UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Normalização textual de conteúdo gerado por usuário

Thales Felipe Costa Bertaglia

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-C²MC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Thales Felipe Costa Bertaglia

Normalização textual de conteúdo gerado por usuário

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Maria das Graças Volpe Nunes

USP – São Carlos
Outubro de 2017

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

B536n Bertaglia, Thales Felipe Costa
Normalização textual de conteúdo gerado por
usuário / Thales Felipe Costa Bertaglia; orientador
Maria das Graças Volpe Nunes. -- São Carlos, 2017.
137 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2017.

1. Normalização textual. 2. Conteúdo gerados por
usuário. 3. Análise de textos ruidosos. I. Nunes,
Maria das Graças Volpe, orient. II. Título.

Thales Felipe Costa Bertaglia

User-generated content text normalization

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Maria das Graças Volpe Nunes

USP – São Carlos
October 2017

Aos meus avós e à memória de meu avô Enelvo Bertaglia.

*o p que
no pequeno &
se esconde
eu sei por q*

*só não sei
onde nem e
(Paulo Leminski, Caprichos & Relaxos – 1983)*

RESUMO

BERTAGLIA, T. F. C. **Normalização textual de conteúdo gerado por usuário**. 2017. 137 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

Conteúdo Gerado por Usuário (CGU) é a denominação dada ao conteúdo criado de forma espontânea por indivíduos comuns, sem vínculos com meios de comunicação. Esse tipo de conteúdo carrega informações valiosas e pode ser explorado por diversas áreas do conhecimento. Muito do CGU é disponibilizado em forma de textos – avaliações de produtos, comentários em fóruns sobre filmes e discussões em redes sociais são exemplos. No entanto, a linguagem utilizada em textos de CGU diverge, de várias maneiras, da norma culta da língua, dificultando seu processamento por técnicas de PLN. A linguagem de CGU é fortemente ligada à língua utilizada no cotidiano, contendo, assim, uma grande quantidade de ruídos. Erros ortográficos, abreviações, gírias, ausência ou mau uso de pontuação e de capitalização são alguns ruídos que dificultam o processamento desses textos. Diversos trabalhos relatam perda considerável de desempenho ao testar ferramentas do estado-da-arte de PLN em textos de CGU. A Normalização Textual é o processo de transformar palavras ruidosas em palavras consideradas corretas e pode ser utilizada para melhorar a qualidade de textos de CGU. Este trabalho relata o desenvolvimento de métodos e sistemas que visam a (a) identificar palavras ruidosas em textos de CGU, (b) encontrar palavras candidatas a sua substituição, e (c) ranquear os candidatos para realizar a normalização. Para a identificação de ruídos, foram propostos métodos baseados em léxicos e em aprendizado de máquina, com redes neurais profundas. A identificação automática apresentou resultados comparáveis ao uso de léxicos, comprovando que este processo pode ser feito com baixa dependência de recursos. Para a geração e ranqueamento de candidatos, foram investigadas técnicas baseadas em similaridade lexical e *word embeddings*. Concluiu-se que o uso de *word embeddings* é altamente adequado para normalização, tendo atingido os melhores resultados. Todos os métodos propostos foram avaliados com base em um corpus de CGU anotado no decorrer do projeto, contendo textos de diferentes origens: fóruns de discussão, *reviews* de produtos e publicações no Twitter. Um sistema, Enelvo, combinando todos os métodos foi implementado e comparado a um outro sistema normalizador existente, o UGCNormal. Os resultados obtidos pelo sistema Enelvo foram consideravelmente superiores, com taxa de correção entre 67% e 97% para diferentes tipos de ruído, com menos dependência de recursos e maior flexibilidade na normalização.

Palavras-chave: normalização textual, conteúdo gerado por usuário, análise de textos ruidosos.

ABSTRACT

BERTAGLIA, T. F. C. **User-generated content text normalization**. 2017. 137 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

User Generated Content (UGC) is the name given to content created spontaneously by ordinary individuals, without connections to the media. This type of content carries valuable information and can be exploited by several areas of knowledge. Much of the UGC is provided in the form of texts – product reviews, comments on forums about movies, and discussions on social networks are examples. However, the language used in UGC texts differs, in many ways, from the cultured norm of the language, making it difficult for NLP techniques to handle them. UGC language is strongly linked to the language used in daily life, containing a large amount of noise. Spelling mistakes, abbreviations, slang, absence or misuse of punctuation and capitalization are some noises that make it difficult to process these texts. Several works report considerable loss of performance when testing NLP state-of-the-art tools in UGC texts. Textual Normalization is the process of turning noisy words into words considered correct and can be used to improve the quality of UGC texts. This work reports the development of methods and systems that aim to (a) identify noisy words in UGC, (b) find candidate words for substitution, and (c) rank candidates for normalization. For the identification of noisy words, lexical-based methods and machine learning ones using deep neural networks were proposed. The automatic identification presented results comparable to the use of lexicons, proving that this process can be done with low dependence of resources. For the generation and ranking of candidates, techniques based on lexical similarity and word embeddings were investigated. It was concluded that the use of embeddings is highly suitable for normalization, having achieved the best results. All proposed methods were evaluated based on a UGC corpus annotated throughout the project, containing texts from different sources: discussion forums, product reviews and tweets. A system, Enelvo, combining all methods was implemented and compared to another existing normalizing system, UGCNormal. The results obtained by the Enelvo system were considerably higher, with a correction rate between 67 % and 97 % for different types of noise, with less dependence on resources and greater flexibility in normalization.

Keywords: text normalization, user-generated content, noisy text analysis.

LISTA DE ILUSTRAÇÕES

Figura 1 – <i>Review</i> escrita por usuário da <i>web</i>	4
Figura 2 – A colmeia de blocos funcionais de uma mídia social.	19
Figura 3 – Uma publicação no Twitter utilizando linguagem informal.	22
Figura 4 – Exemplos de <i>reviews</i> com erros anotados.	24
Figura 5 – Distribuição de erros por categoria.	25
Figura 6 – Modelos do <i>word embeddings</i> Word2Vec.	34
Figura 7 – Modelos de <i>word embeddings</i> Wang2Vec.	34
Figura 8 – Modelo de <i>word embeddings</i> FastText.	35
Figura 9 – Similaridades entre vetores de <i>word embeddings</i>	35
Figura 10 – Arquitetura de uma MLP.	41
Figura 11 – Neurônio de uma rede neural recorrente.	43
Figura 12 – Unidade recorrente desenrolada.	44
Figura 13 – Unidade de LSTM básica.	45
Figura 14 – <i>Pipeline</i> de normalização da ferramenta UGCNormal	52
Figura 15 – <i>Pipeline</i> de execução da metodologia proposta por Stilpen e Merschmann (2016).	55
Figura 16 – Modelo <i>encoder-decoder</i> para normalização.	65
Figura 17 – <i>Interface</i> de anotação do corpus TUB-UGC.	74
Figura 18 – <i>Interface</i> da seleção de categorias e inserção de correção.	74
Figura 19 – Distribuição de ruídos por categoria no corpus TUB-UGC.	77
Figura 20 – Frequência dos dez ruídos mais comuns no corpus TUB-UGC.	78
Figura 21 – Distribuição de ruídos por categoria para palavras únicas no corpus TUB-UGC.	78
Figura 22 – Distribuição de ruídos por categoria em <i>posts</i> no fórum da UOL.	79
Figura 23 – Distribuição de ruídos por categoria em <i>reviews</i> do Buscapé.	80
Figura 24 – Distribuição de ruídos por categoria em <i>tweets</i>	80
Figura 25 – Arquitetura da RCNN para identificação de ruídos.	90
Figura 26 – 25 vizinhos mais próximos da palavra <i>você</i> no modelo de <i>embeddings</i> Word2Vec.	99
Figura 27 – Arquitetura neural para correção de erros ortográficos contextuais. . .	109
Figura 28 – Fluxo de funcionamento da ferramenta Enelvo.	112
Figura 29 – <i>Interface</i> da anotação.	132
Figura 30 – Página inicial do sistema.	132

Figura 31 – Menu de anotação 133

LISTA DE TABELAS

Tabela 1 – Dados sobre participação em atividades relacionadas a CGU	15
Tabela 2 – Número de erros ortográficos em C1 por categoria.	26
Tabela 3 – Número de erros ortográficos em C2 por categoria.	27
Tabela 4 – Matriz de confusão	47
Tabela 5 – Sugestões de correção dadas por cada técnica de ranqueamento	53
Tabela 6 – Resultado da avaliação intrínseca do UGCNormal	54
Tabela 7 – Resultados obtidos pela metodologia proposta por Stiilpen e Merschmann (2016)	57
Tabela 8 – Resultados obtidos pela arquitetura proposta por Min e Mott (2015) .	63
Tabela 9 – Resultados obtidos pelo modelo proposto por Ikeda, Shindo e Matsumoto (2016)	66
Tabela 10 – Síntese dos trabalho de normalização textual apresentados	67
Tabela 11 – Distribuição de categorias do Córpus UGCNormal.	71
Tabela 12 – Estatísticas sobre amostras de CGU.	73
Tabela 13 – Os 10 ruídos mais frequentes em cada modalidade de CGU.	81
Tabela 14 – Valores de concordância obtidos na anotação.	82
Tabela 15 – Avaliação dos léxicos utilizados na identificação de ruídos.	88
Tabela 16 – Principais hiper-parâmetros de treinamento da arquitetura RCNN. . .	91
Tabela 17 – Predições realizadas pela RCNN no conjunto de treino.	92
Tabela 18 – Predições realizadas pela RCNN no conjunto de teste.	92
Tabela 19 – Cobertura da geração de candidatos por distância de edição.	94
Tabela 20 – Número médio de candidatos e potencial de correção para distância de edição.	95
Tabela 21 – Cobertura da geração de candidatos por LCS.	96
Tabela 22 – Número médio de candidatos e potencial de correção para LCS. . . .	96
Tabela 23 – Cobertura da geração de candidatos por <i>simlex</i>	97
Tabela 24 – Número médio de candidatos e potencial de correção para <i>simlex</i>	97
Tabela 25 – Resultados para correção de ruídos obtidos de cada modelo Word2Vec.	101
Tabela 26 – Palavras mais próximas à <i>simplesmente</i> em diferentes modelos de <i>embeddings</i>	102
Tabela 27 – Correções aprendidas pelo método de geração de candidatos por <i>embeddings</i>	103
Tabela 28 – Cobertura da geração de candidatos por <i>word embeddings</i>	103

Tabela 29 – Número médio de candidatos e potencial de correção para geração por <i>embeddings</i>	104
Tabela 30 – Cobertura obtida no ranqueamento de candidatos gerados por distância de edição.	105
Tabela 31 – Cobertura obtida no ranqueamento de candidatos gerados por complemento da razão da LCS.	106
Tabela 32 – Cobertura obtida no ranqueamento de candidatos gerados pelo modelo Word2Vec.	106
Tabela 33 – Cobertura obtida no ranqueamento de candidatos gerados pelo modelo Wang2Vec.	107
Tabela 34 – Cobertura obtida no ranqueamento de candidatos gerados pelo modelo FastText.	107
Tabela 35 – Principais hiper-parâmetros de treinamento da arquitetura <i>encoder</i> . . .	110
Tabela 36 – Avaliação da arquitetura <i>encoder</i> para correção de erros ortográficos contextuais.	110
Tabela 37 – Avaliação do <i>baseline</i> de frequência para correção de erros ortográficos contextuais.	111
Tabela 38 – Exemplo de ranqueamento de candidatos	115
Tabela 39 – Comparação de resultados entre as ferramentas Enelvo e UGCNormal no corpus TUB-UGC.	116

