3
A importância da utilização dos dados da Internet

Como já dissemos na introdução deste trabalho, não há pesquisa que trate especificamente do processamento do Internetês por *parsers* (analisadores sintáticos) e da necessidade de, para isso, lidar com a abreviação. Procuramos convergir trabalhos de diferentes autores para que, cada qual em seu campo, pudesse contribuir com nossa pesquisa.

Além disso, ainda há pouco desenvolvimento em língua portuguesa nesse campo. Por tal motivo, a maioria dos autores que nos servirão como base para esta seção trata da língua inglesa (Mikheev, 2002), (Shieber; Baker, 2003), (Yu et al, 2006), (Park; Byrd, 2001), (Terada; Tokunaga; Tanaka, 2002). Teremos, também, a contribuição de um trabalho que tem como objeto de estudo a língua portuguesa (Gouveia; Teixeira; Freitas, 2000). Porém, a dificuldade de bibliografia na área serve para nos estimular ainda mais em nossa busca do desenvolvimento de um trabalho em nossa língua.

Alguns artigos que utilizamos neste trabalho ressaltam a pouca atenção dada ao reconhecimento das abreviaturas para o processamento automático de linguagem natural, como os trechos citados na introdução desta tese.

Com as citações, percebemos como seria importante reconhecer automaticamente, de forma correta, as abreviaturas encontradas em diversos documentos, facilitando a extração de informação. Dannélls (2006) confirma a importância desse reconhecimento:

> There are many on-line documents which contain important information that we want to understand, thus the need to extract glossaries of domain specific names and terms increases, especially in technical fields such as biomedicine where the vocabulary is quickly expanding. One known phenomenon in biomedical literature is the growth of new *acronyms*.[10]

O que queremos deixar claro é que o estudo que propomos, de descrição das várias estratégias de expressão no internetês, pode servir como ponto de partida

[10] Há muitos documentos on-line que contêm informações importantes que queremos compreender, portanto, a necessidade de extrairmos glossários de nomes específicos de domínios e termos aumenta, especialmente nos domínios técnicos como a biomedicina, onde o vocabulário

para outras áreas da ciência, como vemos nas referências aqui utilizadas. Documentos da área médica, da biologia, relatórios de aviação, entre outros, contêm uma infinidade de abreviaturas cujo reconhecimento automático, após uma formalização de regras, facilitaria imensamente a compreensão das informações neles contidos.

Em suas considerações finais, o trabalho de Park e Byrd (2001) indica a necessidade de se desenvolverem programas que reconheçam abreviaturas em situações informais, uma das possibilidades que se apresentam após o desenvolvimento de nossa pesquisa:

> Mechanisms for processing these abbreviations, which tend to occur in informal text such as email, chat rooms, or customer service call records, are the subject of ongoing research in our project. (p. 8)[11]

Alguns dos autores pesquisados, como Terada, Tokunaga e Tanaka (2002), alertam para a dificuldade encontrada por pessoas que não estão acostumadas com certos tipos de abreviaturas. O caminho natural percorrido por elas é buscar seus significados no contexto e sugeri-los por meio dos caracteres das abreviaturas:

> Although the use of abbreviations does save space, they are sometimes hard to understand by those who are not familiar with the field where these abbreviations are commonly used. Readers can expand abbreviations by looking at words around them (context information) and by analyzing the characters in the abbreviations. (p. 2)[12]

No caso das palavras do internetês, como já dissemos anteriormente, às vezes falta informação contextual. Por exemplo, uma pessoa pouco acostumada com esse tipo de linguagem, ao se ver diante da mensagem “bom fds”, não terá como procurar seu significado no texto. O único meio de descobrir é supor, que, estando em uma sexta-feira, e sendo essa uma saudação no fim da mensagem, esse “fds” seja a sigla de “fim de semana”. Com nosso estudo, pretendemos descrever

---

vem rapidamente se expandindo. Um fenômeno conhecido na literatura biomédica é o crescimento de novas siglas.

[11] Mecanismos para o processamento dessas abreviaturas, que tendem a ocorrer em texto informal, como email, salas de chat, ou registros de atendimento ao cliente, são objeto de investigação em curso em nosso projeto.

[12] Embora o uso de abreviaturas economize espaço, elas às vezes são difíceis de entender por quem não está familiarizado com o campo onde essas abreviações são comumente usadas. Os leitores podem expandir as abreviações de olhar para as palavras em torno deles (informações de contexto) e, analisando os personagens do abreviaturas.

os vários possíveis caminhos ao se abreviar palavras na Internet, o que facilitaria o entendimento da sigla citada.

Outro texto que nos auxilia em nossa pesquisa é o de Climent, Moré e Oliver (2003), que trata da tradução automática de e-mails do catalão para o espanhol e vice-versa. Algumas observações dos autores nos são bastante úteis:

> ...communication via e-mail is strongly characterized by their intensive use of non-standard language – plus some visual information resources, and a wide range of unforeseeable errors. (p. 1-2)[13]

Os autores reconhecem a dificuldade de padronizar textos tão informais e livres como os que são praticados em e-mails. A proposta desse trabalho é ser fiel ao texto original, sem fazer qualquer tipo de modificação que altere seu caráter, para que o resultado da tradução não resulte em um produto artificial. Mais à frente, é feita uma lista com os diversos tipos de desvios ocorridos em e-mails, os quais trazem dificuldades com trabalhos de reconhecimento automático, como falta de pontuação, que causa problemas na delimitação de sentenças; falta de acentuação; utilização de palavras ambíguas; dificuldade em decidir pela tradução ou não tradução de nomes próprios.

Esse trabalho é mais um a mostrar a utilidade de se conhecer mais profundamente a linguagem informal da Internet.

