mais consistente em provas com pseudopalavras linguisticamente motivadas, pelo facto de o processamento fonológico da crianças disléxica se encontrar perturbado e ser melhor avaliado por provas com pseudopalavras".

O estudo analisou os resultados de quatro provas que usou pseudopalavras linguisticamente motivadas: "uma prova de discriminação auditiva, uma prova de leitura e duas provas de repetição (uma com pseudopalavras fonologicamente motivadas e outra com pseudopalavras morfologicamente motivadas)". Os resultados destes testes foram bastante positivos e significativos, tendo-se verificado uma "correlação positiva entre o desempenho dos sujeitos e o grau de Índice de probabilidade funológica (IPF) associado às pseudopalavras". Também é verificado que "quanto maior o índice de probabilidade fonológica (IPF) maior o número de respostas corretas e melhor o desempenho dos pacientes. Tal, valida, a utilização deste tipo de instrumento no diagnóstico desta patologia."

Com base nestes resultados é validada a importância da existência de algoritmos que geram pseudopalavras com um elevado IPF para diagnóstico de défices fonológicos.

2.2 WordGen

O WordGen, é um *software* gerador de pseudopalavras e de palavras, que tem como base de dados, o CELEX (holandês,inglês,alemão e cirílico) e o Lexique (francês). Este sistema permite a geração de palavras/pseudopalavras em diversas línguas entre elas, o holandês, o inglês, o alemão e o francês; através da combinação de restrições linguísticas.

2.3 MCWord

MCWord é um gerador de pseudopalavras e/ou palavras em inglês, que tem como base de dados o CELEX, permite a obtenção de métricas lexicais, permite gerar pseudopalavras com diferentes graus de aproximação inglesa. Os resultados, mesmo que inglês ficaram muito aquém do esperado.

2.4 Wuggy

O WUGGY [2] é um *software* gerador de pseudopalavras que veio melhorar os métodos existentes, até então (2010). Permite a geração de pseudopalavras polissilábicas que obedeçam a determinadas restrições fonotáticas de diferentes línguas. Recebe como argumentos de entrada, ou uma palavra ou uma pseudopalavra, de modo a retornar ou palavras ou pseudopalavras ou ambos, através da estrutura silábica e das frequências de transição, isto sem precisar de percorrer a lista de todas as possíveis combinações.

O programa está disponível, em holandês, inglês, alemão, francês, espanhol, sérvio, basco e vietnamita, com a possibilidade de ser expandido para outras línguas e funciona com base num dicionário de palavras divididas em sílabas.

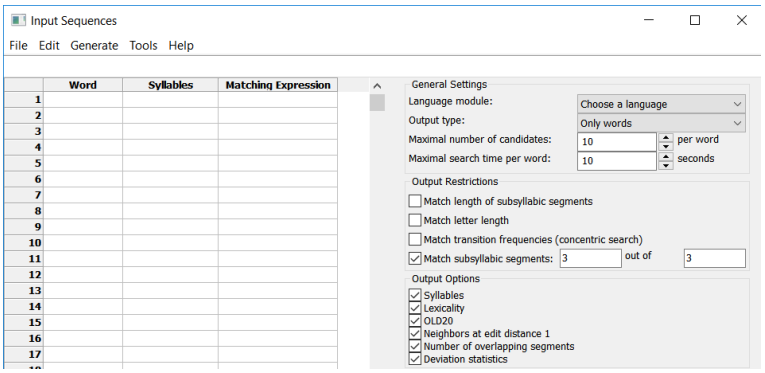

Figura 2.1: Janela do Wuggy.

Em termos de especificações gerais, começamos por escolher a língua em que queremos trabalhar e de seguida escolhemos se queremos gerar palavras e/ou pseudopalavras. Escolhemos também a quantidade de palavras/pseudopalavras que queremos gerar e o tempo que queremos esperar de modo a obtermos a solução pretendida.

De seguida escrevemos a(s) palavra(s) desejada(s), a sua divisão silábica e o *matching expression* desejado. (opcional).

Em termos de restriçções o Wuggy apresenta as seguintes:

Match length of subsyllabic segments: escolhendo esta restrição, o *output* de palavras vai apresentar a mesma estrutura silábica da(s) palavra(s) introduzida, ou seja, se a(s) palavra(s) introduzidas forem, por exemplo, consoante,vogal-consoante,vogal, à saída te-

remos a mesma estrutura silábica.

Match letter length: com esta restrição, teremos à saída correspondências com o mesmo número de letras da(s) palavra(s) à entrada do programa.

Match transition frequencies (concentric search).

Match subsyllabic segments: corresponde ao quão parecido queremos uma lista de palavras com a(s) palavra(s) originais introduzidas, por predefinição é 2 em 3 (2/3), mas podemos alterar consoante o que nós queremos. Se, por exemplo, queremos uma relação 2/3 teremos palavras muito semelhantes mas que não são facilmente identificadas como correlacionadas com uma dada palavra.

2.4.1 Informação das palavras/pseudopalavras

No Wuggy podemos saber que informações queremos retirar das palavras que foram geradas, através das opções seguintes.

Syllables: teremos uma coluna em que as palavras que foram geradas através de outra(s) estão divididas em sílabas (divisão silábica).

Lexicality: esta opção permite saber se um resultado é palavra (w) do inglês *word* ou pseudopalavra (n) do inglês *nonword*.

Ortographic Levenshtein Distance 20 (OLD20): a média da distância de *Levenshtein* dos 20 vizinhos/candidatos mais próximos de uma dada palavra. A distância de *Levenshtein* é equivalente ao número mínimo de operações necessárias para transformar uma palavra noutra, ou seja é igual ao número mínimo de inserções, apagamentos e substituições para uma dada *string* a obtenção da outra. Escolhendo esta opção vai tornar a geração de palavras mais lentas, já que, é necessário calcular a distância de Levenshtein entre a(s) palavra(s) candidata e as vinte palavras mais semelhantes no léxico. Se o valor for baixo do OLD20, quer dizer que podem ser obtidas várias palavras alterando apenas uma letra. O OLD20_diff obtém a diferença entre a palavra gerada e a palavra escolhida.

Neighbors at edit distance 1 (Ned1): *neighbors at edit distance 1* são o número de palavras do léxico que se podem obter a partir de cada da(s) palavra(s) candidata(s) substituindo, apagando ou inserindo uma letra. Esta opção também diminui a rapidez do sistema.

Number of overlapping segments: Se tivermos o *Match subsyllabic segments* com um visto, quer dizer que nesta coluna vamos ter a fração introduzida (por predefinição é 2/3), caso não esteja escolhida nenhuma vai mostrar o seu rácio de parecença com a palavra original.

Por fim, *Deviation Statistics*: que mostra a maior diferença *in transition frequencies* entre as sílabas na sequência gerada e na sequência original introduzida. Se por exemplo, o valor for 14, quer dizer que existem mais 14 palavras com a mesma transição. Esta

opção também mostra a soma de todas as variações de frequência de transição (valores absolutos). E por fim também tem uma coluna que mostra o sítio onde existe o maior desvio de transição.