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
BRNN	<i>Bidirectional Recurrent Neural Network</i>
BTT	<i>Backpropagation Through Time</i>
CBOW	<i>Continuous bag-of-words</i>
CER	<i>Character Error Rate</i>
CGU	Conteúdo Gerado por Usuário
GRU	<i>Gated Recurrent Unit</i>
LCS	<i>Longest Common Subsequence</i>
LSTM	<i>Long Short-term Memory</i>
ML	Modelo de Língua
MLP	<i>Multilayer Perceptron</i>
MO	Mineração de Opinião
OOV	<i>Out-of-vocabulary</i>
PB	Português Brasileiro
PLN	Processamento de Língua Natural
RCNN	<i>Recurrent Convolutional Neural Network</i>
RNN	<i>Recurrent Neural Network</i>
RWE	<i>Real Word Error</i>
WER	<i>Word Error Rate</i>

SUMÁRIO

1	INTRODUÇÃO	1
1.1	Contextualização e Motivação	1
1.2	Objetivos	6
1.3	Organização da Monografia	7
2	CONTEÚDO GERADO POR USUÁRIO	9
2.1	Conceitos, Definições e Aplicações	9
2.2	Motivações do Usuário	12
2.3	A Linguagem do Conteúdo Gerado por Usuário	22
3	NORMALIZAÇÃO TEXTUAL	31
3.1	Fundamentos	32
3.1.1	<i>Representação de palavras</i>	32
3.1.2	<i>Representação de similaridade entre palavras</i>	36
3.2	Abordagens de normalização	38
3.2.1	<i>Baseadas em Léxico</i>	39
3.2.2	<i>Baseadas em Noisy-channel</i>	39
3.2.3	<i>Baseadas em Redes Neurais</i>	40
3.2.4	<i>Avaliação de Resultados</i>	46
4	TRABALHOS RELACIONADOS	49
4.1	Baseados em Léxico	49
4.1.1	<i>Trabalho de Clark e Araki (2011)</i>	49
4.1.2	<i>Trabalho de Duran, Avanço e Nunes (2015)</i>	51
4.1.3	<i>Trabalho de Stiilpen e Merschmann (2016)</i>	55
4.2	Baseados em Noisy-channel	57
4.2.1	<i>Trabalho de Brill e Moore (2000)</i>	57
4.2.2	<i>Trabalho de Choudhury et al. (2007)</i>	58
4.2.3	<i>Trabalho de Cook e Stevenson (2009)</i>	60
4.3	Baseados em Redes Neurais	61
4.3.1	<i>Trabalho de Min e Mott (2015)</i>	61
4.3.2	<i>Trabalho de Leeman-Munk, Lester e Cox (2015)</i>	63
4.3.3	<i>Trabalho de Ikeda, Shindo e Matsumoto (2016)</i>	64
4.4	Síntese dos Trabalhos Discutidos	66

5	CÓRPUS DE CGU	69
5.1	Cópus Buscapé	69
5.2	Cópus UGCNormal	71
5.3	Cópus TUB-UGC	72
6	MÉTODOS DE NORMALIZAÇÃO TEXTUAL	83
6.1	Considerações Iniciais	83
6.2	Métodos de Identificação de Ruídos	86
6.3	Métodos de Geração de Candidatos	93
6.4	Métodos de Ranqueamento de Candidatos	104
6.5	O Normalizador Enelvo	111
7	CONCLUSÕES E TRABALHOS FUTUROS	117
	REFERÊNCIAS	121
APÊNDICE A	MANUAL DO ANOTADOR DO CÓRPUS TUB-UGC	131
A.1	A Anotação na Prática	132
A.2	Categorias de Ruído	134
A.3	Resumo Passo-a-Passo	136
A.4	Algumas Definições	137

INTRODUÇÃO

1.1 Contextualização e Motivação

O crescimento massivo da Internet nos últimos anos revolucionou o compartilhamento de informação. A grande disponibilidade de computadores e o surgimento de dispositivos móveis – como *smartphones* e *tablets* – possibilitou que a Internet chegasse em larga escala à população. Esse alcance global tornou os meios de comunicação rápidos e acessíveis, acarretando uma capacidade sem precedentes de compartilhar informações.

A facilidade de se obter acesso à Internet criou espaço para o surgimento de novas mídias, diferentes das tradicionais como jornal, rádio e televisão. *Blogs*, páginas de notícias e *wikis* são alguns exemplos de meios de comunicação que surgiram na *web* e tornaram o compartilhamento de conhecimento mais democrático. O surgimento da Web 2.0 reforçou ainda mais esse processo de descentralização, visto que a criação de páginas mais interativas, que incentivavam a participação do usuário, começou a crescer. Esse tipo de página possibilitou aos usuários comuns interagirem com os veículos de comunicação em vez de apenas receber informações passivamente. O aumento de interatividade faz com que usuários da Internet tenham a oportunidade de atuar como criadores de conteúdo – e não apenas como consumidores.

Conteúdo gerado por usuário (CGU) é a denominação dada ao tipo de conteúdo criado por usuário comuns, que não têm associação com veículos midiáticos tradicionais (KRUMM; DAVIES; NARAYANASWAMI, 2008). Esses usuários da Internet contribuem voluntariamente com informações, muitas vezes compartilhadas em redes sociais. Diferentemente do conteúdo criado por profissionais pagos, o CGU é produzido de forma autêntica e tem o potencial de refletir o pensamento do público em geral. Essas características tornam CGU uma relevante fonte de conhecimento a ser analisada.

É possível encontrar CGU em diversas formas na *web*. Avaliações de usuários (*revi-*

ews) sobre produtos, comentários em *sites* de notícias, publicações em *blogs* e mensagens em redes sociais são alguns exemplos. Analisar esse tipo de conteúdo é útil tanto para organizações quanto para outros usuários. *Reviews* sobre produtos, por exemplo, permitem que lojas compreendam as características mais relevantes para os clientes e que consumidores tenham mais informações para decidir sobre uma compra. CGU está disponível na *web* em vastas quantidades, visto que seu compartilhamento é geralmente gratuito e pode ser feito por qualquer usuário. Sua alta disponibilidade e sua relevância fazem do CGU um candidato perfeito para ser explorado computacionalmente, principalmente para extração automática de conhecimento.

O Processamento de Língua Natural (PLN), de modo geral, lida automaticamente ou semiautomaticamente com a língua humana, escrita ou falada. Muito do CGU é disponibilizado de forma textual. *Blogs* pessoais, *wikis*, tutoriais sobre assuntos diversos e *reviews* são todos tipos de CGU publicados majoritariamente por meio de textos. Desse modo, é natural que haja interesse em utilizar ferramentas de PLN para processá-los. Essa interseção dá origem a aplicações interessantes e promissoras. A Mineração de Opinião (MO), subárea de PLN que visa a analisar computacionalmente opiniões, sentimentos e subjetividade presentes em textos, é um grande exemplo de como CGU por ser explorado (PANG; LEE, 2008). Apesar de ter suas origens na Computação Afetiva, em meados da década de 90, a MO prosperou a partir do advento do compartilhamento de CGU. Algumas aplicações de MO que fazem uso de CGU são: classificação de opinião de usuários sobre produtos, filmes, serviços e entidades; análise de publicações em redes sociais para identificar tendências políticas e econômicas; distribuição estratégica de propagandas em páginas *web* e muitas outras (LIU, 2012).

Além do exemplo da Mineração de Opinião, é possível notar que pesquisas recentes em PLN têm voltado seu foco ao CGU. A construção de *corpus* constituídos por esse tipo de conteúdo é um forte indicativo desse fato. Especificamente para o português brasileiro, há alguns exemplos relevantes:

- O *corpus* Resenhas de Livros (ReLi), composto por 2056 resenhas de 13 livros (7 autores), totalizando cerca de 300 mil palavras e 15 mil sentenças (FREITAS *et al.*, 2012). As resenhas foram extraídas do *website* Skoob¹, uma rede social de livros e leitores cujo objetivo é o compartilhamento de avaliações a respeito de livros lidos pelos usuários. O *corpus* possui anotação de polaridade de sentimento, positiva ou negativa, para cada avaliação e já foi utilizado em trabalhos de Mineração de Opinião (BRUM; ARAUJO; KEPLER, 2016; FILHO; PARDO; ALUISIO, 2013; BRUM; KEPLER, 2015).
- O *corpus* de *reviews* de produtos, descrito em Hartmann *et al.* (2014), é formado

¹ <skoob.com.br>

por avaliações escritas por consumidores a respeito de produtos diversos. As *reviews* foram extraídas de três *websites* que permitem compartilhamento desse tipo de texto e já foram utilizadas em diversos trabalhos de PLN (AVANÇO; BRUM; NUNES, 2016; ANCHIÊTA *et al.*, 2015; ALMEIDA *et al.*, 2016).

- O *corpus* 7x1-PT, constituído de publicações na rede social Twitter (denominadas *tweets*) durante o jogo entre Brasil e Alemanha na Copa do Mundo de 2014 da FIFA (MORAES; MANSSOUR; SILVEIRA, 2015). O *corpus* contém 2.728 *tweets* em língua portuguesa, totalizando 35.024 *tokens* e 4.925 *types*. Uma característica interessante do 7x1-PT é que, diferente de avaliações de produtos, os *tweets* a respeito do jogo não foram necessariamente publicados com o intuito de compartilhar informação. Muitos são simplesmente comentários informais, decorrentes da reação dos autores à partida. Para refletir essa noção, a anotação considerou quatro classes de polaridade: positiva, negativa, neutra e ironia. Desse modo, esse *corpus* possibilita a análise de opiniões por outra perspectiva.
- O *corpus* Computer-BR, composto por 2.317 *tweets* extraídos durante o ano de 2015 contendo palavras-chave relacionadas a computadores, tais como *notebook*, *análise* e *teste* (MORAES *et al.*, 2016). Como os autores optaram por não considerar ironia, as polaridades anotadas são positivas, negativas ou neutras. Esse *corpus* é mais um com publicações advindas do Twitter e mostra uma tendência em processar textos de redes sociais.

Com base nesses exemplos, é possível notar um número expressivo de recursos de CGU disponíveis. Embora o propósito original dos *corpus* seja a Mineração de Opinião, os textos carregam informações ricas e permitem a extração de conhecimento por diversas tarefas de PLN. Apesar de oferecer aplicações promissoras, na prática o conteúdo gerado por usuário é difícil de ser tratado por ferramentas linguístico-computacionais. Enquanto textos provenientes de mídias tradicionais, como revistas, são escritos por profissionais especializados e geralmente passam por um processo minucioso de revisão, textos criados por usuários comuns refletem a desatenção dos autores à qualidade de escrita. Ao gerar conteúdo, o foco do usuário é o conteúdo em si. Ao escrever uma avaliação sobre um produto, por exemplo, a intenção principal do autor é informar sua opinião a respeito do produto para potenciais leitores. Sendo assim, a forma que esse conteúdo é exposto não é um motivo de grande preocupação. Desse modo, textos de CGU tendem a apresentar uma linguagem muitas vezes descompromissada com a norma culta padrão.

Como CGU é criado de forma democrática, por qualquer pessoa com acesso à Internet, é esperado que haja autores de conteúdo com variados níveis culturais e sociais. Combinando isso ao fato de que CGU é em grande parte veiculado em redes sociais, que são plataformas altamente informais, textos desse tipo apresentam diversos desvios

em relação à língua considerada padrão. Para melhor ilustrar esse fenômeno, a [Figura 1](#) apresenta uma *review* de um produto escrita por um usuário comum.

Figura 1 – *Review* escrita por usuário da *web*.

eleh eh mtt daora para vc mecher entao eu recomendu conprarem ese sangsung
eleh otimo alem de ser barato eh dahora

Fonte: [Avanço \(2015\)](#).

Analisando a *review*, é possível notar vários desvios. Há presença de erros ortográficos (como *mecher* em vez de *mexer*), gírias (como *daora*) e expressões características do internetês (como *mtt* para designar *muito* e *vc* para *você*). A ausência de pontuação e de capitalização também denota a baixa qualidade do texto. Além de a letra inicial da sentença não estar capitalizada, o mesmo ocorre no nome próprio *sangsung*, versão incorreta de Samsung. Esses ruídos podem ser em grande parte justificados pela informalidade do meio, que incentiva o uso de linguagem próxima à utilizada no cotidiano, fortemente influenciada pela oralidade – ou seja, usuários da *web* tendem a escrever da maneira como falam.