A seguir, veremos alguns resultados já obtidos no que se refere ao tratamento automático de palavras não dicionarizadas, principalmente as abreviaturas. Primeiramente, discutiremos o trabalho desenvolvido em língua portuguesa, em que são expostas regras para a divisão silábica automática de textos, as quais julgamos serem úteis no desenvolvimento de nossa pesquisa.

3.1
Divisão silábica automática em língua portuguesa

[13] ...a comunicação via e-mail está fortemente caracterizada pelo uso intensivo de língua não-padrão – além de alguns recursos de informação visual, e uma grande variedade de erros imprevisíveis.

Em busca de um caminho a seguir no reconhecimento de abreviaturas, pensamos, inicialmente, em encontrar algum programa que fizesse, automaticamente, a divisão silábica das palavras, o que facilitaria uma posterior comparação entre palavras “estendidas” e abreviaturas.

Foi nesse caminho que encontramos o trabalho de Gouveia; Teixeira; Freitas (2000), o qual apresenta um algoritmo que permite realizar automaticamente a separação silábica como uma etapa de um trabalho mais extenso, que é o estudo de modelos prosódicos para o português europeu, enquadrado no desenvolvimento de um sintetizador de fala, como define o próprio autor no resumo de seu trabalho.

Para os fins de nossa pesquisa, interessa-nos analisar a parte do artigo em que são analisados dados de textos escritos. No desenvolvimento do referido trabalho, são expostas as possibilidades de divisão silábica em língua portuguesa. Além das considerações mais conhecidas e básicas, os autores contribuem com algumas informações bastante detalhadas. Como não nos compete, neste momento, listar todo o trabalho, elegemos alguns trechos representativos:

> a divisão silábica nunca pode ocorrer entre uma consoante e uma vogal (C-V); então as duas consoantes começam uma nova sílaba se constituírem um par de consoantes inseparáveis, isto é, a primeira delas pertence ao conjunto {*b, p, d, t, g, k, v, f*} (*k* corresponde a uma das letras *k*, *c* ou *g*), e a segunda pertence ao conjunto {*l*, e *r*} (ex: a-tlas); caso contrário a divisão silábica ocorrerá necessariamente entre as duas consoantes (ex: al-tas). [...] três consoantes seguidas nunca podem pertencer a uma mesma sílaba. (p.4)

Podemos perceber que é feita uma análise cuidadosa e uma explicação bastante clara sobre quais consoantes são inseparáveis e quais não o são. Esses detalhes são interessantes na nossa busca em descrever detalhadamente o internetês.

Outro trecho que demonstra esse cuidado na descrição da estrutura silábica da língua portuguesa é o seguinte:

> Considera-se uma semivogal fonética se se tratar de um *i* ou *u* e não lhe suceder um *r* ou um *l* como última letra da palavra ou como primeira de duas ou mais consoantes (ex: cai, cai-ro, ca-ir e ca-ir-mos). Não é considerada semivogal quando precedida por uma vogal igual (ex: ni-ilismo), ou então, se lhe suceder a vogal *u* (ex: ca-iu) ou uma consoante indicadora de nasalidade (ex: a-in-da). Para além dos casos referidos, considera-se ainda como semivogal a letra *o* precedida pela letra *a* (ex: ao). (p.4)

Aqui, os autores fazem uma explanação sobre as semivogais, informações igualmente importantes em nossas investigações sobre as diversas formas de expressão da linguagem da Internet.

Uma estratégia sugerida pelos autores, ainda no referido artigo, para o pré-processamento que teremos de fazer antes de identificarmos as abreviaturas e seus respectivos pares, é a substituição de algumas partes da palavra, como dígrafos e ditongos, por códigos, o que facilitaria a formalização das regras.

Munidos dessas informações, pensamos na possibilidade de se desenvolver um programa que automatize os seguintes passos: 1) haveria, em um banco de dados, um dicionário com diversas palavras em sua forma normal (estendida), separadas silabicamente; 2) ao digitarmos uma abreviatura (pensemos em *ksa*, por exemplo), o programa verificaria se ali há sílabas válidas em língua portuguesa; 3) não encontrando essa sílaba válida, o programa recorreria a outro banco de dados, em que estariam disponíveis informações sobre as sílabas mais abreviadas pelos internautas (aqui, estaria disponível a informação de que *k*, estatisticamente, é a abreviatura mais usada para sílabas que começam com *ca*; 4) substituir-se-ia a letra *k* pela sílaba *ca*; 5) novamente, o programa recorreria ao banco de dados com as sílabas válidas e verificaria a palavra; 6) agora, então, o programa apresentaria a forma estendida *casa*.

Imaginemos agora uma abreviação como “kro” (quero). A busca se daria inicialmente pela sílaba “ca”, e encontraríamos a palavra “caro”, e o processo pararia aí, com recuperação da forma errada. Uma solução possível para esse tipo de problema seria a apresentação de caixa de diálogo para o usuário, indicando todas as possibilidades de extensão da abreviatura para sua escolha, nos moldes do que faz atualmente um corretor ortográfico. Veja-se, como exemplo, a mensagem de um corretor:

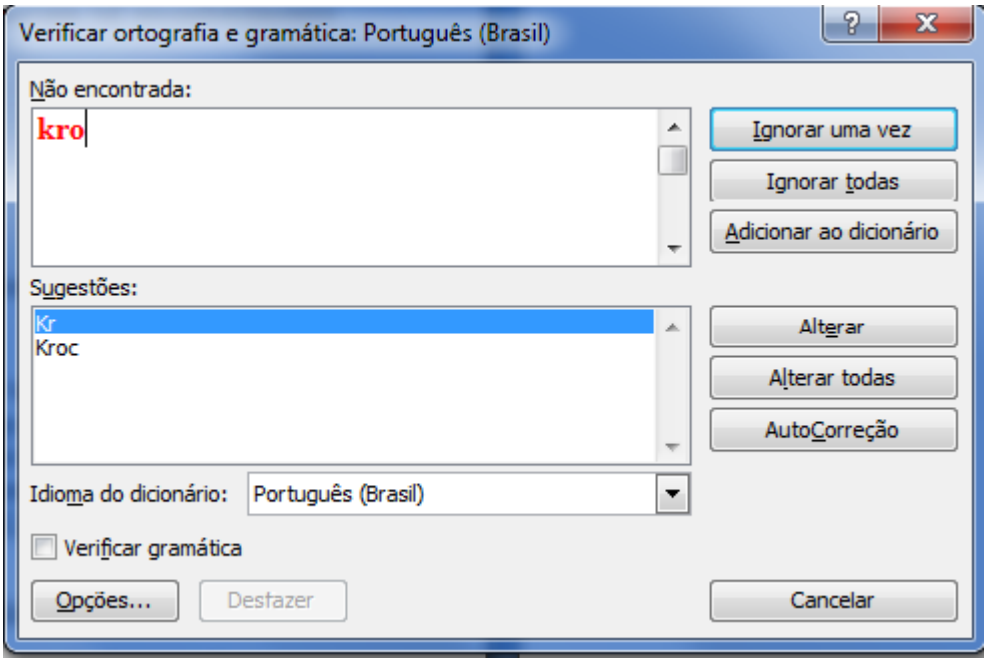


Imagem 6 – Exemplo de caixa de diálogo

Esse seria um caminho duvidoso, portanto, e por isso teríamos de buscar outras estratégias de reconhecimento de abreviaturas. Veremos, a seguir, alguns estudos que buscam tal reconhecimento e um posterior desenvolvimento dessas palavras.

3.2
Reconhecimento e expansão das abreviaturas

Para iniciar esta seção, vejamos o que nos sugere o artigo de Shieber; Baker (2003):

> For instance, the string of words <an> <example> <of> <num> <words> would be successively assigned a probability according to the language model (LM); converted to the sequence of characters “an example of <num> words" (SP); abbreviated to the sequence “an exmpl of <num> wrds" (CMP); and completed by instantiation of the special token <num> to, e.g., “an exmpl of 5 wrds" (UNK). (p.294)[14]

[14] Por exemplo, a sequência de palavras <an> <example> <of> <num> <words> seria sucessivamente atribuída a uma probabilidade de acordo com o modelo de linguagem (LM); convertida para a sequência de caracteres "an example of <num> words" (SP); abreviado para a

O LM (language model) representa modelos de frases a serem reconhecidas pelo programa, antes treinado com milhões de frases coletadas no *Wall Street Journal*.

O SP (spelling model) converte as possibilidades listadas no LM em caracteres reais a serem analisados, preservando os chamados tokens[15] especiais, como entrada de algarismos numéricos.

Por exemplo, no caso acima, em língua portuguesa, o LM traria a possibilidade da frase <artigo> <nome> <preposição> <número> <nome>. O SP encaixaria a frase real nesse modelo. Essa frase poderia ser <uma> <casa> <com> <cinco> <quartos>.

Já, o CMP (compression model) implementa o modelo de abreviação, removendo as vogais e as consoantes duplas. Porém, algumas regras são preestabelecidas aqui, como, por exemplo, não remover as vogais que iniciam a palavra, o que prejudicaria o entendimento da mesma. Aplicando-o ao exemplo em português, teríamos <um> <cs> <cm> <cinco> <qrts>. Observa-se que o campo referente ao algarismo numérico não sofre modificações, pois o modelo o considera um token especial.

Outra característica a ser levada em conta nessa fase do processo é o fato de a palavra <uma>, em língua portuguesa, ter sua vogal final retirada no *compression model*, o que traria um problema no reconhecimento da forma feminina da palavra. Além disso, "uma" pode representar ora um artigo feminino, ora um numeral. Esses e outros problemas que podem aparecer nos alertam para a necessidade de se elaborarem regras específicas para a nossa língua, já que as regras desenvolvidas para a língua inglesa não cobrem todas as nossas peculiaridades. Porém, aqui, estamos tratando apenas de modelos que podem nos servir de inspiração em nossa pesquisa.

Por último, tratamos do UNK (unknowns model), o qual substitui os tokens especiais (numerais, símbolos etc.) por algarismos numéricos ou outros caracteres.

No nosso exemplo, teríamos a frase final, após todos os processos: <um> <cs> <cm> <5> <qrts>.

---

sequência de "an exmpl of <num> wrds"(CMP); e completado pela instanciação do token especial <num> para, por exemplo, "an exmpl of 5 wrds" (UNK).

[15] Tal expressão é utilizada para representar o número de ocorrências total de cada palavra. Se, por exemplo, a palavra hoje aparece cinco vezes no corpus, temos o total de cinco *tokens*, apesar de a palavra ser a mesma.

Do trecho acima, o que mais nos interessa é o chamado *Compression Model* (CMP), em que são retiradas das palavras todas as vogais, menos as iniciais. Para fins de reconhecimento de palavras do internetês, adotar-se-ia essa estratégia como um pré-processamento, uma primeira tentativa de identificar uma abreviatura e sua forma estendida.

Por exemplo, digamos que, em um banco de dados, haja um dicionário de língua portuguesa e que haja também outro banco em que todas as palavras do dicionário estejam na forma comprimida sugerida por Shieber e Baker (2003), com suas regras adaptadas à nossa língua. Para processar um texto advindo de um meio informal da Internet, nosso programa identifica a palavra abreviada *vc*. Em nosso banco de dados, haverá a forma comprimida *vc*, ligada à forma estendida *você*. Esse problema já seria resolvido de forma simples. Porém, pelo fato de haver diversas outras formas de abreviação, essa estratégia resolveria apenas pequena parte dos problemas no reconhecimento automático de palavras.