Durante a realização deste trabalho foi possível colocar o *Wuggy* a gerar pseudopalavras em português, através da introdução de um ficheiro de léxico de antigos projetos, em português. Num pequeno código foi feita a introdução de letras acentuadas, de letras que podiam ser seguidas de outras e letras duplamente acentuadas, em *Python*.

Os resultados foram geradas (quase) sempre a mesma sequência de pseudopalavras se mantivermos as mesmas especificações, ou seja, é determinístico. Os resultados através deste *software* ficaram aquém do esperado o que motivou a realização de algoritmos de pseudopalavras em pt-PT.

O Wuggy apresenta como desvantagens, gerar (quase) sempre a mesma sequência de palavras se mantivermos as mesmas especificações, ou seja, é determinístico. E muitas pseudopalavras não eram possíveis de ler, o que motivou o desenvolviment de novos algoritmos geradores de pseudopalavras em pt-PT.

Capítulo 3

Corpus lexical

Neste capítulo é abordada a forma como foram extraídas as palavras que formam a nossa base de dados lexical, de onde foram extraídas e que técnicas foram utilizadas.

Para a formação de pseudopalavras é necessário ter uma boa base de dados lexical com diferentes tipos de informação, isto é o corpus lexical tem que ser o mais rico possível. O primeiro ficheiro que servirá como base aos nossos sistemas vai conter um vocabulário com a grande maioria das palavras do léxico, incluindo flexões de lemas e a respetiva divisão silábica de todas essas palavras.

A esse ficheiro de léxico derivam diferentes ficheiros que serão, por sua vez, a espinha dorsal para todo este projeto. Esses processo serão descrito nas secções seguintes.

Com a necessidade de uma base de dados em pt-PT, uma das escolhas recaiu para o projeto P-Pal, que depois de comparada com outras base de dados pareceu a mais completa.

3.1 P-Pal

O primeiro passo deste projeto foi a extração de todas as formas provenientes do P-Pal. O P-Pal, também conhecido como Procura Palavras, é uma aplicação *Web* desenvolvida pela Universidade do Minho e baseia-se num *corpus* de 227 milhões de palavras. A construção da base de dados de suporte ao P-Pal é proveniente de textos jornalísticos, literários, género técnico-científico e didático e ainda o género miscelânea. A percentagem de contribuição dos diferentes géneros e diferentes léxicos encontra-se na figura 3.1.

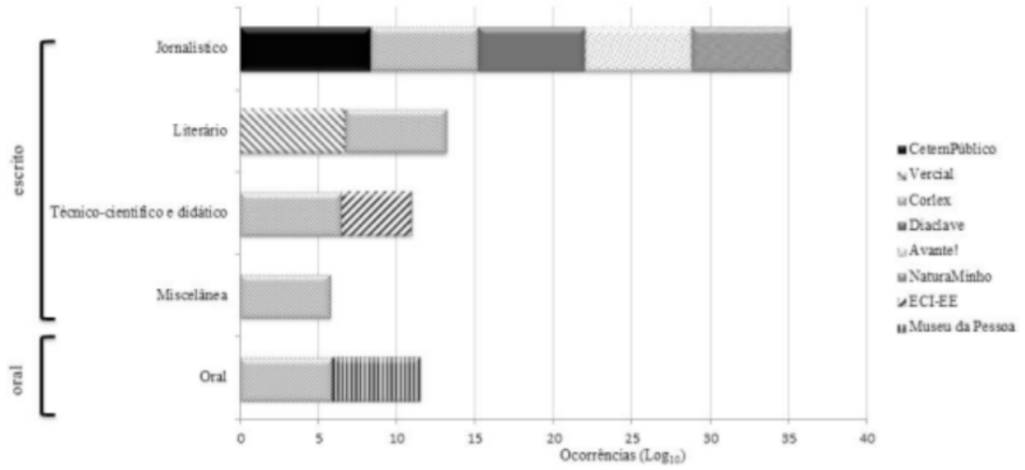

Figura 3.1: Distribuição dos *corpora* do P-Pal por género e tipo linguísticos.

Uma das técnicas utilizadas, de modo, a podermos expandir ainda mais o nosso léxico, foi a utilização do software "Lince", que se encontra analisado em detalhe na secção seguinte, para "tratar" o léxico, de maneira a termos um ficheiro mais rico em léxico que contivesse ambos os acordos ortográficos. Ao construirmos uma base de dados cada vez maior faz com que possamos ter cada vez mais sílabas e mais encontros silábicos, o que no seu conjunto origina mais pseudopalavras.

3.2 Lince

O Lince é um software, utilizado neste projeto, proveniente do portal da língua portuguesa, que permite converter ficheiros de texto para o novo acordo ortográfico (AO90).

3.3 Técnicas para tratamento do léxico

3.3.1 Estrangeirismos

Estrangeirismos são palavras provenientes de outras línguas que são utilizadas e/ou empregadas na nossa língua, como por exemplo, surf e jazz, através de certas combinações de sílabas que na nossa língua muitas vezes não existem. Apesar de alguns estrangeirismos já serem aportuguesados nem todos se pode dizer o mesmo.

Um dos objetivos deste trabalho passava pelo tratamento devido da base de dados, depois de recolhidas as palavras provenientes do P-Pal e do CETEMPúblico, de modo, a que não existam nenhuns estrangeirismos, ou no pior caso, o menor número possível dos mesmos. Como seria de esperar é impossível afirmar que não exista pelo menos um estrangeirismo, contudo foram utilizadas diferentes técnicas para garantir que esse número tende para zero.

Muitos estrangeirismos foram identificados pela falta de divisão silábica das palavras provenientes do P-Pal, já noutros não foi assim tão simples e assim foi necessário fazer uma procura manual através de técnicas de expressões regulares, por exemplo. Os outros estrangeirismos foram identificados através de alguns dos seguintes critérios:

- Palavra que contivessem "k", "y"e "w", exemplo de "ketchup".
- Dígrafos em "cc", "dd", "ff", "gg", "ll", "pp", "tt", "zz", exemplo de "Garrett".
- Dígrafos derivados do inglês: "th", "sh", 'oo", exemplo "theme".
- Composições de letras não existentes no português, tais como "ght", "gns", exemplo "light".
- Palavras terminadas em "ing", "ingle", "ium", "n"exemplo "jingle".
- Palavras começadas com "up", exemplo "upgrade".
- Palavras que contivessem "diesel", "design", "hertz", exemplo biodiesel.

É de notar que nem todas as palavras que contivessem uma certa sequência de letras, enumeradas ou não anteriormente, eram, efetivamente, estrangeirismos, ja que todas as palavras apagadas do léxico foram revistas uma a uma.