Técnicas e ferramentas de PLN foram originalmente projetadas levando em consideração apenas textos escritos em linguagem padrão. Até recentemente, a maior parte dos corpúis disponíveis eram jornalísticos, ou seja, formados de textos revisados e bem escritos. Ao aplicar ferramentas construídas com base nesse tipo de corpúis aos textos de CGU, os resultados são muito inferiores. Um exemplo é o sistema de Stanford para reconhecimento de entidades nomeadas, que obtém 86% de *F-measure* no conjunto de teste do CoNLL² mas só atinge 44% no Twitter ([RITTER et al., 2011](#)). Muitas técnicas de PLN dependem de características linguísticas que só são consistentemente encontradas em textos que aderem à norma culta. Ao serem aplicadas a CGU, essas técnicas têm de lidar com uma elevada quantia de ruído e desvios da língua padrão a que foram destinadas a processar.

Abordagens tipicamente seguidas por tarefas de PLN são baseadas em regras, conhecimento linguístico e estatísticas de corpúis. No caso das regras, é difícil adaptá-las para CGU, pois a linguagem varia muito de acordo com o domínio e há pouca regularidade a ser extraída. Conhecimento linguístico é empregado com sucesso em diversas tarefas. No reconhecimento de entidades nomeadas, por exemplo, a presença de letras maiúsculas é um forte indício de nome próprio. Analisadores sintáticos (*parsers*) obtêm resultados superiores e funcionam de maneira mais consistente quando lidam com textos com pontuação bem estruturada ([READ et al., 2012](#)). No entanto, padrões linguísticos encontrados em textos que aderem à norma culta muitas vezes não estão presentes em

² <cnts.ua.ac.be/conll2003/ner/>

textos de CGU. Desse modo, não é trivial aplicar conhecimento linguístico tradicional a textos da *web*. Como visto na Figura 1, é comum que autores de CGU ignorem o uso de pontuação e de capitalização. Sendo assim, ferramentas tradicionais de PLN encontram dificuldades ao lidar com CGU e podem apresentar resultados insatisfatórios. O mau desempenho pode ocorrer de forma progressiva. Sistemas complexos para processamento de língua são muitas vezes compostos de módulos menores que têm objetivos específicos. Um etiquetador morfosintático (*tagger*), por exemplo, é frequentemente aplicado como passo intermediário em diversas tarefas de PLN. Se seu funcionamento não é bom o bastante, consequentemente o desempenho de módulos posteriores será afetado negativamente.

As abordagens estatísticas têm como princípio extrair conhecimento diretamente de *corpus* anotados. Algoritmos de Aprendizado de Máquina (AM) são usualmente a escolha para efetuar o processo de aprendizado. Nesse caso, conhecimento linguístico pode ser utilizado apenas na forma de *features* (informações adicionais sobre o problema que dão direcionamento ao aprendizado) ou ser desconsiderado – fazendo com que o algoritmo dependa totalmente das amostras anotadas. Para que o treinamento de métodos de AM ocorra com sucesso, são necessárias quantias consideráveis de dados. No caso de PLN, são necessários grandes *corpus* anotados (MANNING; SCHÜTZE *et al.*, 1999). Embora atualmente existam *corpus* de CGU, ainda são poucos e em menor tamanho, considerando o número de textos, se comparados aos grandes *corpus* jornalísticos. Portanto, a diversidade e a irregularidade linguística do CGU podem ser obstáculos que dificultam o processo de treinamento, visto que pode não haver amostras suficientes para que os algoritmos aprendam satisfatoriamente as nuances dos textos da *web*.

Apesar dos desafios, há maneiras de facilitar o processamento de textos ruidosos por ferramentas de PLN. Uma dessas maneiras é a normalização textual: o processo de transformar palavras fora do padrão em outras que são reconhecidas pela aplicação com que se deseja trabalhar (SPROAT *et al.*, 2001). De modo geral, a normalização textual visa a adaptar textos escritos com linguagem ruidosa a fim de torná-los mais próximos à língua padrão. Por padrão, geralmente entende-se a linguagem que atende à norma culta, aquela reconhecida oficialmente por dicionários e gramáticas. No entanto, esse padrão pode ser moldado conforme a aplicação que está sendo investigada. Ou seja, a normalização pode ser customizada de acordo com a tarefa específica, de modo que o padrão se torna a linguagem que pode ser processada adequadamente.

Não há consenso na literatura sobre o que é de fato normalização. O processo de normalização textual pode englobar correção ortográfica (BRILL; MOORE, 2000), capitalização (TIAN; DINARELLI; TELLIER, 2015), segmentação de sentenças (ZHANG *et al.*, 2013), substituição de gírias (CLARK; ARAKI, 2011), expansão de acrônimos (DURAN; AVANÇO; NUNES, 2015) e outras tarefas relacionadas. De modo geral, a normalização é desempenhada como um passo intermediário, visto muitas vezes como

uma etapa de pré-processamento. Por conseguinte, é comum que trabalhos considerem como normalização aquilo que foi necessário para a tarefa que se desejava abordar. Esse fato dificulta a formalização da área, fazendo com que haja pouco consenso sobre o que a tarefa de normalização deva desempenhar.

Como consequência, é difícil definir qual o estado-da-arte na área. Como cada trabalho trata normalização de uma maneira, as avaliações feitas são pouco replicáveis. Também há uma ausência de *corpus* de avaliação que atuem como *benchmarks* – ou seja, que sejam utilizados como comparação de diferentes trabalhos. Em geral, os sistemas de normalização são testados com *corpus* pequenos elaborados especificamente para a tarefa que necessita de normalização. Esses *corpus* dificilmente são disponibilizados, tornando ainda mais difícil a comparação de resultados e a replicação de experimentos. Ainda assim, há um pouco de consenso no tipo de texto utilizado. Usualmente, a normalização textual é empregada em SMSs, *reviews* de produtos e publicações em redes sociais, principalmente Twitter.

Especificamente para o português brasileiro, o trabalho de [Duran, Avanço e Nunes \(2015\)](#) é um dos poucos a tratar de normalização textual de conteúdo gerado por usuário. Nele, um normalizador de CGU focado em *reviews* de produtos é proposto. O sistema, denominado UGCNormal, realiza tokenização, segmentação de sentenças, correção ortográfica, expansão de acrônimos, capitalização de nomes próprios e substituição de internetês. Apesar de efetuar diversas tarefas, o domínio de aplicação é restrito, visto que o normalizador foi especialmente projetado para lidar com *reviews* de produtos do *corpus* descrito em [Hartmann et al. \(2014\)](#). Grande parte dos módulos é baseada em léxicos – ou seja, no uso de listas que contêm pares de palavras fora do padrão e suas respectivas correções. Embora funcione bem em um domínio restrito, essa abordagem não é escalável nem flexível, tornando-a inapta a lidar com outros tipos de CGU. Nesse contexto, é válido investigar com mais profundidade métodos de normalização de conteúdo gerado por usuário para o português brasileiro, e é a isso que esse projeto de pesquisa se propõe.

1.2 Objetivos

O projeto aqui descrito tem como objetivos gerais:

- Investigar e classificar os diferentes tipos de ruído em textos da *web* gerados por usuário;
- Desenvolver métodos de normalização textual genéricos e flexíveis, que possam ser adaptados para diferentes modalidades de CGU e para aplicações variadas;
- Avaliar profundamente o comportamento dos métodos na correção de diferentes tipos de ruído.

Considerando que a proposta envolve lidar com diferentes modalidades de CGU, foi necessária a anotação de um corpus de ruídos para efetuar a avaliação do sistema e dos métodos desenvolvidos. O corpus é constituído de *posts* em fóruns de discussão sobre assuntos diversos, de *reviews* de produtos e de publicações no Twitter. Os *posts* em fóruns foram escolhidos pois contemplam assuntos variados, com participação de usuários de diferentes origens sociais, culturais e com diversos níveis de instrução. As *reviews* foram escolhidas para manter a comparabilidade com o trabalho de [Duran, Avanço e Nunes \(2015\)](#), visto que esse é o domínio ao qual ele é restrito. *Tweets* foram selecionados por serem alvo de diversas pesquisas em PLN, serem facilmente obtidos em larga escala, e por representarem alta variedade linguística. A anotação do corpus seguiu a categorização proposta por [Hartmann et al. \(2014\)](#), de modo a manter a uniformidade nas pesquisas sobre normalização para o português brasileiro.

Como a área de normalização textual possui pouca padronização de técnicas e de métodos de avaliação, foi necessário explorar e implementar diferentes técnicas visando a analisar o impacto de cada uma na normalização dos diversos tipos de ruído. As principais questões de pesquisa que este trabalho pretende responder são:

1. Que tipos de ruídos estão presentes em textos de CGU e qual a frequência de cada um?
2. Como desenvolver métodos flexíveis de normalização que se adequem a diferentes modalidades de CGU?
3. Como desenvolver métodos de normalização com baixa dependência de recursos?
4. Qual a melhor maneira de adaptar o processo de normalização conforme as necessidades da aplicação em específico?
5. Em que medida combinar diferentes técnicas de normalização melhora os resultados obtidos?

1.3 Organização da Monografia

O [Capítulo 2](#) apresenta com profundidade as características do conteúdo gerado por usuário, tratando sobre motivações dos usuários ao produzi-lo e discutindo brevemente sobre redes sociais – principal plataforma de compartilhamento de CGU. Também são apresentadas aplicações que motivam pesquisas relacionadas a esse tipo de conteúdo e desafios para PLN ao lidar com sua linguagem ímpar. No [Capítulo 3](#), são introduzidos os principais conceitos de normalização textual, incluindo a fundamentação teórica necessária, as principais técnicas empregadas e os métodos mais comuns de avaliação. O [Capítulo 4](#) traz uma revisão e discussão crítica sobre os principais trabalhos da área e seus

resultados. No [Capítulo 5](#) os corpúsculos de CGU disponíveis para o português são apresentados e o corpúsculo que foi anotado como parte deste projeto é descrito detalhadamente. No [Capítulo 6](#) diversos métodos de normalização para identificação de ruídos e para geração e ranqueamento de candidatos são apresentados e avaliados. Por fim, as conclusões deste trabalho e perspectivas de trabalhos futuros são apresentadas no [Capítulo 7](#).

CONTEÚDO GERADO POR USUÁRIO

A criação e o compartilhamento de conteúdo são inerentes ao ser humano e acontecem há milênios. Pinturas rupestres, pergaminhos do Antigo Egito e a prensa de Gutenberg nos tempos modernos são exemplos de que a necessidade de transmitir conteúdo surgiu nos primórdios da humanidade e só aumentou com o passar do tempo. Embora a geração de conteúdo em larga-escala – como a por jornal, rádio e televisão – aconteça há tempos, a possibilidade de indivíduos comuns gerarem conteúdo de impacto por si só surgiu recentemente. O crescimento da Internet e das mídias sociais, entre outros desenvolvimentos tecnológicos, permitiu que indivíduos não relacionados a meios de comunicação pudessem gerar informação e influenciar um grande número de pessoas. Esse tipo de conteúdo, denominado conteúdo gerado por usuário (CGU), carrega informações valiosas e pode ser explorado em diversas áreas do conhecimento. Este capítulo tem como objetivo apresentar conceitos e definições de CGU, analisar as motivações do usuário ao produzir conteúdo, explorar os principais meios de divulgação de CGU (as mídias sociais) e avaliar porque é difícil lidar com tal tipo de conteúdo – principalmente no contexto do Processamento de Língua Natural.