Com um estudo mais aprofundado sobre as diversas formas de representação das palavras no meio virtual, essa associação seria bem mais eficaz em seu propósito, tornando úteis milhões e milhões de dados hoje em dia deixados de lado por conta do difícil processamento de palavras não reconhecidas pelos *parsers*.

Outro trabalho interessante é o de Park e Byrd (2001). Os autores definem uma fórmula para a criação de um sistema de reconhecimento automático de abreviaturas e seus pares:

> Uma regra de abreviação descreve como uma abreviatura é formada a partir de sua definição. Uma regra de abreviação, R, consiste em um modelo de abreviação (A_Pattern), uma definição de um modelo (D_Pattern) e de uma regra de formação (F_Rule).
>
> R = <A_Pattern, D_Pattern, F_Rule>[16]

Nesse modelo sugerido, há a necessidade de a definição da abreviatura estar no texto, para que o sistema possa testar os possíveis pares, combinando-os. Park; Byrd (2001) sugerem os seguintes passos para o reconhecimento automático das abreviaturas:

---

[16] Tradução nossa.

1) Dada uma abreviatura, substituem-se os caracteres alfabéticos por ‘c’ e as sequências de caracteres numéricos e de pontuação (incluindo ‘.’ e ‘,’) por ‘n’. Dessa forma, *SK8* seria um exemplo de abreviatura *ccn*;
2) Depois, a definição da abreviatura (a palavra inteira) seria procurada, no texto, à direita e à esquerda do contexto em que ela aparecera;
3) Substituem-se, nas possíveis definições procuradas, os caracteres especiais, como a barra (*/*), por espaços. Por exemplo, Input/Output => Input Output;
4) Separam-se caracteres alfabéticos dos numéricos: Windows98 => Windows 98;
5) Separam-se prefixos das “headwords[17]”: reusable => re usable;
6) A definição do exemplo de abreviatura seguiria o esquema de substituição: word (w); stopword (s)[18]; prefix (p); headword (h)[19] e number (n);
7) Formam-se regras, que definem cinco tipos de métodos de formação: F, em que o primeiro caractere da palavra aparece na abreviatura; I, em que um caractere do meio da palavra aparece na abreviatura; L, em que o último aparece; E, em que há a substituição exata (com caracteres numéricos – o caractere *3*, por exemplo, representando a palavra *três*); e R significa uma substituição – por exemplo, se é encontrado *X* na abreviatura, ele pode estar substituindo *hex*, *ex*, *trans*, or *cross*; *1* pode significar *primeiro*, *um* (numeral), *um* (artigo) etc.
8) Validados os pares, criam-se regras de abreviação, com base no exemplo de abreviação, na definição dos exemplos e na formação da regra.

Após um treinamento de um programa com um corpus (situação em que textos são disponibilizados para que as regras sejam testadas pelo programa, em que a definição da abreviatura deve estar no próprio texto), através dessas regras, poderíamos ter a seguinte situação:

> Suponhamos, por exemplo, que a abreviação “5GL” e a definição “fifth generation language” sejam encontradas em um texto. O sistema os preprocessa e gera seus

[17] Seria a palavra principal, nuclear, ou a palavra primitiva.
[18] Palavras curtas, que são recorrentes, com funções específicas, como preposições e advérbios.
[19] Palavra principal, como *utilizável*, em reutilizável.

> modelos. Nesse caso, o A_Pattern da abreviação é "ncc" e o D_Pattern é "www". Uma regra de formação <(1, R) (2, F) (3, F)> é associada à regra-base. Então, o sistema aplica a regra para determinar a validade do par abreviação/definição. A primeira palavra ('fifth') pode ser substituída por '5' [(1, R)]; o primeiro caractere da segunda palavra é 'G' [(2, F)]; e o primeiro caractere da terceira palavra é 'L' [(3,F)]. O par é válido e 'fifth generation language' é considerada a definição de '5GL'. (p. 5)[20]

A partir da validação do par, o sistema cria regras para que outros pares do tipo sejam formados, descobrindo-se, assim, a definição da abreviatura. Porém, conforme alertado anteriormente, esse modelo só funciona se a definição da abreviatura se encontrar no próprio texto. Note-se que, se a unidade de análise for um corpus e não cada texto que o compõe isoladamente, a possibilidade de encontrarmos pares de siglas e expressões estendidas aumenta enormemente.

Como exemplo na língua portuguesa, podemos trazer a abreviatura (ou sigla, neste caso) *FDS* e sua definição *fim de semana*. Aplicando as regras descritas, teríamos <ccc, www, (1,F) (2,F) (3,F)> Ou seja, *F*, *D* e *S* são caracteres alfabéticos, por isso a sequência *ccc*. Além disso, tanto *F*, quanto *D* e *S* representam palavras (www). As representações (1,F) (2,F) (3,F) significam que o primeiro caractere de cada palavra abreviada coincide com a sua definição, ou seja, FimDeSemana. Ao encontrar a sequência *fim de semana* no texto e aplicar as regras descritas, o sistema traria como resultado o par *FDS / fim de semana* como válido.

Para que essas regras sejam bem aplicadas e funcionem efetivamente, há a necessidade de uma descrição bastante detalhada das possíveis abreviações que possam ser criadas, de modo que não haja falhas nas associações entre a palavra abreviada e sua definição.