Foi feito um algoritmo para identificar possíveis sílabas que só aconteciam uma vez em todo o léxico, o que deu entre 400 a 500 sílabas. De seguida, foi feita uma verificação manual e concluiu-se que muitas são estrangeirismos, outras tinham a divisão silábica mal feita. As sílabas que faziam parte de estrangeirismos, resultou num apagamento de todos esses estrangeirismos.

As palavras que continham sílabas com frequência igual a 1 que foram apagadas foram: "bad", "bies", "blen", "blé", "blés", "clai", "clips", "corn", "cors", "cto", "cue", "cues", "céns", "dho", "proust", "dplo", "due", "fif", "fles", "gangs", "gies", "gins", "glam", "gne", "gour", "greens", "grom", "gés", "hot", "ib", "il", "jé", "leib", "lud", "lús", "ners", "nietz", "nox", "ohm", "ohms", "pgra", "poufs", "poule", "prez", "reau", "rries", "rrés", "sign", "soc", "spiel", "stend", "ssier", "sta", "teaus", "vs", "tris", "zló", "zón"e "ción". Nota: caso anteriormente não se tivesse apagado muitos estrangeirismos através de palavras que contivessem uma certa sequência de letras, encontraríamos muitas mais sílabas com frequência igual a 1 em termos de ocorrência na língua que seriam para apagar.

3.3.2 Siglas

As siglas também foram retiradas, caso de por exemplo, "ADN", já que, não vão beneficiar em nada na formação de PP.

3.3.3 Hífen

Foram retiradas todas as palavras com hífen da nossa base de dados até então, mas não apagadas, isto é, as palavras foram separadas em duas diferentes pelo elemento em comum entre elas, o hífen, de modo a serem comparadas individualmente com cada uma das palavras do léxico. Se essas palavras não forem estrangeirismos e caso não existam ainda, são adicionadas, de modo a enriquecerem a nossa base de dados. Por exemplo, a palavra "escolas-piloto"foi separada em "escolas"e "piloto". De seguida foi comparada com o léxico até então, de modo a verificar se existe "escolas"e "piloto", se não existir "escolas", esta é adicionada e se não existir por sua vez "piloto"também é igualmente adicionada.

3.3.4 CETEMPúblico

O CETEMPúblico é um *corpus* composto por palavras do português europeu, provenientes do jornal Público, que conta com cerca de 180 milhões de palavras.()

Depois de termos a base de dados (supostamente) completa verificamos que a mesma ainda não estava rica o suficiente. Por isso, recorreu-se a outra base de dados proveniente do CETEMPúblico, utilizada em projetos anteriores que decorreram no laboratório, com as palavra mais frequentes do CETEMPúblico no laboratório e comparou-se as duas BD. Verificou-se que, essencialmente, não haviam nomes próprios, tais como nomes de pessoas, cidades , países e assim estes foram adicionados, mas o cuidado de não serem adicionados, estrangeirismos, palavras com hífen, siglas , etc. e por isso esta BD também passou pelos processos anteriores antes de ser adicionada à base de dados lexical até então. Depois de todos estes processos concluímos a nossa BD lexical.

3.4 Base de dados lexical

Foi possível obter depois de todos estes processos e técnicas, um ficheiro final com o léxico e a sua divisão silábica. Este ficheiro tem o nome Dicionário_div_PPAL_v7.txt tem as seguintes características:

1. Está ordenado por ordem alfabética;

2. Contém 194,034 palavras do léxico e a correspondente divisão silábica;

A base de dados deste projeto não ficou por aqui, pois sentiu-se a necessidade de ter mais informações que facilitariam a geração de PP. Como é o caso da necessidade de um vocabulário de sílabas e bigramas de sílabas. Bigramas de sílabas são definidos como pares de sílabas que ocorrem no léxico. Este ficheiro conteria diversa informação acerca de todos os pares de sílabas que será descrita nas secções seguintes. As bigramas vão facilitar à criação de pseudopalavras, já que as pseudopalavras serão formadas por sílabas que tenham relações entre si.

3.5 Vocabulários e bigramas

Depois de criado o nosso ficheiro de léxico, foi criada uma função, em *MATLAB*, Cria_voc_bigrama que recebe como parâmetros de entrada todas as palavras do nosso léxico e a respetiva divisão silábica, de modo a criar e a escrever para quatro ficheiros diferentes, com diferentes números colunas, de linhas e essencialmente diferentes conteúdos.

3.5.1 Ficheiros criados a partir da base de dados lexical

É escrito para um primeiro ficheiro, word_v2.txt todas as palavras do léxico, apenas, de modo a ser posteriormente usado para cálculos lexicais que estão feitos em C++. Apresenta as seguintes características:

1. Está ordenado por ordem alfabética;
2. Contém todas as palavras do léxico (apenas);
3. Ficheiro com 1 colunas e 194,034 linhas.

Outro dos ficheiros, de nome vocabulario_v4.txt, tem apenas uma coluna e contém um vocabulário de sílabas, onde são escritas todas as sílabas únicas existentes no léxico. Para isso, foram divididas todas as sílabas das divisões silábicas e foram adicionadas todas as sílabas não repetidas ao ficheiro. Apresenta as seguintes características:

1. Está ordenado por ordem alfabética;
2. Contém todas as sílabas (únicas);
3. Ficheiro com 1 coluna e 2749 linhas.

Outro, vocabulario_v4_nlex.txt, é semelhante ao enunciado anteriormente, ou seja, é um vocabulário de sílabas, mas neste caso difere do anterior, já que as sílabas não podem ser sozinhas palavras, como por exemplo, não existem as palavras “tem”, “a”, entre outros. Tem as seguintes características:

1. Está ordenado por ordem alfabética;
2. Contém sílabas que sozinhas não formam palavra, ”abs”, ”guim”, ”fer”(Não contém palavras que sozinhas formam palavra, exemplo, ”tem”, ”a”, etc);
3. Não tem palavras/sílabas, que comecem por ”rr”, ”ç”, ”nh”, ”ss”;
4. Não tem palavras/sílabas, que acabem com uma das seguintes consoantes, ”c”, ”d”, ”f”, ”g”, ”j”, ”k”, ”n”, ”p”, ”q”, ”t”, ”v”, ”w”, ”y”.
5. Ficheiro com 1 colunas e 1596 linhas.

E por fim mas não menos importante, o ficheiro denominado de bigrama de sílabas (todas_big_v7.txt) que vai ter um papel crucial na criação das pseudopalavras dentro dos algoritmos. Este ficheiro contém na 1ª coluna todos os pares de sílabas possíveis, tome como exemplo, "a-ba", "por-ta", "ta-ção".

A segunda e a terceira coluna, correspondem ao índice no vocabulário completo de sílabas (vocabulario_v4.txt), da primeira e da segunda sílaba, respetivamente, do par de sílabas. Por exemplo no par de sílabas "a-ba", na segunda coluna e terceira teremos 1 e 25 respetivamente. 1 porque "a" é a primeira sílaba no vocabulário e 25 porque "ba"tem o índice 25 nesse vocabulário-

A quarta coluna indica se a primeira sílaba pode ser de início através de valores lógicos e a quinta coluna indica se a última sílaba pode ser de fim através de também valores lógicos. A obtenção do valor lógico destas duas colunas, proveio da verificação em todo o léxico se alguma vez determinada sílaba inicial do par começou ou se a sílaba final foi fim de palavra, respetivamente.