2.1 Conceitos, Definições e Aplicações

O advento tecnológico da Web 2.0, a partir do início dos anos 2000, proporcionou uma mudança de paradigma na interação entre usuário e Internet. Em vez de páginas da Internet servirem apenas como documentos informativos, com os quais os usuários interagem passivamente, a Web 2.0 deu origem a páginas altamente interativas, com foco na participação do usuário. *Sites* como Facebook, Wikipédia, Youtube, Twitter, entre outros, permitem que os usuários se tornem criadores e consumidores ativos de conteúdo. A proliferação global da Internet e a alta disponibilidade de dispositivos móveis acarretaram uma explosão de conteúdo gerado por usuário. Atualmente, esse tipo de conteúdo já

transpassou suas mídias de origem (principalmente as redes sociais) e faz parte de meios tradicionais de comunicação. Cada vez mais jornais e programas de televisão abrem espaço para que os consumidores também sejam produtores de conteúdo. Essa interação, que pode ser considerada como pós-Web 2.0, abre espaço para diversas pesquisas que envolvam CGU.

De modo geral, o termo “conteúdo gerado por usuário” se refere ao conteúdo criado pelo público comum, em vez de profissionais pagos, e é principalmente distribuído pela Internet (DAUGHERTY; EASTIN; BRIGHT, 2008). Uma definição um pouco mais específica, dada por Krumm, Davies e Narayanaswami (2008), diz que CGU origina-se de indivíduos comuns, que voluntariamente contribuem com dados, informações e mídias, que são então divulgados em meios principalmente digitais, e posteriormente consumidos por outros indivíduos. Os meios de divulgação incluem redes sociais, páginas com avaliações de produtos, wikis, *blogs* pessoais e muitos outros. O uso de CGU para fins de pesquisa, por organizações e por usuários finais, cresce constantemente. Para organizações e para usuários, o fato de CGU ser produzido de forma autêntica – ou seja, sem intervenção de profissionais pagos – permite sua utilização de forma ampla na tomada de decisão. Opiniões de usuários sobre determinado produto, por exemplo, são relevantes tanto para empresas quanto para consumidores interessados em adquiri-lo. Ademais, por ser em grande parte disponível gratuitamente e existir em grandes quantidades, CGU pode ser adequado para formar bases de dados para pesquisas em Inteligência Artificial.

Para compreender a imensidão do conteúdo disponível *online*, as informações obtidas do InternetLiveStats¹, página que disponibiliza estatísticas sobre uso da Internet, são de grande valia. Segundo a página, atualmente cerca de 40% da população mundial tem acesso à Internet. Em 1995, esse valor era menos de 1%. Apenas no período de quatorze anos entre 1999 e 2013, o número de usuários da Internet aumentou dez vezes. A quantidade de conteúdo gerado e consumido *online* é impressionante. Estima-se que 6000 *tweets* sejam publicados por segundo – totalizando 500 milhões diariamente. O Twitter, lançado em 2006, levou três meses para atingir a marca de um bilhão de *tweets* enviados. Atualmente, esse valor é alcançado em dois dias. Já no YouTube, cerca de 67726 vídeos são visualizados por segundo. Esses valores mostram que o alcance e o impacto do conteúdo disponível *online*, que muitas vezes é gerado por usuários, são imensos. Essa vasta quantia de dados é frequentemente disponibilizada de forma gratuita e pode ser acessada publicamente. Desse modo, pode ser uma fonte ideal para aplicação de métodos de Inteligência Computacional, como algoritmos de aprendizado de máquina. Essas características possibilitam a intersecção de CGU com diversas áreas da computação, inclusive com PLN.

Além da imensa quantia de dados disponível, a variedade de conteúdo é um fato que reforça a riqueza de informação contida no CGU. Apenas em se tratando da *web*, já

¹ <internetlivestats.com>

é possível identificar diversos meios de geração e divulgação de conteúdo. Redes sociais, como Facebook e Twitter, embora tenham como função principal a interação social entre membros, constituem berços de discussões ricas de informação sobre vários assuntos – como política e acontecimentos mundiais. Páginas específicas para compartilhamento de vídeos e imagens, como YouTube e Flickr, hospedam conteúdo diverso que frequentemente é visualizado por milhões de pessoas. Além do conteúdo em si, os comentários dos usuários proporcionam alta interatividade e conseqüentemente novas informações relevantes. Lojas virtuais frequentemente têm espaço para que os usuários avaliem os produtos, possibilitando desse modo a abertura de um amplo nicho de análise envolvendo essas avaliações. A *web* também abriga *wikis*, *blogs* pessoais, páginas para compartilhamento de receitas, músicas, dicas de viagem e uma infinidade de conteúdo.

Indo além de exemplos já bem conhecidos e presentes no cotidiano das pessoas, CGU também pode ser encontrado em outros meios não tão óbvios. A popularização de dispositivos móveis, como *smartphones* e *smartwatches*, possibilita aos usuários a criação de conteúdo pervasivo. Um exemplo que ilustra esse conceito é o projeto OpenStreetMap, cujo objetivo é criar e disponibilizar gratuitamente mapas de estradas baseados em informações de GPS enviadas por voluntários (KRUMM; DAVIES; NARAYANASWAMI, 2008). Outros exemplos do uso de CGU de maneira pervasiva são: análise da circulação de notas de dinheiro para descobrir padrões de movimentação humana; análise de *check-ins* e de marcações em fotos de turistas em Roma para entender como exploram a cidade, e até mesmo instalações em museus que permitem com que usuários enviem mensagens de texto, vídeo e áudio para formar uma grande obra de arte colaborativa (KRUMM; DAVIES; NARAYANASWAMI, 2008).

A combinação de vasta quantia e de grande variedade de dados disponíveis torna o CGU um candidato promissor para ser utilizado computacionalmente para extração de conhecimento. As informações contidas nesse tipo de conteúdo podem ser valiosas para empresas, organizações, instituições governamentais e para outros usuários. Discussões políticas em fóruns da *web* podem ser analisadas para prever o resultado de eleições ou mensurar a popularidade do governo. Avaliações de usuários sobre produtos são úteis para potenciais consumidores e para os fabricantes – que podem melhor entender quais características de seu produtos são mais importantes para o consumidor. Publicações no Twitter podem ser utilizadas para identificar fenômenos mundiais impactantes de modo a alertar as autoridades. Vídeos do Youtube, artigos da Wikipédia e receitas compartilhadas *online* podem ser fonte de conhecimento e facilitar a vida de milhões de pessoas que acessam tal conteúdo. As possibilidades de aplicação de CGU são diversas, porém sua totalidade não pode ser explorada simplesmente pelo consumo do conteúdo. Considerando a imensa quantia de informação produzida, há muito conhecimento relevante a ser extraído. Esse processo de extração de conhecimento pode ser desempenhado de maneira exemplar por técnicas computacionais, especialmente as relacionadas à Inteligência Artificial. Nesse

contexto, o PLN pode ser empregado para analisar um grande nicho de CGU.

Muito do conteúdo gerado por usuários é disponibilizado em forma de texto – ou seja, língua humana escrita. Avaliações de produtos, comentários em fóruns sobre filmes e discussões em redes sociais são alguns exemplos. Mesmo quando o meio é focado em mídias de outros formatos, como vídeos, é comum que haja espaço para discussão entre usuários – que aí sim ocorre textualmente. Frequentemente comentários advindos de discussões sobre determinado conteúdo trazem informações relevantes sobre o conteúdo em si. Desse modo, é possível que comentários sobre um vídeo no Youtube ajudem a entender o próprio vídeo. Assim, o PLN pode atingir até mesmo meios não textuais. Considerando tudo isso, é de se esperar que haja alguma intersecção entre PLN e CGU. De fato, nos últimos anos muitos projetos de pesquisa que envolvem processamento textual são aplicados a textos provenientes de CGU. Um exemplo é a Análise de Sentimentos (ou Mineração de Opinião), uma subárea de PLN que abrange o estudo da subjetividade e de opiniões, sendo que uma de suas principais tarefas é classificar a polaridade de opiniões. O crescimento da geração de conteúdo na Web está intimamente ligado ao desenvolvimento da análise de sentimentos. Muito do CGU contém opiniões, sendo que a principal fonte de conteúdo opinativo são as avaliações – que podem ser sobre produtos, filmes, músicas ou qualquer outra entidade sobre qual usuários possam opinar. Sendo assim, a análise de sentimentos pode fazer uso de CGU em diversas aplicações práticas, como: previsão de desempenho de vendas, análise de postagens no Twitter para entender o comportamento do mercado, distribuição estratégica de propagandas em páginas Web, previsão de resultados das eleições e análise de posicionamento partidário (AVANÇO, 2015). Embora as aplicações sejam promissoras, lidar com CGU oferece desafios para PLN. A linguagem presente nesse tipo de conteúdo frequentemente contém desvios da língua padrão, acarretando dificuldades para métodos tradicionais de PLN, que são geralmente baseados em textos formais e bem escritos. Esses problemas serão discutidos com profundidade na Seção 2.3.

2.2 Motivações do Usuário

Um ponto positivo no conteúdo gerado por usuários é a sua suposta autenticidade, que permite análises isentas da influência de pessoas pagas para produzir conteúdo. Ao efetuar tal análise, é possível detectar informações efetivamente importantes ao público em geral. No entanto, seria seguro assumir que CGU é de fato autêntico? Para responder essa questão, é necessário analisar as motivações do usuário ao produzir conteúdo. Ao se compreender o que leva um usuário a produzir determinado conteúdo, pode-se entender melhor as nuances envolvidas nesse conteúdo – permitindo assim uma melhor extração de conhecimento. O trabalho de Trammell, Hofomokl e Sapp (2006) nos permite ter noção das motivações dos usuários a produzirem conteúdo. Nele, é realizada uma análise de 358 postagens em *blogs* de língua polonesa visando identificar as principais motivações em

publicá-las. Os seis motivos mais frequentes são: Auto-expressão; Interação social; Entretenimento; “Passar o tempo”; Obter informação; Desenvolvimento profissional. Embora apresente resultados advindos de CGU na prática, a análise ainda é superficial. Para melhor entender as motivações do usuário é necessário ir além.

De acordo com [Daugherty, Eastin e Bright \(2008\)](#), os meios de comunicação tradicionais têm se segmentado e perdido espaço para mídias mais interativas. Nesse mercado com forte segmentação, meios de comunicação formados majoritariamente por CGU em vez dos veiculados por grandes emissoras e editoras têm emergido. Com essas mídias, usuários comuns conseguem ter voz em meio a um turbilhão de informação e propaganda. Esse é possivelmente um incentivo à criação de conteúdo. O trabalho de [Daugherty, Eastin e Bright \(2008\)](#) busca embasamento na teoria funcional de [Katz \(1960\)](#) para entender as motivações dos usuários. Para os autores, essa teoria é considerada por muitos como essencial para compreender a motivação, muitas vezes complexa, que leva a atitudes. Segundo [Katz \(1960\)](#), qualquer atitude obedece a uma ou mais das quatro funções de personalidade: a utilitária; a de conhecimento; a ego-defensiva e a de expressão de valor. Considerando essas funções, [Daugherty, Eastin e Bright \(2008\)](#) as contextualizam na motivação para criação de CGU e as descrevem como:

- **Utilitária:** baseia-se no fato de que as pessoas são motivadas, por interesse próprio, a receber recompensas e a evitar punições em seu ambiente. No contexto de CGU, isso indica que usuários criam conteúdo primariamente por incentivo próprio, ou seja, para se sentirem bem consigo mesmos.
- **De conhecimento:** essa função indica que as pessoas são movidas pela necessidade de buscar informações que as auxiliem a entender e organizar seu ambiente e a melhor compreender suas experiências pessoais. Sendo assim, criadores de CGU são motivados a criar conteúdo para aprender sobre um assunto, sobre seu ambiente ou até mesmo sobre si mesmos – pois o compartilhamento de informações provê bem-estar pessoal e sabedoria interna.
- **Ego-defensiva:** representa a necessidade humana de proteção sobre inseguranças internas ou de ameaças externas, sendo que internamente sua função principal é a defesa da imagem-própria. Nesse caso, os criadores de CGU geram conteúdo para diminuir suas dúvidas internas, sentir como se pertencessem a algo, e possivelmente para evitar com que se sintam mal por não contribuir.
- **De expressão de valor:** engloba atitudes que permitem com que as pessoas expressem seus valores, melhorando assim suas imagens perante o mundo. Desse modo, criadores de CGU podem se sentir gratos por terem criado conteúdo e por fazerem parte de uma comunidade que compartilha princípios considerados importantes por

eles. Isso valida seus valores, os ajuda a se sentirem melhores consigo mesmos e consolida suas visões de mundo.