Da mesma forma que as regras descritas anteriormente, desenvolvidas por Shieber e Baker (2003), há a necessidade aqui de adaptação para o caso da língua portuguesa. Porém, esse também é um bom caminho a se seguir no estudo das palavras não dicionarizadas do internetês, principalmente as siglas, como *FDS* (fim de semana), *TDB* (tudo de bom) etc.

Reconhecemos a complexidade de um sistema como esse para aqueles que não têm muita familiaridade com programação de sistemas. Nosso intuito é mostrar superficialmente o que se tem disponível hoje em dia a respeito do

[20] Tradução nossa.

reconhecimento automático de abreviaturas e palavras não dicionarizadas. No caso da linguagem da Internet, na maioria das vezes, a definição da abreviatura não está no próprio texto, mas fora dele. Porém, as ideias aqui sugeridas podem ser aproveitadas para formalizar as abreviaturas que aparecem no internetês.

Vejamos a seguir outras pesquisas que buscam um resultado mais eficiente no tratamento automático de palavras.

3.2.1
Os lexical blendings

Iniciaremos esta seção com o trabalho desenvolvido por Cook e Stevenson (2007), que busca encontrar automaticamente significados de palavras criadas pelo chamado "lexical blending", que seria um tipo de composição de palavras por aglutinação. Eles dão como exemplo inicialmente a palavra *brunch* (breakfast + lunch).

Como justificativa para sua pesquisa, eles alertam que esse processo de formação de palavras é muito recorrente, o que torna muito útil um trabalho nessa área. Vejamos um trecho de seu texto:

> Fortunately, people rarely create completely new words that give no clue as to their meaning. One common means for introducing novel words is subtractive word formation processes, in which existing words are clipped and possibly combined. Such methods include shortening (*lab* for *laboratory*), forming an acronym (*CL* for *computational linguistics*), or lexical blending. (p. 1)[21]

Para sermos mais específicos, o trabalho, baseado em diversas regras criadas, propõe-se a encontrar, em uma lista, quais palavras serviram de base para a criação da nova, aglutinada.

A busca é feita descobrindo-se, em um primeiro passo, prefixos e sufixos das palavras bases que formaram a nova palavra. Ou seja, se a palavra é *boatel* (boat + hotel), o programa considera como prefixos e sufixos os possíveis pares:

[21] Felizmente, as pessoas raramente criam palavras completamente novas, que não dão nenhuma pista sobre o seu significado. Um meio comum para a criação de "novas" palavras é o processo de formação por aglutinação, em que as palavras existentes são cortadas e, possivelmente combinadas. Tais métodos incluem a diminuição (*lab* para *laboratório*), a formação de um acrônimo (*LC* para a *Linguística Computacional*), ou os *lexical blendings.*

bo + atel, boa + tel, boat + el. Após esse passo, com o auxílio de uma lista, a comparação será feita entre os prefixos da primeira palavra e os sufixos da segunda e assim por diante. A partir daí, outras fórmulas estatísticas são utilizadas para que se encontrem as palavras-base e o possível significado daquela nova palavra formada por aglutinação. Os resultados encontrados são bons, mas a eficiência ainda não é total com todas as formas aglutinadas.

O que mais nos interessa nessa pesquisa são dois pontos: o reconhecimento de que é necessário se identificar novas palavras automaticamente, que aqui, no caso, são os *lexical blendings*; e o fato de a pesquisa utilizar-se de listas prontas, o que confirma nossa tese de imobilidade e ineficiência nos resultados, já que as listas teriam de ser atualizadas muitas vezes, haja vista o enorme crescimento lexical de todas as línguas vivas.

3.2.2
Os autômatos de estados finitos

Veremos a partir de nossos dados que muitas são as peculiaridades das palavras do internetês. Mais do que abreviaturas, essa linguagem é composta por diversas outras formas de expressão, cada qual com sua intenção. Além de a descrição feita por nós servir como orientação para professores e pessoas que desejem entender melhor como funciona esse mecanismo, ela também pode auxiliar na criação de programas que reconheçam automaticamente a linguagem da Internet, possibilitando que pesquisadores a utilizem como corpus para diversos estudos. Vimos, até agora, ao longo deste trabalho, diversos estudos que propõem procedimentos para o reconhecimento automático de abreviaturas e outras palavras desconhecidas, como no caso do internetês, que poderiam ser adaptados à nossa língua e abastecidos com nossas descobertas.

Como um exemplo de proposta de reconhecimento automático de palavras provenientes do internetês, trazemos outra linha de pensamento, advinda da área da informática, à qual tivemos acesso através dos diagramas apresentados por Allen (1994) para o tratamento de linguagem natural por computadores. Adaptaremos tais diagramas de análise de sentenças para *parsing*, utilizando-os

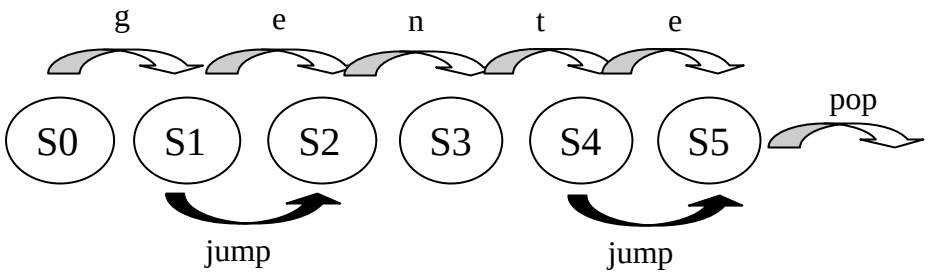
em palavras, mostrando as possíveis escolhas dos usuários do internetês na abreviação de uma palavra.