A sexta coluna também tem valores lógicos caso a primeira sílaba do par tenho algum acento e a sétima coluna indica a mesma coisa mas para a última sílaba do par.

O ficheiro está ordenado de forma decrescente pela coluna seguinte, que é neste caso a oitava, que contém o número de ocorrências de cada par de sílabas no léxico.

Por fim as três últimas colunas, indicam o número de ocorrências de cada par de sílabas como início, meio e fim, respetivamente, no léxico. Esta verificação foi feita para cada par de sílabas, quantas vezes ocorrem como início de palavra, no meio e no fim. Um resumo deste ficheiro é apresentando de seguida:

1. Contém todas as combinações de pares de sílabas possíveis;
2. Informação acerca dos índices das sílabas no vocabulário_v4.txt;
3. Indica se a primeira sílaba do par pode ser de início de pseudopalavra e a última se pode ser de fim;
4. Indica se a primeira e a última sílaba do par são acentuadas ou não;
5. Está ordenado de forma decrescente pelo número de ocorrências de cada par de sílabas;
6. Número de ocorrências no início, meio e fim das palavras;
7. Ficheiro com 11 colunas e 52,319 linhas.

Capítulo 4

Geração de Pseudopalavras

Neste capítulo são abordados em detalhe os algoritmos que foram desenvolvidos para a geração de PP, nomeadamente os algoritmos principais e os auxiliares.

4.1 Algoritmos principais para a geração de PP

São chamados de algoritmos principais aos dois métodos diferentes de gerar pseudopalavras, "Gerador de PP de 1-10 sílabas"(gera_pp) e o "Palavra Protótipo"(palavra_prot).

4.1.1 Gerador de PP de 1-10 sílabas

O algoritmo "Gerador de PP de 1-10 sílabas"(gera_pp), é uma função que gera pseudopalavras através da combinação aleatória de pares de sílabas e verifica se essas existem no léxico, de modo a poder retornar o número de pseudopalavras desejado ou tantas quanto possível. Como o nome indica, gera no mínimo PP de 1 sílaba e no máximo PP de 10 sílabas. O maior número encontrado de sílabas numa palavra, em todas as palavras da nossa BD lexical, foi 10, daí esse ser o máximo possível que uma pseudopalavra pode ter em termos de número de sílabas.

O ficheiro de bigramas de sílabas terá um papel crucial na formação das pseudopalavras, uma vez que contém informação detalhada de cada par de sílabas, o que permitirá saber se os pares de sílabas são válidos nas diferentes posições das palavras.

Consoante o número de sílabas desejado, são procurados pares de sílabas nas diferentes posições de cada pseudopalavra cada uma com diferentes valores. Se o número de sílabas for igual a 2, teremos 4*Número de pseudopalavras (Npp) hipóteses (no máximo), já que a maior parte destes, provavelmente, serão palavras do léxico o que resultará em muitos pares de sílabas descartados porque já são por natureza palavras do léxico. Se o número de sílabas for igual a 3, teremos na 1ª posição 2*Npp hipóteses e na última posição 1 hipótese. Caso o valor seja superior a 3, na 1ª posição teremos Npp, na 2ª posição 2 e na última 1. Depois de vários testes, estas hipóteses foram as que retornaram melhores PP.

Inicialmente é guardada numa variável valores lógicos com o tamanho da bigrama. Esses valores vão sendo verdadeiros ou falsos, consoante se um par de sílabas pode ou não acontecer.

O algoritmo vai de par em par de sílabas. Começando pela 1ª posição são verificadas que pares de sílabas são de início e se este par de de sílabas acontece pelo menos uma vez como início de palavra.

Há outra verificação a nível da acentuação, que caso as pseudopalavras desejadas tenham mais de 3 sílabas, a 1ª sílaba do primeiro par de sílabas não pode ser acentuado. Depois este processo vai se repetindo, garantido que a penúltima sílaba nunca seja acentuada. E no último par de sílabas, tem que ser garantida que esse par pode ser de fim de palavra e aconteça pelo menos uma vez como fim de palavra.

O processo a seguir é feito através da função datasampleMex que está implementada em C++, o que torna bastante eficiente esta procura, que por sua vez está interligado com o *MATLAB*, já que o *MATLAB* tem ferramentas que permite chamar funções compiladas em C++ e usá-las como se deste se tratassem. As rotinas especiais feitas para serem executadas e chamadas pelo *MATLAB* são denominadas de *MEX Files*. É uma função que retorna observações de uma amostra segundo as suas probabilidades (sem reposição). A seleção de sílabas é feita através de alguma aleatoriedade, já que, os algoritmos estão feitos de maneira a que as sílabas escolhidas dependam do seu valor de ocorrência, no início, meio ou fim de cada par de sílabas correspondente. O objetivo deste método é garantir que surjam, com mais frequência, nas PP, sílabas que geralmente são início de palavras, meio e fim nas palavras lexicais. Aqui são retirados o número de amostras indicado anteriormente, que está dependente do número de sílabas desejado e da posição da palavra.

Depois deste processo armazena-se numa variável os índices das sílabas (no vocabulário de sílabas) que cumpram todos os critérios enunciados. De seguida são transformados estes índices para, efetivamente, sílabas através duma função contrária à anteriormente indicada de nome int2sil. Estas sílabas são armazenadas numa variável que tem colunas com igual número ao número de sílabas da palavra protótipo.

Por fim são feitas várias verificações às PP, de modo a garantir que nenhuma PP é palavra. Caso aconteça alguma destas verificações, existe uma variável binária se indica se a pseudopalavra é palavra do léxico ou não:

1. Começa por pesquisar se a pseudopalavra existe no léxico;
2. Se a pseudopalavra acabar com "s", verifica se existe no singular no léxico;
3. Se a pseudopalavra não acabar com "s", verifica se existe no plural no léxico;
4. Verifica se tem alguma letra acentuada, se tiver não pode acabar com, "z", "u", "us", "bi", "ci", "di", "fi", "gi", "gui", "hi", "ji", "li", "mi", "ni", "pi", "qui", "ri", "si", "ti", "vi", "xi", "zi", "bis", "cis, "dis", "fis", "gis", "guis", "his", "jis", "lis", "mis", "nis", "pis", "quis", "ris", "sis", "tis", "vis", "xis", "zis";

5. Verifica se acaba com "c", "d", "f", "g", "j", "k", "n", "p", "q","t","v","w","y";
6. E por fim verifica se começa com "rr", "ç", "nh", "ss".

De seguida, se o número de pseudopalavras pedido for maior que as pseudopalavras geradas, então retorna todas as pseudopalavras. Caso contrário retorna apenas o número de pseudopalavras desejado.