Embora essas quatro funções possam ser consideradas como o núcleo constituinte da motivação dos usuários, ainda há outros pontos a serem explorados. O trabalho de [Leung \(2009\)](#) analisa os motivos de criação de conteúdo sob a ótica do empoderamento psicológico e do engajamento cívico. Empoderamento é o processo que permite com que pessoas assumam o controle de suas vidas, melhorem seus pontos fortes e competências e desenvolvam comportamentos proativos que auxiliam nas relações sociais ([KATZ, 1984](#)). Uma maneira de desenvolver o empoderamento psicológico é tornando-se envolvido em decisões que afetem a vida em comunidade, como por meio da contribuição de ideias e opiniões em um contexto público e social. Engajamento cívico pode ser definido como um processo no qual indivíduos participam na tomada de decisões sobre instituições, projetos e ambientes que os afetam ([LEUNG, 2009](#)). Há uma conexão clara entre empoderamento psicológico e engajamento cívico. Pessoas engajadas em suas comunidades frequentemente apresentam características de empoderamento. O estudo conduzido por [Leung \(2009\)](#) visa encontrar associação entre esses dois conceitos e a criação e consumo de CGU.

O estudo foi realizado com base em entrevistas feitas com 626 usuários da Internet com idades entre 14 e 70 anos. As entrevistas tinham como objetivo mensurar diversas características, sendo elas:

- **Gratificação de geração de conteúdo na Internet:** visa identificar a gratificação que os entrevistados sentem ao fazer uso de itens como meios de comunicação (como televisão), computadores pessoais, telefones celulares, a internet, serviços de mensagens instantâneas, entre outros.
- **Empoderamento psicológico:** foi medido por meio de uma escala de cinco itens, com base na concordância, ou não, de afirmações como “Posso permanecer calmo ao lidar com situações adversas pois confio em minhas habilidades” e “Não importa o que apareça em meu caminho, geralmente sou capaz de lidar com isso.”
- **Conteúdo gerado por usuário:** visa medir o consumo de CGU pelos entrevistados. Os participantes do estudo tiveram de responder a: “Durante os últimos seis meses, quanto tempo por dia você gastou em atividades que envolvem CGU, como: páginas pessoais, *blogs*, fóruns, postagem de vídeos no Youtube e contribuição de conteúdo na Wikipédia?” A escala utilizada foi: 1 = 15 minutos ou menos; 2 = 16–30 minutos; 3 = 31–59 minutos; 4 = 60–120 minutos; 5 = 121 minutos ou mais.
- **Atenção à mídia:** os entrevistados tiveram de responder com que frequência semanal usavam revistas, jornais, rádio e televisão para acompanhar acontecimentos

globais e políticos. A frequência foi categorizada entre 0 = nenhum dia e 7 = sete dias.

- **Eficácia política externa:** os entrevistados tiveram de concordar ou discordar com as afirmações: “Eu não acho que oficiais públicos se importem muito com o que pessoas como eu pensam” e “Pessoas como eu não têm nada a dizer sobre o que o governo faz.”
- **Eficácia política interna:** os entrevistados tiveram de concordar ou discordar com as afirmações: “O voto é a única maneira que pessoas como eu têm de dizer algo sobre o que o governo faz” e “Às vezes a política e o governo parecem tão complicados que uma pessoa como eu não consegue realmente entender o que se passa.”
- **Interesses em assuntos públicos:** medido com base na resposta a “Algumas pessoas pensam sobre o que está acontecendo no governo na maior parte do tempo, independente de estar havendo eleições; outras, não são tão interessadas. Você se interessa pelo governo na maior parte do tempo (4), às vezes (3), só de vez em quando (2) ou raramente (1)?”
- **Informações demográficas:** informações pessoais como idade, gênero, salário e nível educacional foram perguntadas e registradas.

O objetivo principal do estudo era encontrar correlações entre todas as características avaliadas nas entrevistas. Embora muitos resultados tenham sido obtidos, apenas os mais relevantes a este trabalho serão apresentados. Primeiramente, os dados sobre consumo de CGU permitem compreender o alcance dessas mídias ao público comum. A Tabela 1 apresenta a porcentagem de participantes do estudo que participaram de atividades relacionadas a algum tipo de CGU e o tempo médio diário gasto na atividade, de acordo com a escala apresentada anteriormente.

Tabela 1 – Dados sobre participação em atividades relacionadas a CGU

Tipo de CGU	Porcentagem de participação	Tempo médio diário
Fóruns	73.8	2.42
<i>Blogs</i>	65.5	2.62
Wikipédia	38.1	2.43
Páginas pessoais	26.7	2.51
Youtube	20.8	2.29

Fonte: Adaptada de [Leung \(2009\)](#).

É importante ressaltar que a participação em atividades envolve tanto consumo quanto produção de CGU. Sendo assim, os resultados obtidos são expressivos. No geral, os participantes passam em média de 16 a 30 minutos por dia envolvidos nessas atividades.

Outro resultado altamente relevante é o que apresenta os motivos que fazem com que usuários participem em geração de conteúdos *online*. As motivações são divididas em quatro categorias – necessidades de reconhecimento, cognitivas, sociais e de entretenimento –, sendo que cada uma contém diversos itens. As principais motivações dos usuários para gerar conteúdo *online* identificadas pelo estudo são:

- **Necessidades de reconhecimento:**

1. Para estabelecer identidade pessoal
2. Para ganhar respeito e suporte
3. Para aumentar a confiança
4. Porque é prazeroso
5. Para promover ou fazer propaganda de conhecimento técnico

- **Necessidades cognitivas:**

1. Para expandir conhecimento
2. Para descobrir acontecimentos sociais
3. Para entender eventos recentes
4. Para refinar o pensamento

- **Necessidades sociais:**

1. Para expressar sentimentos
2. Para compartilhar experiências, visões de mundo e pensamentos
3. Para informar família e amigos sobre novos acontecimentos pessoais

- **Necessidades de entretenimento:**

1. Para passar tempo
2. Por curiosidade
3. Porque é divertido
4. Porque está na moda

Os resultados são similares aos descritos por [Daugherty, Eastin e Bright \(2008\)](#), porém os complementam com mais profundidade, adicionando novas dimensões a serem consideradas. Pode-se concluir que grande parte da motivação advém do próprio usuário. Mesmo as motivações de cunho social podem ter raiz interna e podem ser explicadas por motivos pessoais. No entanto, ainda podem existir fatores externos que influenciam fortemente os usuários. Existem meios tão difundidos e populares que por si sós são capazes

de motivar usuários a gerar conteúdo. Um grande exemplo que ilustra esse conceito são as mídias sociais.

Segundo [Kaplan e Haenlein \(2010\)](#), mídias sociais são grupos de aplicações *online* construídas sobre fundamentos tecnológicos e ideológicos da Web 2.0, e que permitem criação e compartilhamento de conteúdo gerado por usuário. Embora sirvam como repositório de CGU, vale ressaltar que mídias sociais são por si só uma plataforma de compartilhamento de informação. Desse modo, podem haver interação com usuários sem necessariamente haver geração de conteúdo. O Facebook, por exemplo, embora possa hospedar grupos de discussão sobre variados assuntos, também pode servir simplesmente como uma ferramenta para contato familiar. Ainda assim, mídia sociais são frequentemente plataformas atrativas aos usuários, que incentivam a geração de conteúdo. Para compreender esse fenômeno, primeiramente é necessários analisar os diferentes tipos de mídias sociais. Os autores propõem uma divisão em seis categorias, cada uma com atrativos próprios para criação de CGU. As quatro mais importantes para o escopo deste trabalho são ([KAPLAN; HAENLEIN, 2010](#)):

- **Projetos colaborativos:** são projetos que permitem a criação de conteúdo por diversos usuários de forma simultânea e conjunta. Pode ser considerada uma das formas mais democráticas de CGU, visto que vários usuários podem participar igualmente. Um exemplo popular desse tipo de mídia social são as *wikis*, como a Wikipédia, que está disponível em mais de 230 idiomas e é frequentemente utilizada como fonte de informação na *web*. A principal motivação para criar conteúdo em projetos colaborativos é o fato de que o esforço conjunto entre diversas pessoas tende a levar a resultados superiores aos de ações individuais. Desse modo, os usuários se sentem pertencentes a uma comunidade e valorizados por contribuírem com conteúdo que será útil a outras pessoas.
- **Blogs:** são páginas *web* para compartilhamento de conteúdo por meio de *posts*, que são geralmente dispostos em ordem cronológica reversa. *Blogs* possuem diversos propósitos, podendo tratar sobre experiências pessoais do criador ou assuntos específicos – como culinária, esportes e filmes. Geralmente são gerenciados por apenas uma pessoa, mas cada vez mais surgem *blogs* que reúnem diversos autores, atuando assim como um grande meio de geração de conteúdo. Além dos tradicionais *posts* textuais, há os que hospedam conteúdo composto por imagens e vídeos. O formato de *posts* em vídeos tem se tornado altamente popular, especialmente por meio da plataforma YouTube, na qual há autores que atingem marcas de milhões de visualizações de seus vídeos. Apesar de o intuito ser a geração de conteúdo informativo, o aspecto social dos *blogs* advém dos comentários. É usual que haja espaço para discussão e para comentários de leitores nos *posts*, formando consequentemente uma mídia social. As principais motivações de autores de *blogs* são: exposição de conheci-

mento técnico, satisfação ao serem lidos (ou assistidos), criação de rede de contatos com pessoas com interesses afins e realização pessoal.

- **Comunidades de distribuição de conteúdo:** tem como objetivo o compartilhamento de mídia entre os participantes. Não envolvem a geração de conteúdo propriamente dito, pois seu foco é no compartilhamento de conteúdo já existente. Essas comunidades envolvem a distribuição de diversos tipos de mídia, incluindo fotos (Flickr²), vídeos (YouTube) e *slides* de apresentações (SlideShare³). Um cenário que melhor ilustra o funcionamento desse tipo de mídia pode ser descrito, tomando como base o YouTube, do seguinte modo: um usuário pertencente a uma comunidade de fotógrafos reúne diversos vídeos tratando sobre técnicas de fotografia, os seleciona e cria uma *playlist* que será compartilhada com os demais membros da comunidade. Apesar de os vídeos não terem sido criados pelo responsável pela compilação, a *playlist* em si ainda pode ser considerada como uma forma de conteúdo gerado por usuário, visto que a reunião e a seleção de mídias é personalizada e traz informações particulares ao autor. Uma vez compartilhada na comunidade, os demais membros podem interagir, opinar e eventualmente modificar a *playlist* original. Sendo assim, há um claro aspecto de mídia social. Nesse meio de geração de conteúdo, as motivações são análogas às dos *blogs*, mas são mais focados em compartilhamento de informação do que em aspectos pessoais.
- **Redes sociais:** talvez a forma mais popular desse tipo de mídia, as redes sociais têm como objetivo fomentar a interação entre usuários. Cada membro de uma rede cria um perfil, no qual pode compartilhar várias informações pessoais. Também existem páginas ou grupos tratando sobre temas diversos, incluindo assuntos específicos, celebridades, entidades, etc. A interação se dá por meio da troca de mensagens, de visitas aos perfis, de “curtidas” em comentários, de postagens em grupos e diversas outras formas. No Facebook, por exemplo, os usuários podem compartilhar uma infinidade de conteúdo em seus perfis, incluindo vídeos, fotos, reportagens e *posts* de *blogs*. A seleção de o que um usuário compartilhará em sua página envolve muito de sua personalidade – ou seja, pode ser considerado, além de simples compartilhamento de informação, uma forma de autoexpressão. Desse modo, nessa mídia as principais motivações são relacionadas aos conceitos de empoderamento e engajamento, sendo que a força motriz para geração de conteúdo é a exposição pessoal e o fortalecimento da imagem própria perante os demais usuários.