Tais diagramas são um exemplo de autômato de estados finitos. Com base no trabalho de Hopcroft, Motwani e Ullman (2003), podemos dizer que um autômato de estados finitos baseia-se numa máquina (sistema) que, ao receber uma entrada, "salta" por uma série de estados de acordo com uma função de transição (a qual pode ser expressa numa tabela). A entrada é lida passo a passo, ou estado a estado (no nosso caso, letra a letra), até que seja completamente consumida. Uma vez esgotada toda a entrada, dizemos que o autômato está parado. Dependendo do estado em que o autômato para, diz-se que o autômato aceitou ou rejeitou a entrada. Se ele paralisou no estado de aceitação, dizemos que ele aceitou a palavra. Se, por outro lado, ele parou no estado não aceitação, a palavra foi rejeitada.

Mostraremos alguns exemplos de como o autômato funcionaria, baseado nos nossos dados, incluindo os detalhes sobre as categorias de palavras do internetês, os padrões de sílabas mais abreviadas, as partes das sílabas mais suprimidas.

Vejamos, então, o primeiro exemplo:

A palavra *gente*, abreviada como *gnt* no corpus do Orkut, teria o seguinte diagrama, em que S = estado; jump = salto de algum estado; pop = final do processo:

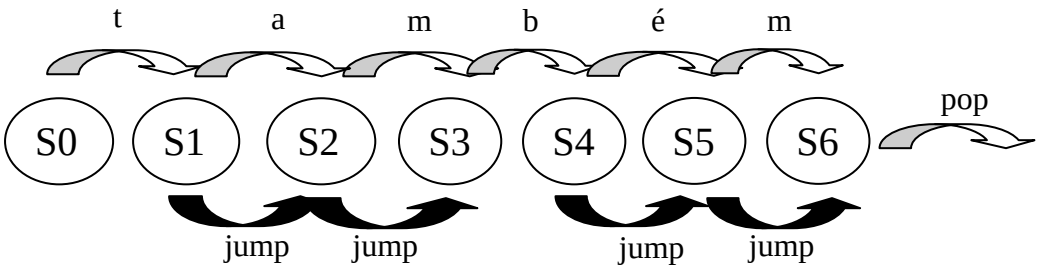


Vemos que o usuário, ao utilizar a palavra *gente*, pode seguir dois caminhos: a forma estendida *gente* (S0-S1-S2-S3-S4-S5-pop)[22] ou a forma abreviada *gnt* (S0-S1-jumpS2-S3-S4-jumpS5-pop), que aproveita o próprio som

[22] Há ligações que "carregam" a letra consigo e outras que apenas pulam de um estado para o outro, sem carregar consigo qualquer letra. Esses pulos são indicados pela palavra "jump".

das letras *g* e *t*. Neste caso, as sílabas abreviadas são CVC, e CV, como descritas por Camara Jr.

Como existem algumas palavras que possuem mais de uma abreviatura correspondente, o gráfico poderia mostrar outras opções de caminho:



Os caminhos seriam o da forma estendida *também* (S0-S1-S2-S3-S4-S5-S6-pop), o da forma abreviada *tb* (S0-S1-jumpS2-jumpS3-S4-jumpS5-jumpS6-pop) e o da outra forma abreviada *tbm* (S0-S1-jumpS2-jumpS3-S4-jumpS5-S6-pop).

Ou seja, de acordo com os dados encontrados por nós, seriam montados os gráficos, explicitando quais caminhos seriam os mais prováveis para a formação da abreviatura.

Ressalvamos mais uma vez que os dados tratados nesta seção servem apenas como amostra do que se pode fazer a partir de nossa pesquisa. Especialistas podem montar programas muito mais complexos, baseados nas informações alcançadas por nós.

3.2.3
Outras ferramentas

Outro sistema que descobrimos por meio da leitura do artigo da revista Língua portuguesa (Marconato, 2003) é o utilizado no site Fórum PCs[23], que expande automaticamente alguns tipos de abreviatura do internetês que venham a ser utilizados naquele ambiente.

Também contatamos os responsáveis pelo site, para investigar que tipo de sistema eles utilizavam para fazer tal expansão. Eles nos responderam que

[23] www.forumpcs.com.br

também se baseiam em listas prontas e que, além das abreviaturas, também censuram qualquer tipo de palavra chula, substituindo-as por “xxxxx”.

Analisando alguns comentários feitos no referido site, pudemos perceber que algumas palavras provenientes do internetês são realmente encontradas e substituídas, como q (que), naum (não) e ta (está). Porém, outras bastante utilizadas, mas que estão fora da lista, passam como palavras comuns.

Há também outras ferramentas na Internet que fazem, até mesmo, o caminho inverso, ou seja, a transformação de textos comuns em textos de internetês, como o “Miguxeitor[24]”, o qual se intitula “Tradutor on-line de português para miguxês”. Por exemplo, se digitamos a frase "Nossa pesquisa trata do internetês", temos três opções de "tradução":

Miguxês arcaico: que mostra as características do internetês na época em que o principal programa de bate-papo usado era o ICQ. As alterações são poucas, como a transposição de letras maiúsculas para minúsculas, a retirada de notações lexicais e o corte de algumas palavras. O resultado da tradução seria: "nossa pesquisa trata do netes".

Miguxês moderno: que mostra o internetês mais evoluído que o arcaico, com o MSN sendo a principal ferramenta de bate-papo. Além de serem feitas as alterações anteriormente descritas, substituem-se os dígrafos *qu* e as letras *c*, quando formam sílabas com *a*, *o* e *u* por *k*; as letras *o* e *l* finais por *u*; e as letras *s*, quando formam sílaba com a vogal *a*, por *z*. Há também alterações em outras palavras específicas, como a palavra *mercado*, que é alterada para *mercadeenhu*. O resultado da tradução seria: "nossa peskiza trata du netes".