Por exemplo, se o número de sílabas introduzido for 3 e se se quiserem 5 PP, um resultado possível seria: "afica", "megaça", "hisparos", "mereda"e "multivou".

4.1.2 Palavra Protótipo

O algoritmo "Palavra Protótipo"é uma função que gera o Npp desejado ou tantas quanto possível, através de uma palavra protótipo, palavra essa que tem de existir no léxico.

Este algoritmo agrupa sílaba a sílaba até chegar ao nível de sílabas final isto contrariamente ao algoritmo exposto na subsecção 4.1.1 que vai buscar par a par de sílaba até chegar ao nível de sílabas menos um.

O algoritmo começa por verificar quantas sílabas tem a palavra introduzida através da informação contida no ficheiro da divisão silábica, contando quantos hífens a divisão silábica da palavra tem. Se não tiver nenhum, isto quer dizer que a palavra tem 1 sílaba e assim este algoritmo invoca um algoritmo auxiliar denominado de gera_pp_1sil que é descrito em 4.2.1. Se tiver apenas um hífen, a palavra tem 2 sílabas e assim este algoritmo invoca o algoritmo palavra_prot_2sil descrito em 4.2.2. Mais do que 1 hífen (mais do que 2 sílabas), a geração é toda feita neste algoritmo. Todos os ficheiros que temos como espinha dorsal dos algoritmos, neste algoritmo vão ser todos usados à exceção do vocabulario_nlex.txt.

O objetivo deste algoritmo é arranjar combinações diferentes na 1ª sílaba, mantendo as outras, combinações diferentes na 2ª sílaba mantendo as outras e por aí adiante.

Inicialmente, este algoritmo divide a palavra protótipo introduzida em sílabas e transforma essas sílabas em inteiros, inteiros esses que são os índices dessas sílabas no vocabulário de sílabas, isto resulta numa maior eficiência em comparação a trabalhar com *strings*. Para esse efeito foi criada uma função sil2int que faz isso mesmo.

De seguida é guardado numa variável valores lógicos com o tamanho da bigrama. Esses valores vão sendo verdadeiros ou falsos, consoante se um par de sílabas pode ou não acontecer. O algoritmo começa por ir de sílaba a sílaba. Começando pela 1ª sílaba são verificadas que sílabas são de início, não podem ser iguais à 1ª sílaba da palavra protótipo e que sílabas são pares de sílabas com a 2ª sílaba da palavra protótipo. Há outra verificação seguinte a nível da acentuação, caso nas outras sílabas já existam alguma sílaba acentuada, esta 1ª sílaba não pode acentuada e igualmente caso a palava protótipo tenha no mínimo 4 sílabas.

Esse processo é feito através da função datasamplemex, explicada no algoritmo em 4.1.1. Aqui são retirados o número de amostras igual ao número de pseudopalavras de-

sejado para cada posição (em termos de sílabas) da palavra protótipo. Neste caso para a primeira sílaba os valores de ocorrência serão originários da bigrama de sílabas que contém informação do número de ocorrências dum par de sílabas no início de palavra. Depois para a 2ª sílaba, o objetivo é encontrar sílabas que sejam consecutivas da 1ª sílaba da palavra protótipo, diferentes da 2ª sílaba da palavra protótipo e que sejam sílabas consecutivas da 3ª sílaba da palavra protótipo. Se a palavra for de 3 sílabas, estas sílabas não podem ser acentuadas porque são palavras graves. E caso, claro, existam já acentuações noutras sílabas. A probabilidade de ocorrência das sílabas que entrará neste caso são o número de ocorrências no meio. E este processo vai se repetindo até que se chegue à última sílaba, que terá que garantir que a penúltima sílaba seja par de sílaba com estas últimas e a probabilidade que aqui vai interessar é a sua ocorrência como fim de palavra.

Depois deste processo armazena-se numa variável os índices (no vocabulário de sílabas) correspondentes às sílabas que cumpram estes critérios. De seguida são transformados estes índices para, efetivamente, sílabas através duma função contrária à anteriormente indicada de nome int2sil. Estas sílabas são armazenadas numa variável que tem colunas com igual número ao número de sílabas da palavra protótipo.

Por fim são feitas várias verificações às PP, de modo a garantir que nenhuma PP é palavra. Caso aconteça alguma destas verificações, existe uma variável binária se indica se a pseudopalavra é palavra do léxico ou não:

1. Começa por pesquisar se a pseudopalavra existe no léxico;
2. Se a pseudopalavra acabar com "s", verifica se existe no singular no léxico;
3. Se a pseudopalavra não acabar com "s", verifica se existe no plural no léxico;
4. Verifica se tem alguma letra acentuada, se tiver não pode acabar com, "z", "u", "us", "bi", "ci", "di", "fi", "gi", "gui", "hi", "ji", "li", "mi", "ni", "pi", "qui", "ri", "si", "ti", "vi", "xi", "zi", "bis", "cis, "dis", "fis", "gis", "guis", "his", "jis", "lis", "mis", "nis", "pis", "quis", "ris", "sis", "tis", "vis", "xis", "zis";
5. Verifica se acaba com "c", "d", "f", "g", "j", "k", "n", "p", "q","t","v","w","y";
6. E por fim verifica se começa com "rr", "ç", "nh", "ss".

De seguida, se o número de pseudopalavras introduzido for maior que o número de pseudopalavras geradas, então retorna todas as pseudopalavras pela ordem da sílaba mudada. Caso contrário retorna as PP com o número de PP desejado mas não ordenadas pela sílaba alterada.

Por exemplo se a palavra protótipo introduzida for "abacate", teremos a-ba-ca-te, o que retornará por exemplo 4 pseudopalavras do género, "bilbacate"(mudou a 1ª sílaba), "acacate"(mudou a 2ª sílaba), "abanhaste"(mudou a 3 ª sílaba) e por fim "abacasso"(mudou a 4ª sílaba).

4.2 Algoritmos auxiliares para a geração de PP

Os algoritmos auxiliares são incorporados nos algoritmos principais e são invocados em determinados casos concretos passando despercebidos quando o utilizador está a gerar pseudopalavras. Estes algoritmos serão abordados e explicados devidamente nas subsecções, 4.2.1 e 4.2.2 .

4.2.1 Gerador de PPde 1 sílaba

A função que gera PP de 1 sílaba é denominada de gera_pp_1sil e pode ser invocada por qualquer um dos dois algoritmos principais, isto é, caso pretenda-se gerar PP de 1 sílaba tanto se pode escolher o algoritmo descrito em 4.1.1 como o algoritmo em 4.1.2, caso a escolha recaia em 1 sílaba ou na introdução duma palavra protótipo de 1 sílaba respetivamente. Tal processo está ilustrado na figura 4.1.