É possível notar que as mídias sociais são um ambiente altamente propício para geração de conteúdo, com vários incentivos para os usuários. Outro fato interessante é

² <flickr.com>

³ <slideshare.net>

que uma única plataforma pode se encaixar em diferentes categorias de mídia social. Consequentemente, a mesma plataforma pode atrair usuários de diferentes maneiras e motivá-los a produzir variados tipos de conteúdo. O YouTube, por exemplo, pode ser considerado uma plataforma de *blogs*, pois permite que usuários postem vídeos autorais, e uma comunidade de distribuição de conteúdo, pois possibilita a criação de *playlists*. Desse modo, é relevante analisar com mais profundidade o que constitui uma mídia social. O que torna o YouTube uma plataforma multifacetada? Por que usuários criam determinados conteúdos em plataformas sociais específicas? Essas perguntas só podem ser respondidas por meio de um estudo minucioso dos constituintes das mídias sociais.

Um possível ponto de vista para analisar as mídias sociais é o da Teoria Funcional (KATZ, 1960). Analogamente a este trabalho de Katz, que explica motivações por meio da atribuição de funções à personalidade, é possível atribuir funções às plataformas sociais. O trabalho de Kietzmann *et al.* (2011) segue essa abordagem e propõe que mídias sociais são formadas por sete blocos funcionais – que dão origem à denominação escolhida pelos autores: a colmeia das mídias sociais. A Figura 2 ilustra a representação da colmeia, sendo que cada bloco indica uma funcionalidade. Embora cada um tenha significado individualmente, é a colmeia como um todo que de fato representa a complexidade de uma mídia social.

Figura 2 – A colmeia de blocos funcionais de uma mídia social.

Fonte: Elaborada pelo autor.

Segundo Kietzmann *et al.* (2011), cada bloco permite destrinchar e examinar um ponto de vista específico da experiência dos usuários em determinada mídia social. Eles não são mutuamente exclusivos e não fazem parte necessariamente de todo tipo de mídia social. Desse modo, plataformas específicas têm influência diferente de cada bloco, de modo que é possível caracterizá-las com base nessas funcionalidades. A função de cada bloco pode ser melhor descrita como:

- **Identidade:** esse bloco funcional indica até que ponto os usuários revelam suas identidades pessoais nas mídias sociais. Esse conceito engloba desde informações básicas (nome, idade, gênero) até informações mais íntimas que podem caracterizar a personalidade de um usuário (gosto musical, preferências culinárias, ídolos). Em redes sociais (como Facebook) a identificação é fundamental, visto que primariamente têm função de criação de um perfil para o usuário. Consequentemente, conteúdo gerado nessas plataformas tende a ser de cunho mais pessoal, com motivação principalmente pautada em autopromoção, autoafirmação e consolidação de imagem pessoal.
- **Conversação:** indica até que ponto um usuário de determinada mídia social se comunica com outros usuários. Muitas plataformas têm como objetivo principal promover a comunicação e a interação entre usuários e grupos. Esse processo de comunicação pode ocorrer por meio de mensagens diretas, *tweets*, *posts* em *blogs* e em fóruns de discussão, entre outros. Os usuários buscam a comunicação por diversos motivos, como: para expor suas ideias e ouvir opinião de outras pessoas, para encontrar amigos e para melhorar a autoestima. Em geral, essa motivação está relacionada à construção de comunidade, ou seja, os usuários sentem a necessidade de pertencer a algo. Conteúdo produzido em plataformas com forte influência da conversação tende a ser focado em compartilhamento de ideias e na noção de grupo.
- **Compartilhamento:** representa a extensão da troca, distribuição e geração de conteúdo entre usuários. Diferentemente do bloco de conversação, o compartilhamento assume que a interação entre usuários ocorre pelo fato de que desejam compartilhar algum tipo de mídia. Comunidades de distribuição de conteúdo são predominantemente influenciadas por esse bloco, visto que são focados na distribuição de conteúdo entre membros. É importante ressaltar que compartilhamento não obrigatoriamente implica em conversação. No Facebook, por exemplo, usuários podem compartilhar vídeos em grupos específicos sem que haja a interação de nenhum outro usuário. Desse modo, a principal motivação ao gerar conteúdo é de fato o compartilhamento de informação, acima dos motivos pessoais.
- **Presença:** trata-se da informação na mídia social que permite que um usuário saiba se outros usuários estão presentes. Em redes sociais, por exemplo, essa informação pode ser dada por meio de *status* “disponível” ou “oculto”. O conceito de presença pode ainda se estender para a vida real. Os *check-ins*, mecanismos que indicam a localização física do usuário, são frequentemente utilizados no Facebook. Um exemplo de seu uso é na escrita de avaliações sobre restaurantes, nas quais usualmente o autor faz *check-in* para indicar que estava presente no restaurante avaliado – e assim dar respaldo à sua opinião. O bloco de presença pode ser visto como uma extensão do de identidade, pois permite com que o usuário compartilhe informações pessoais

íntimas, como sua localização física. Desse modo, geração de conteúdo em mídias baseadas em presença é principalmente motivada por fatores pessoais dos usuários, como autopromoção e necessidade de reconhecimento.

- **Relacionamentos:** indica até que ponto os usuários podem se relacionar. O relacionamento envolve qualquer forma de associação entre usuários – compartilhamento de informações, conversas, encontros pessoais, ou simplesmente a adição como amigo em uma rede social. Enquanto o bloco de conversação trata apenas de comunicação intra-mídia social, o relacionamento também engloba interações que têm capacidade de ultrapassar os limites virtuais e se estender à realidade dos usuário. Sendo assim, plataformas influenciadas por esse bloco funcional tendem a incentivar a geração de conteúdo colaborativo, com características motivadas por interação social, inclusão em grupos e satisfação pessoal do autor por fazer parte de “algo maior”.
- **Reputação:** representa a possibilidade de usuários de uma mídia social terem destaque perante outros usuários. Em plataformas sociais, a reputação é geralmente mensurada por informações numéricas – como quantia de seguidores de um perfil, curtidas de uma página e visualizações de um vídeo. Embora essas medidas não assegurem qualidade de conteúdo, são comumente vistas pelos usuários como forma de mérito. Sendo assim, mídias sociais pautadas em reputação motivam a geração de conteúdo que muitas vezes visa a atingir números elevados nas medidas de reputação. Consequentemente, as motivações dos autores são relacionadas a autopromoção e satisfação pessoal.
- **Grupos:** indica a capacidade de formação de grupos que a mídia social oferece aos usuários. De modo geral, quanto maior a possibilidade de interação social em uma plataforma, maior o incentivo à formação de grupos. Esses grupos podem ser sobre interesses específicos, análogos a clubes na vida real, ou somente um agrupamento de pessoas conhecidas – como círculos de amigos e familiares. Esse bloco funcional pode atuar de modo semelhante ao de compartilhamento, quando usuários publicam informações em grupos de assuntos específicos, ou como o de relacionamento no caso de grupos voltados a relações interpessoais.

Com base na colmeia dos blocos funcionais, é possível compreender melhor a constituição de uma plataforma social. Cada bloco oferece influências diferentes, que consequentemente impactam nas motivações dos usuários ao produzirem conteúdo. Por esse motivo, diferentes mídias sociais incentivam diferentes tipos de conteúdo. Cada mídia social pode ser influenciada por variados blocos e em diferentes graus de influência. Ou seja, desde que um bloco influencie uma plataforma e essa influência não seja determinante (não essencial ao seu funcionamento), cada usuário pode interpretá-la de uma maneira – dando origem a características multifacetadas de certas plataformas.

Como visto, motivações dos usuários para gerar conteúdo se dão nas mais diversas formas. Satisfação pessoal, vontade de se destacar em um grupo, de compartilhar informações relevantes com outros usuários são alguns exemplos. A maioria das motivações está relacionada ao próprio usuário, aos usuários que consumirão o conteúdo ou ao conteúdo em si – ou seja, sua qualidade. No entanto, há pouca preocupação com o meio ou com a forma com que o conteúdo é transmitido. Para conteúdo textual, por exemplo, a forma e a correção dos textos geralmente não são de grande relevância para os usuários. Por ser divulgado em grande parte em mídias sociais, o CGU tende a se adequar ao estilo da mídia, consequentemente tende a ser informal. No caso de conteúdo textual, essa informalidade pode ser vista por meio do não atendimento às normas padrão da língua, dando origem a uma linguagem original e específica a CGU. Essa linguagem traz desafios ao PLN e necessita ser analisada e tratada meticulosamente, dado o risco de comprometer o funcionamento de sistemas que venham a processá-la.

2.3 A Linguagem do Conteúdo Gerado por Usuário

Conteúdo gerado em mídias tradicionais, como jornais, geralmente passa por um processo de revisão para garantir a qualidade do texto. Sendo assim, é usual que esse tipo de texto atenda totalmente às normas padrão da língua, sendo isento de erros e desvios. Já nas mídias sociais, nas quais o conteúdo é criado frequentemente por e para usuários comuns, o processo de revisão raramente ocorre. Desse modo, os textos publicados comumente contêm desvios da língua padrão – como erros ortográficos e ausência de pontuação. Essa linguagem encontrada nos textos de CGU é fortemente ligada à língua informal utilizada no cotidiano, portanto muitos de seus desvios remetem à oralidade. Para exemplificar, a Figura 3 apresenta uma publicação no Twitter.

Figura 3 – Uma publicação no Twitter utilizando linguagem informal.

**As frustrações de hj, usamos como
esperiencia para conceguirmos coisas
maiores num furuto proximo!**

Fonte: Extraída do Twitter.

Pode-se notar que a publicação contém diversos desvios da língua padrão: erros involuntários (como a troca de caracteres em *maiores* e *furuto*); erros ortográficos simples (como a ausência de acento agudo em *proximo* e de cedilha em *frustrações*); abreviaturas frequentemente utilizadas na Internet (*hj* para denotar *hoje*) e erros que indicam o desconhecimento da forma padrão (como *esperiencia* e *conceguirmos*). Erros desses tipos, e muitos outros, são encontrados com frequência em todo tipo de CGU. De *reviews* de produtos até comentários em notícias, os textos publicados tendem a apresentar um alto

grau de desvio em relação à norma padrão da língua. Para identificar os diversos tipos de desvios, é necessário analisar uma grande quantia de CGU.