Neo-Miguxês: que trata dos exageros cometidos por alguns internautas a fim de tornar sua linguagem única. Acumula as alterações comentadas anteriormente e insere outras, como alternância entre letras maiúsculas e minúsculas; acréscimo da letra *h* no final da palavra que termina com a vogal *a*; substituição do dígrafo *ss* por *xx*, entre outras. A tradução ficaria: "nOXXAh peSKIZah trAtah dU NeTEixXx".

Apesar de oferecer diversas possibilidades de transformação de textos comuns em textos do internetês, o programa também é baseado em listas prontas, as quais se desatualizam facilmente.

[24] http://www.coisinha.com.br/miguxeitor/

Vale ressaltar que tais ferramentas servem basicamente para o entretenimento e diversão de seus usuários, não buscando uma análise mais detalhada e profunda do fenômeno das abreviaturas e dos outros tipos de recursos utilizados no internetês.

3.3
Dificuldades no reconhecimento automático de abreviaturas

Como todo trabalho novo, o nosso também enfrentou muitas dificuldades. Algumas delas foram as mesmas listadas por autores experientes em pesquisas de reconhecimento automático de abreviaturas. Apesar de nenhum deles tratar especificamente de abreviaturas do internetês, seus relatos nos são muito úteis para o desenvolvimento de nosso trabalho.

Um dos maiores problemas em identificar precisamente, de forma automática, uma abreviatura é o fato de haver várias palavras que ela possa representar. Em seu trabalho sobre abreviaturas utilizadas em documentos de biomedicina, citando outros autores, Yu et al (2006) chamam a atenção para esse problema de ambiguidade:

> the abbreviation “CAT” denotes chloramphenicol acetyl transferase, computer-aided testing, computerautomated tomography, choline acetyltransferase, and computed axial tomography [Rimer and O’Connell 1998, apud YU et al, 2006] depending on the context. (p.381)[25]

Os autores ainda complementam:

> We identified abbreviations associated with two or more different full forms within the same MEDLINE record. (p.387)[26]

Em nossa pesquisa, até agora, pudemos identificar que costuma acontecer algo semelhante nas palavras do internetês: há mais de uma abreviação para a

[25] a abreviatura "CAT" denota 'chloramphenicol acetyl transferase', 'computer-aided testing', 'computerautomated tomography', 'choline acetyltransferase', and 'computed axial tomography' [Rimer e O'Connell 1998, apud YU et al, 2006], dependendo do contexto.
[26] Identificamos abreviaturas associados com dois ou mais diferentes formas completas dentro do mesmo registro MEDLINE.

mesma palavra. Por exemplo, *bj* e *bjo* representam *beijo*. Contudo, é importante que possamos identificar em quais contextos aparece uma ou outra forma.

Os métodos descritos anteriormente por Park e Byrd (2001) poderiam ser úteis nesses casos, já que buscariam pares dentro do contexto de informação. Porém, como já fora alertado, o sucesso do método proposto por eles depende de que haja a forma estendida da palavra dentro do próprio texto ou conjunto de textos analisados.

Algumas restrições são impostas para identificar as abreviaturas no trabalho dos autores já citados nesta seção, tais como:

> These rules include: the first letter of an abbreviation must match the first letter of the meaningful word of the full form; the abbreviation must match the first letter of each word in the full form; the abbreviation letter must match consecutive letters of a word in the full form; and the abbreviation letter must match a middle letter of a word in the full form if the first letter of the word matches the abbreviation. (p.383)[27]

> we consider a word so as an abbreviation if it matches to a list of common biomedical abbreviations, or the word is a one-letter word (a–z, but not “I”), or the word has no vowels and no digits, or the word contains a period and has no digits. (p.388)[28]

Em nosso trabalho, há casos em que essas regras não são totalmente obedecidas, já que, em alguns momentos, utilizam-se letras ou algarismos numéricos que representam sílabas inteiras (ksa = casa, crpente = serpente, 10bravando = desbravando). Nesse último caso, aproveita-se a fonética do algarismo *10*, mesmo que, na escrita, seja utilizada a letra *z*, em vez de *s*, como na palavra *desbravando*. Ou seja, tais restrições impedem a aplicação integral das regras em nosso estudo.

Já Terada, Tokunaga e Tanaka (2002) fazem uma observação para o caso em que é explorada a fonética de uma letra para a composição da palavra:

[27] Estas regras incluem: a primeira letra de uma abreviatura deve coincidir com a primeira letra da palavra significativa do formulário completo; a sigla deve coincidir com a primeira letra de cada palavra no formulário completo; a letra da abreviatura deve coincidir com as letras consecutivas de uma palavra em sua forma completa, e a abreviatura deve coincidir com uma letra do meio de uma palavra no formulário completo, se a primeira letra da palavra corresponde à abreviatura.

[28] consideramos uma palavra uma abreviatura se ela corresponder a uma lista de abreviações comuns na biomédica; ou se a palavra é uma palavra de uma letra (a-z, mas não "I"); ou se a palavra não tem vogais nem dígitos; ou se a palavra contém um ponto e não tem dígitos.

> Abbreviation expansion candidates include more characters than abbreviation candidates and abbreviation expansion candidates include the same characters in the same order as abbreviation candidates except for 'X', '-', and '/'. (p. 6)[29]

Nos casos citados, podemos ter, por exemplo, na língua inglesa, a letra X representando *ex* (Xtra, Xplain), *trans* (Xformer), *cross* (Xover), *ks* (ThanX), *kiss* (Xes), dentre outras, o que não corresponde exatamente às regras formuladas, assemelhando-se ao caso citado das regras de formação *R* de Park; Byrd (2001).

Também achamos válido citar Mikheev (2002). Em seu trabalho, o autor analisou corpora de diferentes origens: alguns abrangendo uma linguagem mais formal, outros abrangendo linguagem menos formal.