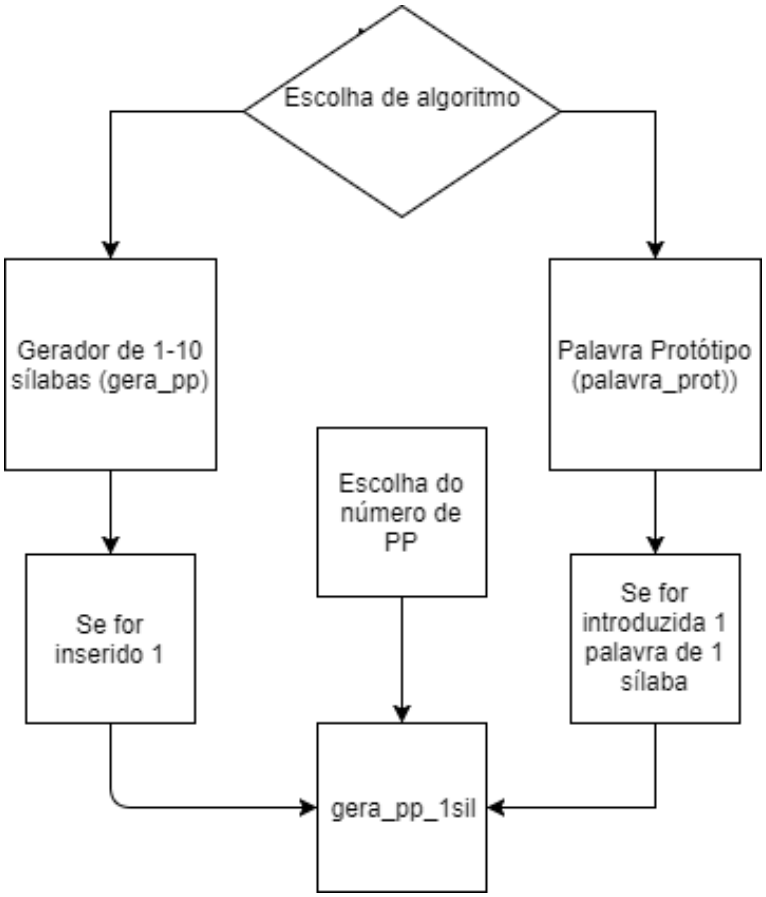

Figura 4.1: Fluxograma explicativo da invocação do algoritmo gera_pp_1sil.

Este algoritmo é de longe o mais simples e menos complexo em comparação com os algoritmos desenvolvidos e criados durante a realização desta dissertação, já que não faz praticamente nenhuma operação.

Inicialmente este algoritmo começa por baralhar o índice de todas as sílabas contidas no vocabulário e consoante o número pedido pelo utilizador retorna as primeiras sílabas

(palavras) correspondentes a esses mesmo índices, com um limite máximo de 1596 palavras. Este algoritmo é o único que não é necessário fazer nenhuma verificação se existem PP que sejam palavras porque um dos ficheiros que serve de base de dados para este algoritmo, denominado de vocabulario_v4_nlex.txt, está devidamente tratado conforme descrito em 3.5.1.

4.2.2 Palavra protótipo de 2 sílabas

A função de nome palavra_prot_2sil é invocada quando se introduz uma palavra protótipo de 2 sílabas no algoritmo em 4.1.2 e retorna PP de 2 sílabas variações da palavra protótipo como é possível ver pela figura 4.2.

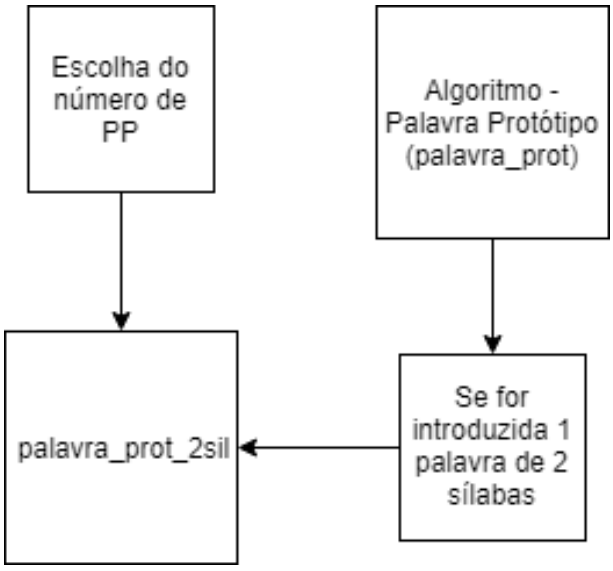

Figura 4.2: Fluxograma explicativo da invocação do algoritmo palavra_prot_2sil.

O algoritmo em questão é praticamente igual ao algoritmo em 4.1.2, com a diferença que neste caso não interessa se a 1 ª sílaba pode ser de início ou não e se a 2ª sílaba pode ser de fim ou não. De resto funciona tudo igual e são feitas as mesmas verificações, sem esquecer que a primeira sílaba não pode ser acentuada, já que as palavras graves não podem ser acentuadas.

4.3 Cálculos e informações lexicais

Para além dos algoritmos de geração de PP, existem algoritmos que fazem determinados cálculos e informações lexicais acerca das PP que serão posteriormente apresentados ao utilizador. Estes cálculos que serão apresentados já de seguida estão implementados através dos MATLAB executable (MEX) Files introduzidos em 4.1.1, o que torna esses cálculos muito eficientes e muito rápidos de obter.

OLD20 é um dos cálculos feitos que é descrito como a média da distância de *Levenshtein* dos 20 vizinhos/candidatos mais próximos de uma dada palavra. A distância de

Levenshtein é equivalente ao número mínimo de operações necessárias para transformar uma palavra noutra, ou seja é igual ao número mínimo de inserções, apagamentos e substituições para uma dada *string* a obtenção da outra.

Outra informação lexical são os Vizinhos de distância 1 só por substituição (Dist1sub), Vizinhos de distância 1 (Dist1), Vizinhos de distância 2 (Dist2) e Vizinhos de distância 3 (Dist3) que indicam quantos vizinhos (palavras do léxico) em comparação às pseudopalavras, existem de distância um só por substituição (Dist1sub); de distância um, dois e três através de substituições, apagamentos e inserções, Dist1, Dist2, Dist3 respetivamente.

Por fim , a última informação lexical indica quais os 20 vizinhos mais próximos das pseudopalavras.

4.4 Interface gráfico

Para o manuseamento dos algoritmos sentiu-se necessidade de ter um *interface* gráfico com os consequente botões, para a fácil manipulação e geração de PP. Como os algoritmos estão implementados em *MATLAB*, decidiu-se que uma solução barata e rápida para o mesmo passasse porque não num *interface* com o utilizador em *MATLAB*.

Os ficheiros que os algoritmos utilizam são logo carregados para memória quando o interface é aberto o que resulta numa melhor eficiência nos algoritmos e num menor tempo de espera de processamento.