O trabalho de Hartmann *et al.* (2014) faz um levantamento sobre palavras fora do padrão em um *corpus* de *reviews* de produtos. Palavras fora do padrão (também denominadas OOV, do inglês *out-of-vocabulary*) são palavras não pertencentes à língua padrão – ou seja, que não constam de um léxico oficial. O *corpus* é constituído de *reviews* escritas em português brasileiro e publicadas nos *websites* Buscapé⁴, Reclame Aqui⁵ e Mercado Livre⁶. Para realizar a análise de palavras OOV, são consideradas 85.190 *reviews* obtidas no Buscapé, totalizando 4.097.905 *tokens* e 68.633 *types*. Foram consideradas como OOV palavras não presentes no dicionário de português Unitex-PB⁷ (MUNIZ *et al.*, 2005). O processo de anotação foi realizado por quatro pares de anotadores, sendo que as palavras fora do padrão foram divididas em oito categorias, sendo elas:

1. **Erros ortográficos (X)**: divididos em três subcategorias:
 - Palavras com ausência de acentos (como *nao* em vez de *não*).
 - Erros ortográficos por ausência ou substituição de letras (como *exelente* em vez de *excelente* e *imagems* em vez de *imagens*).
 - Palavras corretas não inclusas no Unitex-PB.
2. **Acrônimos (SI)**: palavras não contidas no Unitex-PB e formadas pelas letras ou sílabas iniciais de várias outras palavras, como HDTV*i* e Wi-Fi.
3. **Nomes próprios (NP)**: palavras usadas para referenciar entidades, empresas e produtos, não filtrados pelo Unitex-PB. Os nomes próprios podem ser encontrados iniciando ou não com letra maiúscula, tornando assim mais difícil identificá-los.
4. **Abreviações (AB)**: abreviações, geralmente criadas devido ao meio informal dos textos, de palavras comuns. Estabeleceu-se que a abreviação deve ter a mesma letra inicial da palavra que abrevia – por exemplo, *vc* denominando *você* e *tbm* para *também*. Diferentemente de acrônimos, abreviações são sempre formadas por uma única palavra.
5. **Internetês (IN)**: gírias frequentemente presentes na língua empregada na *web* (como *naum* em vez de *não* e *adoreiii* em vez de *adorei*). Uma característica comum no internetês é a influência da oralidade, de modo que as palavras são o reflexo exato de sua forma fonética.

⁴ <buscape.com.br>

⁵ <reclameaqui.com.br>

⁶ <mercadolivre.com.br>

⁷ <nilc.icmc.usp.br/nilc/projects/unitex-pb/web/index.html>

6. **Estrangeirismos (EN):** palavras pertencentes a outros idiomas, divididas em três subcategorias:

- Palavras advindas do inglês (como *touch*).
- Palavras advindas de outros idiomas (como espanhol e alemão).
- *Tokens* que possivelmente representam fragmentos de palavras de outros idiomas ou parte de expressões multipalavra (por exemplo *blu* como possível parte de *blu-ray*)

7. **Unidades de medida (UM):** palavras utilizadas para indicar quantidades, por exemplo *gigabytes*. Abreviações de unidades de medida não são inclusas, então *kg* como designação de quilograma pertence à categoria AB.

8. **Sem categoria (SC):** *tokens* cuja classificação não é clara ou varia conforme o contexto.

Para melhor compreender as categorias propostas, a Figura 4 apresenta textos de três *reviews* escritas por usuários, com os erros destacados e categorizados.

Figura 4 – Exemplos de *reviews* com erros anotados.

ela e [X: é] muito escura quando vc [AB: você] esta [X: está] deitado se vc [AB: você] tiver [IN: estiver] sentado ela e [X: é] boa mas deitada nao [X: não] e [X: é] muito nao [X: não]. A Samsung inova o mercado de tv's [AB: televisões] com uma grande obra de arte que se adequa [X: adequada] à qualquer ambiente. Esta tv [AB: televisão] possui excelente imagem quando ligada a uma fonte de dvd [SI: DVD] com hdmi [SI: HDMI] e na tv [AB: televisão] a cabo (Digital). O som é perfeito quando é personalizado pelo usuário. Ou seja, MENU, SOUD [ES: SOUND], EQUALIZAR E ENTER [ES].

Gostei demais [X: demais] dessa câmera, comprei outra! Além de uma excelente e reconhecida marca, essa câmera tem um design [ES] super inovador e mtu [IN: muito] atraente...uma resolução mtu [IN: muito] boa e [X: é] td [AB: tudo] o que uma boa câmera SONY tem que ter!! Até hj [AB: hoje] nunca me deixou na mão... recomendo!!!

mutu [X: muito] bom para manuziar [X: manusear] quando vc [AB: você] ta [AB: está] trabalhando [X: trabalhando] com este produto ,nao [X: não] tenho que reclamar [X: reclamar] gostei mesmo parabéns. RECOMENDO O PRODUTO, FÁCIL DE USAR, ADOREI !

Fonte: Hartmann *et al.* (2014).

É possível notar uma elevada quantia de desvios da língua padrão. Mesmo no contexto de *reviews* de produtos, que tendem a ser mais sérias que publicações em redes

sociais, a linguagem empregada ainda é altamente informal. Além dos desvios categorizados, a ausência ou mau uso de pontuação e de capitalização são características marcantes em textos gerados por usuário. A distribuição de erros por categoria pode ser vista na Figura 5.

Figura 5 – Distribuição de erros por categoria.

Fonte: [Avanço \(2015\)](#).

A categoria mais predominante é a de erros ortográficos, totalizando quase metade do montante de erros. Os nomes próprios também representam uma parcela considerável, principalmente devido ao fato de que usuários geralmente não se atentam à capitalização correta. Desse modo, é comum encontrar nomes de produtos, marcas e lojas com inicial minúscula. As demais categorias têm distribuição similar e são menos significativas. Como os erros ortográficos são os mais frequentes, é relevante analisá-los com mais profundidade.

É possível analisar erros ortográficos utilizando a métrica de distância de Damerau-Levenshtein, proposta por [Damerau \(1964\)](#). Essa medida, calculada entre duas cadeias de caracteres (*strings*), representa o mínimo de operações necessárias para transformar uma das *strings* na outra. As operações são: **inserção**: adicionar um caractere a uma das *strings*; **remoção**: remover um caractere; **substituição**: mudar um caractere e **transposição**: trocar dois caracteres adjacentes de lugar. A métrica pode ser aplicada no contexto de erros ortográficos considerando duas *strings*, sendo uma a versão correta de determinada palavra, e a outra, a versão com ortografia inválida. Segundo [Damerau \(1964\)](#), essas quatro operações correspondem a 80% de todos os erros ortográficos cometidos. Embora o trabalho original seja para língua inglesa, essa abordagem já foi repetida e corroborada em diversos idiomas. O trabalho de [Gimenes, Roman e Carvalho \(2015\)](#) segue a proposta de Damerau para analisar textos em português brasileiro. A análise é efetuada com base em dois *cópus*, uma para cálculo das estatísticas e um para validação. O primeiro (referido como C1), apresentado por [Roman et al. \(2013\)](#), é constituído de sumários de diálogos. É

composto de 1.808 textos escritos por 452 pessoas diferentes – que não tiveram acesso a corretores ortográficos durante a escrita. O segundo (C2), contém diários de viagem publicados em quatro *blogs* distintos, juntamente com comentários dos leitores, totalizando 192 *publicações* e 26.418 palavras. O objetivo é categorizar os erros ortográficos seguindo as operações propostas por Damerau e em categorias adicionais específicas ao português. Os resultados obtidos para os corpuses C1 e C2 são apresentados, respectivamente, na Tabela 2 e Tabela 3.

Tabela 2 – Número de erros ortográficos em C1 por categoria.

Categoria do Erro	Erros por Palavra				
	Um	Dois	Três	Acima de Três	Total (%)
Inserção	80	2	9	80	119 (10,45)
Remoção	135	45	11	4	195 (17,12)
Transposição	40	2	0	0	42 (3,69)
Substituição	135	10	1	0	146 (12,82)
Acento ausente	395	34	0	0	429 (37,66)
Adição de acento	17	2	0	0	19 (1,67)
Acento errado na letra correta	11	0	0	0	11 (0,96)
Acento correto na letra errada	9	0	0	0	9 (0,79)
Cedilha ausente	46	23	0	0	69 (6,06)
Substituição por espaço	1	0	0	0	1 (0,09)
Inserção de espaço	1	0	0	0	1 (0,09)
Transposição de espaço	1	0	0	0	1 (0,09)
Ausência de espaço	26	1	0	0	27 (2,37)
Outro	69	0	0	0	69 (6,06)
Total	966	140	21	12	1.139 (100)

Fonte: Adaptada de [Gimenes, Roman e Carvalho \(2015\)](#).

Os resultados indicam que erros relacionados a acentos correspondem aproximadamente à metade do total de erros. De modo geral, os resultados obtidos por [Damerau \(1964\)](#) são confirmados também no português brasileiro, visto que cerca de 85% das palavras erradas contêm um único erro. Embora o total de erros em ambos os corpuses seja próximo, C2 apresenta proporcionalmente o dobro de erros em relação a C1. Segundo os autores, essa diferença possivelmente se deve ao fato de que C1 foi gerado por estudantes universitários – ou seja, pessoas com grau de instrução elevado. Já no caso de C2, por ser composto de textos provindos de *blogs*, a linguagem empregada tende a ser mais descontraída e portanto mais propensa a erros. Nesse contexto, é relevante analisar os motivos que levam usuários a se desviarem da língua padrão ao produzir CGU.

Embora a linguagem ruidosa do CGU seja conhecida, os motivos que levam os usuários a empregá-la são pouco explorados. Conforme argumenta [Eisenstein \(2013\)](#), as

Tabela 3 – Número de erros ortográficos em C2 por categoria.

Categoria do Erro	Erros por Palavra				Total (%)
	Um	Dois	Três	Acima de Três	
Inserção	54	20	23	22	119 (9,44)
Remoção	186	65	37	16	304 (24,13)
Transposição	2	1	0	0	3 (0,24)
Substituição	32	17	16	8	73 (5,79)
Acento ausente	519	41	11	2	573 (45,48)
Adição de acento	1	0	1	0	2 (0,16)
Cedilha ausente	16	31	9	0	56 (4,44)
Outro	101	15	14	0	130 (10,32)
Total	911	190	111	48	1.260 (100)

Fonte: Adaptada de [Gimenes, Roman e Carvalho \(2015\)](#).

explicações para tal são geralmente superficiais e parciais. Em geral, quando se fala sobre o uso de linguagem fora do padrão na Internet, os principais argumentos que vêm à tona são “*as pessoas não sabem escrever corretamente*”, “*é mais rápido*”, “*tornou-se a norma*” e “*as pessoas querem fazer uso de seus próprios dialetos*” ([EISENSTEIN, 2013](#)). Apesar de aparentemente suficientes, ainda é necessário analisar as evidências que suportam esses argumentos. Para tanto, o autor propõe a avaliação das seguintes características:

- **Falta de instrução:** é o argumento que afirma que autores de CGU não têm a capacidade de seguir corretamente as normas cultas da língua padrão. No entanto, resultados de pesquisas empíricas indicam que esse não é o caso. Um estudo conduzido por [Drouin e Davis \(2009\)](#) analisou trocas de mensagens de texto entre 80 estudantes e concluiu que não há diferença de alfabetismo entre indivíduos que usam linguagem informal e os que não usam. O trabalho de [Tagliamonte e Denis \(2008\)](#) analisa um corpus, com mais de um milhão de palavras, de mensagens instantâneas trocadas entre adolescentes e conclui que a linguagem usada é conscientemente escolhida dentro de um vasto repertório linguístico – o que não seria possível se houvesse falta de instrução dos autores das mensagens. Desse modo, apesar de a linguagem do CGU ser mais informal e conter mais erros que a padrão, isso dificilmente implica uma relação com falta de instrução.
- **Limites de tamanho:** argumento frequentemente utilizado para justificar o uso de linguagem ruidosa, especialmente no Twitter – que tem limite de 140 caracteres nas publicações. Seguindo essa linha de raciocínio, é de se esperar que publicações com tamanho mais próximo ao limite contenham mais abreviações, como forma de poupar espaço. Para verificar a influência da limitação de tamanho, [Eisenstein](#)

(2013) efetuou uma análise estatística em um corpus de um milhão de *tweets*. A conclusão foi que abreviações são mais frequentes em mensagens curtas, que não estão próximas do limite de espaço. É possível que a limitação sirva como uma influência indireta, fazendo com que usuários inconscientemente abreviem palavras para poupar espaço mesmo que não necessário. No entanto, o argumento não é consistente com os resultados obtidos por meio da análise de corpus.