Uma das conclusões às quais chegou, para a qual nos alerta logo no início de seu artigo, é que cerca de 20% dos nomes ocorrem em posições ambíguas. Como exemplo, cita a palavra *Black,* que, em início de sentença, pode significar um sobrenome ou a cor (p. 290). Seu trabalho consiste em desambiguizar palavras com iniciais maiúsculas e abreviações considerando o contexto em que aparecem e suas repetições nos mesmos documentos (p. 291).

Vejamos um exemplo mais claro de como é feita essa identificação do sentido real da palavra:

> For instance, *Riders* in the sentence *Riders rode all over the green* is equally likely to be a proper noun, a plural proper noun, or a plural common noun. But if in the same text we find *John Riders*, this sharply increases the likelihood that the proper noun interpretation is the correct one, and conversely if we find *many riders*, this suggests the plural-noun interpretation. (p.296)[30]

O sistema desenvolvido pelo autor pode "decidir" qual o melhor caminho a seguir:

> For instance, if a phrase *Rocket Systems Development Co.* is found in a document starting from an unambiguous position (e.g., after a lower-cased word, a number, or a comma), the system collects it and also generates its partial-order subphrases:

[29] Candidatos a expansão da abreviatura incluem mais caracteres que os candidatos a abreviaturas e candidatos a expansão da abreviatura incluem os mesmos caracteres na mesma ordem como candidatos a abreviaturas, exceto para 'X', '-' e '/'.

[30] Por exemplo, *Riders* na sentença *Riders rode all over the green* tem a mesma probabilidade de ser um nome próprio, um nome próprio no plural, ou um substantivo comum no plural. Mas, se no mesmo texto, encontramos *John Riders*, este aumenta significativamentre a probabilidade de que a interpretação *nome próprio* seja a correta, e, inversamente, se encontrarmos *many riders*, isso sugere a interpretação de nome comum no plural.

> *Rocket Systems*, *Rocket Systems Co.*, *Rocket Co.*, *Systems Development*, etc. If then in the same document *Rocket Systems* is found in an ambiguous position (e.g., after a period), the system will assign the word *Rocket* as a proper noun because it is part of a multiword proper name that was seen in the unambiguous context. (p.301)[31]

Outro problema encontrado por Mikheev (2002) foi identificar quando um ponto faz parte de uma abreviação e quando ele encerra uma sentença. A mais simples estratégia para resolver esse obstáculo, segundo ele:

> The simplest strategy for deciding whether a word that is followed by a period is an abbreviation or a regular word is to apply well-known heuristics based on the observation that single-word abbreviations are short and normally do not include vowels (*Mr.*, *Dr.*, *kg.*). Thus a word without vowels can be guessed to be an abbreviation unless it is written in all capital letters and can stand for an acronym or a proper name (e.g., *BBC*). A span of single letters separated by periods forms an abbreviation too (e.g., *Y.M.C.A.*). A single letter followed by a period is also a very likely abbreviation. There is also an additional heuristic that classifies as abbreviations short words (with length less than five characters) that are followed by a period and then by a comma, a lower-cased word, or a number. All other words are considered to be nonabbreviations. (p.295)[32]

No caso das abreviações do internetês, encontramos poucas que utilizam pontos na sua formação.

Percebemos, até aqui, que as diversas pesquisas em relação ao tratamento automático de abreviaturas tratam de casos bastante específicos, como documentos médicos, por exemplo. Porém, o que queremos deixar claro é a importância de se encontrar caminhos para o reconhecimento automático dessas palavras não dicionarizadas. Tratamos especificamente do internetês, mas, com

[31] Por exemplo, se a frase *Rocket Systems Development Co.* é encontrada em um documento a partir de uma posição inequívoca (por exemplo, depois de uma palavra em minúsculas, um número ou uma vírgula), o sistema a recolhe e também gera sua ordem parcial subfrases: *Rocket Systems*, *Rocket Systems Co.*, *Rocket Co.*, *Systems Development*, etc. Se, então, no mesmo documento *Rocket Systems* encontra-se em uma posição ambígua (por exemplo, depois de um ponto), o sistema irá considerar a palavra *Rocket* como um nome próprio, porque é parte de um nome próprio com várias palavras, que foi visto no contexto inequívoco.

[32] A estratégia mais simples para decidir se uma palavra que é seguida por um ponto é uma abreviatura ou uma palavra regular é aplicar uma heurística baseada na observação de que as abreviaturas de palavras simples são curtas e, normalmente, não incluem as vogais (Sr., Dr., kg.). Assim, uma palavra sem vogais pode ser considerada uma abreviatura, a menos que esteja escrito em letras maiúsculas e possa representar um acrônimo ou um nome próprio (por exemplo, BBC). A extensão de letras isoladas separadas por pontos forma uma abreviatura também (por exemplo, Y.M.C.A.). Uma única letra seguida por um ponto muito provavelmente também seja uma abreviatura. Há também uma heurística adicional que classifica como abreviaturas palavras curtas (com comprimento inferior a cinco caracteres) que são seguidas por um ponto e depois por uma vírgula, uma palavra em minúsculas, ou um número. Todas as outras palavras são consideradas "não-abreviaturas".

um caminho definido nesse caso, haveria ideias para serem exportadas para outras áreas, trazendo inúmeros benefícios para cada uma delas.

Há, ainda, a escolha de tratar a questão a partir do paradigma computacional não simbólico, em que as regras são definidas pela frequência estatística de um dado fenômeno em grandes corpora, associada a algoritmos de aprendizagem de máquina. Não nos detivemos nessa linha de pesquisa por implicar o estudo de teorias e métodos que seriam totalmente novos para nós.