4.4.1 Janela de apresentação

O interface com o utilizador foi desenvolvido em *MATLAB* através da ferramenta *Appdesigner*. Na janela inicial há uma breve descrição do sistema, procedente pela escolha do número de pseudopalavras, que permitirá (se possível) o número de pseudopalavras desejado. O interface assenta em dois algoritmos principais assim existe a possibilidade de escolha individual dos mesmos, para a geração algorítmica de PP. Consoante a escolha do algoritmo, diferentes resultados serão obtidos, isto é se a escolha for o "Gerador de pseudopalavras de 1-10 sílabas"o utilizador tem a opção de escolher de quantas sílabas quer que as pseudopalavras sejam formadas. Caso a escolha recaia no segundo algoritmo "Palavra Protótipo"é dada a possibilidade de introdução duma palavra, de forma a que as pseudopalavras sejam variações da mesma.

Depois de escolhidos os algoritmos e o número de pseudopalavras a gerar, existe um último ponto denominado de resultados. As PP e os consequentes resultados serão apresentados na tabela que inicialmente está por preencher. Antes de serem apresentados resultados é necessário que o utilizador escolha que informações quer das PP. Para isso há a possibilidade da obtenção de cálculos lexicais e/ou outras informações, relacionadas às PP. O primeiro parâmetro é denominado de OLD20 que é uma métrica de cálculo que já foi enunciada e explicada ao longo deste trabalho. O segundo parâmetro é denominado de Vizinhos de diferentes distãncias (Dists), que cria quatro colunas na tabela, a primeira Dist1sub, Dist1, **Dist 2!** (**Dist 2!**) e por fim **Dist 3!** (**Dist 3!**). Um último possível parâmetro de resultados, é o Lista dos 20 vizinhos mais próximos (Lists) que

consequentemente cria uma outra coluna na tabela.

Existem dois botões, um que se chama de "Gerar"que é onde é feita, efetivamente, a geração das pseudopalavra; o outro botão denominado de "Guardar como...", permite guardar a informação apresentada na tabela, para um ficheiro .txt ou .xlsx (excel), à escolha do utilizador.

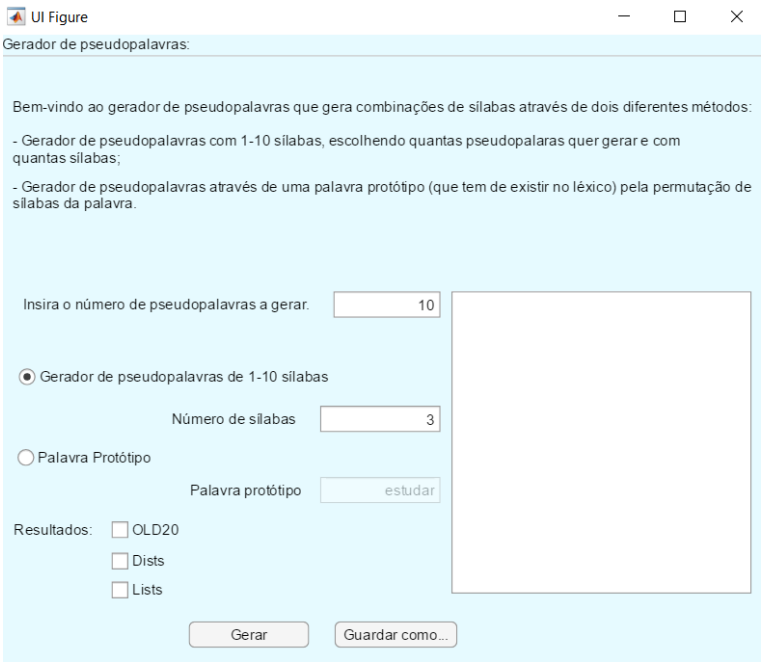

Figura 4.3: Janela de apresentação.

4.4.2 Parâmetros por definição

Como seria expectável e de modo a garantir uma melhor experiência com o utilizador, existem certos parâmetros por definição, caso seja o primeiro contacto com o software ou caso o utilizador não tenha muitos conhecimentos ou interesse na geração mais específica de PP.

Na inserção do número de PP, o valor predefinido é o número 10. O algoritmo selecionado por defeito é o primeiro algoritmo, definido como "Gerador de pseudopalavras de 1-10 sílabas", que por definição gera PP de 3 sílabas. Caso o utilizador mude para o algoritmo "Palavra Protótipo", a palavra protótipo por definição é "estudar". Inicialmente os "Resultados"não estão selecionados.

Capítulo 5

Resultados

Neste capítulo são apresentados possíveis resultados, aquando da experimentação e do contacto do *interface* com o utilizador e os consequentes resultados, consoante certas especificações.

Como já foi enunciado e descrito ao longo deste trabalho, existem dois algoritmos principais e dois auxiliares. Os resultados com estes algoritmos serão apresentados e descritos agora de seguida.

5.1 Resultados com o algoritmo Gerador de Pseudopalavras de 1-10 sílabas

Possíveis resultados com o algoritmo "Gerador de pseudopalavras de 1-10 sílabas", utilizando os parâmetros por definição (10 pseudopalavras e 3 sílabas)e todos os "Resultados"selecionados.

Pseudopalavras	OLD20	Dist 1 sub	Dist 1	Dist 2	Dist 3	Lists
inconte	2.15	0	0	17	197	arconte,conte, ...
ressora	1.85	1	3	34	328	ressoa, ressona, ...
sublecer	2.85	0	0	3	29	sublevar, sublocar, ...
demargar	2.2	1	1	14	142	demarcar, amargar, ...
chilreite	2.9	0	0	5	12	chilreante, chilreia, ...
engalha	1.8	4	4	36	274	encalha, engelha, ...
demarga	2	1	1	18	305	demarca, amarga, ...
lobrica	1.85	3	3	21	239	lobriga, lubrica, ...
internal	1.7	5	6	27	150	infernal, interna, ...
saltaba	1.85	3	3	33	364	saltada, saltara, ...

Tabela 5.1: 10 resultados para a geração de 10 pseudopalavras de 3 sílabas.

O tempo médio de geração das 10 PP foi de 0.69927 segundos.

Na tentativa de gerar 100,000 PP de 3 sílabas, os primeiros 3 resultados e os 2 últimos foram os seguintes:

Pseudopalavras	OLD20	Dist 1 sub	Dist 1	Dist 2	Dist 3	Lists
riana	1.7	4	6	146	1699	ariana, diana, ...
riado	1.3	5	14	240	1971	criado, rfiado, ...
menteca	2.1	0	0	18	288	centena, enoteca, ...
óscane	2.6	0	0	8	156	escale, escame, ...
óscaro	1.95	0	1	19	325	óscar, caro, ...

Tabela 5.2: As primeiras 5 PP na geração de 1 milhão de PP de 3 sílabas.

O tempo médio de geração foi de 679.981 segundos e só foi possível gerar, no máximo, 22,186 PP.

Com o intuitivo de gerar 10 PP de 8 sílabas os resultados foram os seguintes:

Pseudopalavras	OLD20	Dist 1 sub	Dist 1	Dist 2	Dist 3	Lists
flautabotoziguacheste	13	0	0	0	0	farmacologicamente, ...
deladotaipazinado	8.55	0	0	0	0	desacompanhado, ...
desentranamengasteirar	12.65	0	0	0	0	desgovernamentalizar, ...

Tabela 5.3: As 3 primeiras PP na geração de 10 PP de 8 sílabas.

O tempo médio de processamento foi de 0.70161 segundos.

Um dos algoritmos denominado de gera_pp_1sil, gera PP de 1 sílaba tal como o nome indica. Este algoritmo é chamado quando ou no algoritmo "Palavra-protótipo"é introduzida uma palavra de 1 sílaba ou quando no algoritmo "Gerador de sílabas"pretende-se gerar PP de 1 sílaba. Para a geração de 50 PP de 1 sílaba os resultados foram os seguintes:

Pseudopalavras	OLD20	Dist 1 sub	Dist 1	Dist 2	Dist 3	Lists
méns	1.8	2	4	82	922	améns,mins, ...
hís	1.95	1	1	100	846	hás, ais, ...
xou	1.8	3	4	137	1288	dou, ou, ...
déis	1.7	6	6	87	1041	dais, deis, ...
ãos	1.35	7	13	233	1671	aos, dos, ...

Tabela 5.4: As primeiras 5 PP na geração de 50 PP de 1 sílaba.

O tempo médio de geração foi de 0.25131 segundos.

5.2 Resultados com o algoritmo Palavra Protótipo

Para o algoritmo Palavra protótipo, possíveis resultados para a palavra predefinida, "estudar", quando se pretende 100 PP.

Pseudopalavras	OLD20	Dist 1 sub	Dist 1	Dist 2	Dist 3	Lists
sultudar	2.95	0	0	1	58	suturar, aculturar, ...
futudar	2.4	1	1	10	139	futurar, autuar, ...
esture	1.7	5	6	70	473	estere, estire, ...
leitudar	2.8	0	0	4	50	estudar, leituga, ...
esladar	2	0	0	28	429	deslaçar, enfadar, ...

Tabela 5.5: As primeiras 5 PP através de derivações da palavra "estudar"

O tempo médio de geração foi de 4.4376 segundos e foi possível gerar as 100 PP derivadas de "estudar".

Outro algoritmo de nome de palavra_prot_2sil, gera PP de 2 sílabas. Este algoritmo é chamado dentro do algoritmo "Palavra-protótipo"quando é introduzida uma palavra de 2 sílabas. Possíveis resultados para a introdução da palavra, "porta"e com o intuito de serem geradas 1000 PP, serão mostrados apenas cinco resultados, 3 PP que mudassem a primeira sílaba e 2 PP que mudassem a 2ª sílaba.

Pseudopalavras	OLD20	Dist 1 sub	Dist 1	Dist 2	Dist 3	Lists
enta	1.3	4	14	272	2346	anta,benta, ...
nista	1.55	8	9	131	1214	cista, dista, ...
mita	1	18	27	397	2450	cita, dita, ...
porme	1.8	4	4	113	1068	dorme, forme, ...
porquei	1.95	0	1	24	168	porque, aparquei, ...

Tabela 5.6: As primeiras 5 PP através de derivações da palavra "porta"

O tempo médio de geração foi de 5.6606 segundos e só foi possível gerar, no máximo, 242 PP parecidas a "porta".

O objetivo foi atingido, na medida em que é possível, em tempo real, gerar pseudopalavras pronunciáveis e com um tempo de geração (processamento) bastante rápido.

Capítulo 6

Conclusões e trabalho futuro

Foi possível obter um corpus lexical suficientemente grande, sem palavras com hífen, sem estrangeirismos e sem siglas o que é altamente benéfico para a formação de pseudopalavras, já que tendo um bom alicerce, os algoritmos retornam muito bons resultados.

Também foi possível e necessário criar um vocabulário de sílabas, um vocabulário diferente de sílabas que não contém sílabas que sejam palavras no léxico, não contém dígrafos, nem palavras começadas por "nh"e "ç". E uma bigrama de sílabas com diferentes parâmetros associados a cada par de sílabas, através do ficheiro de léxico, de modo a ser possível a geração de pseudopalavras, já que os algoritmos estão altamente dependentes de quatro ficheiros, estes foram enumerados anteriormente.

Os algoritmos de geração de pseudopalavras foram concluídos e com o devido interface gráfico.

Os resultados, que foram apresentados no capítulo 5 mostraram-se satisfatórios, já que, foi possível criar pseudopalavras aleatórias que podem ser lidas, através dos mesmos encontros silábicos, existentes nas palavras do léxico.

Apesar dos resultados terem sido satisfatórios, não é possível afirmar, com certeza, que não existe nenhuma palavra dentro das pseudopalavras, isto porque não foi feito nenhum algoritmo que verifique um possível lema das pseudopalavras geradas.

O primeiro algoritmo descrito foi o denominado de "Gerador de 1-10 sílabas", descrito em 4.1.1, dá origem a muitas pseudopalavras acabadas por " consoante e seguidas por i ", tanto no singular como no plural, o que não é muito provável de acontecer na língua portuguesa, já que, as palavras acabadas pelo enunciado anteriormente, são geralmente formas verbais que indicam passado.

O segundo algoritmo foi o "Palavra Protótipo", descrito em 4.1.2, só retorna pseudopalavras com uma sílaba diferente, em cada posição da palavra protótipo, assim não permite ao utilizador escolher quão diferentes ou parecidas, as pseudopalavras serão. Assim, num melhoramento futuro, devia de ser possível ao utilizador a escolha do número de sílabas diferentes em comparação à palavra protótipo. Esse valor teria que ser maior do que 0 e menor ou igual ao número de sílabas dessa palavra protótipo. Uma solução caso o utilizador escolhesse 3 sílabas diferentes para uma palavra com 3 sílabas, podia ser

a chamada do algoritmo gera_pp. Mas também tinha que ser ser garantido que as sílabas da palavra nunca aconteceriam.

O trabalho futuro passará pela criação de um algoritmo que permita verificar se a pseudopalavra gerada pode ser uma flexão válida do lema identificado mas que não exista no vocabulário (caso mais provável na flexão verbal). Esta verificação garante uma maior probabilidade de não existirem palavras entre as pseudopalavras. Este algoritmo também permitiria mostrar ao utilizador por forma de curiosidade nos resultados, um "pseudolema"para as PP geradas.

Outro objetivo futuro passará pelo desenvolvimento de todos estes algoritmos mas neste caso em C++ e o interface com o utilizador seria em HyperText Markup Language (HTML), de modo a ter algoritmos altamente eficientes e um interface *user friendly* e facilmente utilizável por qualquer pessoa através dum *browser*.

Bibliografia

[1]

[2] Emmanuel Keuleers and Marc Brysbaert. Wuggy: A multilingual pseudoword generator. *Behavior research methods*, 42(3):627–633, 2010.

[3] L.; Mata L; Rosa M. Silva, I. ; Marques. Orientações curriculares para a educação pré-escolar, 2016.