- **Meios de entrada de texto:** afirma que os meios de entrada de texto frequentemente utilizados por usuários – sejam teclados tradicionais ou os preditivos, presentes em *smartphones* – têm influência na linguagem utilizada. Gouws *et al.* (2011) investigou variações ortográficas no Twitter e identificou diferenças dependendo do dispositivo utilizado para entrada do texto. Mensagens publicadas de iPhones, por exemplo, possuem mais contrações que as publicadas de Blackberries, enquanto *tweets* enviados a partir do navegador *web* são mais propensos a apresentar ausência de vogais. Apesar de determinados meios de entradas inibirem certos estilos de escrita e coíbi-rem outros, a relação não é igualmente distribuída entre usuários – ou seja, diferentes usuários utilizando o mesmo dispositivo podem apresentar estilos linguísticos totalmente distintos. Sendo assim, é difícil estabelecer uma conexão clara que embase fortemente esse argumento.

Analisando sob esse ponto de vista, é possível perceber que é difícil definir claramente o que é considerado ruidoso e por que o é. Alguns dos desvios de linguagem que são considerados ruídos podem também ser vistos como escolhas pragmáticas dos usuários. *Emoticons*, por exemplo, apesar de serem uma sequência de símbolos não pertencentes à língua padrão, carregam forte significado não-verbal – usualmente sendo utilizados para expressar emoção. Outra expressão pragmática frequentemente encontrada em CGU é o alongamento proposital de palavras (como *legaaal* em vez de *legal*), que é utilizado para exprimir intensidade e reforçar o sentimento que está sendo demonstrado. Variáveis sociais também podem servir como grande influência para alterar a linguagem utilizada por usuários. É possível que o sotaque ou o dialeto do local que o usuário vive influenciem sua maneira de escrever, visto que muitas vezes a escrita informal adquire características da oralidade. O meio de convívio social também pode ser um fator de impacto. Muitas abreviações e gírias utilizadas na Internet são provenientes de expressões empregadas no cotidiano – ou seja, têm origem no círculo social do usuário.

Desvios da língua padrão, apesar de serem considerados “errados” de acordo com a normal culta, carregam muita informação. Analisar a língua do CGU como ela é, incluindo seus ruídos, permite a obtenção de conhecimento sobre o usuário, seu ambiente social, suas preferências e opiniões. Todavia, do ponto de vista prático, lidar com linguagem fora do padrão traz diversos desafios.

Desde seu surgimento, o PLN é embasado principalmente no conhecimento advindo de textos com linguagem que adere à norma culta. As pesquisas iniciais na área, que em grande parte eram abordagens baseadas em regras, não tinham como preocupação lidar com desvios linguísticos e erros ortográficos. Escrita correta foi considerada canônica e pouco se questionava a respeito. A publicação do *córpus* do Wall Street Journal, descrito em Paul e Baker (1992), transformou a perspectiva do PLN, tornando as abordagens estatísticas foco de pesquisa (PLANK, 2016). Como o *córpus* é composto por textos jornalísticos (que seguem a língua padrão), as pesquisas decorrentes baseadas nele mantiveram o foco na linguagem canônica. Para o português, os grandes *córpus* disponíveis – como os disponibilizados por meio do projeto Lacio-Web⁸ – também seguem a mesma tendência e, em grande parte, contêm apenas textos que se adequam à norma culta. No entanto, ao aplicar sistemas treinados nesse *córpus*, ou criados com base neles, a textos de CGU, os resultados são catastróficos.

Diversos trabalhos relatam perda considerável de desempenho ao testar ferramentas do estado-da-arte de PLN em textos de CGU. Alguns exemplos são: Gimpel *et al.* (2011) relatam a diminuição de 12% de acurácia ao aplicar o *tagger* de Stanford, descrito em Toutanova *et al.* (2003) e que obtém 97% de taxa de acerto no *córpus* do Wall Street Journal. Ao realizar a tarefa de etiquetagem morfossintática em *tweets*, o sistema obtém apenas 85%. Na tarefa de reconhecimento de entidades nomeadas, o sistema de Stanford, que obtém 86% de *F-measure* no conjunto de teste do CoNLL⁹, só atinge 44% no Twitter (RITTER *et al.*, 2011). Foster *et al.* (2011) relatam perda de mais de 10% de acurácia ao testar *parsers* do estado-da-arte em textos de redes sociais. Para o português brasileiro, o trabalho de Duran, Avango e Nunes (2015) relata que o *tagger* MXPOST (RATNAPARKHI, 1996), que atinge 97% de acurácia na etiquetagem morfossintática de textos jornalísticos, consegue alcançar somente 91,35% em *reviews* de produtos. Resultados como esses podem ser encontrados em diversas áreas do PLN. A linguagem ruidosa do CGU é de fato difícil de ser tratada.

Analisando superficialmente o motivo desses resultados, é possível encontrar algumas características relevantes dependendo da abordagem com que a tarefa é resolvida. Em sistemas baseados em regras, que seguem uma abordagem antiga, é comum que a resolução da tarefa seja sistematizada por meio de regras extraídas manualmente dos *córpus* analisados e criados com conhecimento linguístico. Como os *córpus* são de textos canônicos e o conhecimento linguístico é pautado na norma culta da língua, é de se esperar que as regras não sejam capazes de lidar com os ruídos do CGU. Já em abordagens estatísticas, há duas categorias a serem consideradas: abordagens totalmente estatísticas e abordagens que empregam conhecimento linguístico. No primeiro caso, o sistema depende totalmente de observações estatísticas realizadas sobre o *córpus* anotado. Consequente-

⁸ <nilc.icmc.usp.br/nilc/projects/lacio-web.htm>

⁹ <cnts.ua.ac.be/conll2003/ner/>

mente, só é capaz de aprender, e por fim lidar com, textos que seguem o mesmo padrão. Já no segundo caso, além das estatísticas do *corpus*, são incluídas *features* linguísticas ou de conhecimento específico da tarefa. Essas *features* são “dicas” que auxiliam a resolução da tarefa, ajudando a explicitar o conhecimento a ser adquirido. Alguns exemplos são: no reconhecimento de entidades nomeadas, iniciais capitalizadas são um forte indicio de nomes próprios; na análise sintática, a pontuação das sentenças é fundamental para compreender sua estrutura; na etiquetagem morfossintática, informações morfológicas, que dependem da grafia correta da palavra, podem acrescentar conhecimento valioso. No entanto, em textos de CGU, é comum que essas *features* não sejam aplicáveis. Como mencionado, usuários da *web* não se atentam à capitalização correta, ao bom uso de pontuação e à grafia exata. Sendo assim, os sistemas projetados esperando essas características não obtêm resultados satisfatórios.

Os desafios para aplicar PLN ao conteúdo gerado por usuário são muitos e árduos. Apesar disso, o conhecimento a ser extraído de tal conteúdo é valioso e pode ser utilizado em diversas aplicações promissoras. Sendo assim, é válido explorar alternativas para tratar textos ruidosos de maneira satisfatória. Uma saída é utilizar *corpus* anotados de CGU – considerando a linguagem ruidosa como canônica. No entanto, há poucos recursos desse tipo disponíveis, e criar novos é custoso e demorado. Outra alternativa é criar sistemas adaptados especialmente para textos escritos por usuários. Porém, a linguagem do CGU segue poucos padrões e muda constantemente, portanto é difícil projetar um sistema que se adapte ao CGU em geral. Uma opção mais viável é buscar um caminho intermediário: tentar aproximar a linguagem ruidosa e cheia de desvios à norma culta, tornando assim os textos de CGU mais similares aos canônicos. Desse modo, recursos de sistemas já disponíveis podem ser utilizados com pouca degradação de resultados, apenas fazendo adaptações conforme a necessidade da aplicação. Essa abordagem é denominada normalização textual.

NORMALIZAÇÃO TEXTUAL

Normalização textual é o processo de transformar palavras fora do padrão (NSWs, do inglês *non-standard words*) em palavras consideradas canônicas. Em geral, esse padrão é o léxico reconhecido pela forma culta da língua. No entanto, ele pode ser adaptado conforme as necessidades de aplicações específicas. Tipicamente, NSWs são formas ruidosas de palavras corretas, ou seja, contêm ruídos que as tornam irreconhecíveis pelo padrão estabelecido. Esses ruídos são encontrados em diversas formas – como erros ortográficos, siglas, capitalização inválida e mau uso de pontuação. Textos de CGU são fontes de grande quantia de ruídos, devido principalmente à sua natureza informal. Desse modo, é natural considerar a normalização para melhorar a qualidade desses textos.

Métodos que buscam adaptar textos e palavras são propostos há tempos. O trabalho de [Damerau \(1964\)](#), por exemplo, já analisava técnicas para correção ortográfica na década de 60. No entanto, a normalização textual como uma tarefa de PLN por si só surgiu mais recentemente. Dois trabalhos podem ser considerados como os pioneiros que ajudaram a estabelecer a área. O trabalho de [Brill e Moore \(2000\)](#), embora tratasse de correção ortográfica, propôs modelos robustos para normalização de erros que inspiraram o desenvolvimento posterior de várias técnicas, difundindo assim a tarefa. Já o trabalho de [Sproat et al. \(2001\)](#) foi o primeiro a formalizar a normalização textual como um processo completo – dando início a pesquisas na área na forma em que ela é reconhecida atualmente. A proposta principal de [Sproat et al. \(2001\)](#) foi criar técnicas de normalização para sistemas de síntese de voz. Nesse contexto, diversos ruídos precisavam ser tratados, como datas, número de telefones e abreviações. Esses tipos de ruídos foram considerados pois precisam ser processados de forma específica pela síntese de voz, uma vez que precisam ser falados. As principais contribuições do trabalho foram técnicas de normalização, corpus anotados e categorização de erros. Embora projetado para uma tarefa específica, a metodologia empregada foi fundamental para dar início a pesquisas tratando exclusivamente de normalização textual.

Apesar de alguns trabalhos estabelecerem a normalização textual como área de pesquisa, ela seguiu majoritariamente sendo considerada como uma etapa de pré-processamento. Consequentemente, seu desenvolvimento foi em grande parte atrelado a aplicações específicas. Desse modo, há pouco consenso na área sobre o que considerar como ruídos, o que normalizar e quais técnicas são mais eficientes. Como a tarefa é fortemente vinculada a aplicações, os métodos de avaliação, incluindo *corpus*, também o são. Esse fato torna difícil a comparação e replicação das técnicas, visto que cada trabalho trata o processo de normalização de uma maneira. Todavia, alguns conceitos são fundamentais para compreender métodos que visam a normalização de textos. Este capítulo tem como objetivo apresentar a fundamentação que constitui a base da área de normalização textual.

3.1 Fundamentos

A normalização textual trata primariamente de transformar palavras ruidosas em palavras reconhecidas por um padrão. Nesse contexto, é fundamental encontrar maneiras efetivas para representar palavras e como elas se relacionam, ou seja, de verificar se são similares ou não. Essa seção apresenta as principais abordagens de representação de palavras e de similaridades entre palavras utilizadas em técnicas de normalização.

3.1.1 Representação de palavras

Embora existam sistemas de escrita bem estabelecidos para língua natural, eles são projetados para compreensão humana e não dos computadores. Consequentemente, se faz necessário analisar meios de representar a língua – e mais especificamente palavras – de modo a torná-la compreensível pelas máquinas. Uma das formas de representação mais difundidas é por meio de códigos binários que codificam letras em cadeias de *bits*. Um exemplo dessa prática é a tabela UNICODE, que provê uma codificação completa dos caracteres utilizados na escrita. Codificações desse tipo são utilizadas majoritariamente para entrada e saída de dados – ou seja, para ler e mostrar texto na tela do computador. No entanto, codificações binárias não mantêm relação com a língua humana, fazendo com que tenham pouca utilidade para serem aplicadas como entrada em tarefas de PLN.

Aplicações de PLN geralmente recebem textos como entrada, portanto palavras podem ser consideradas como unidade básica de processamento. Sendo assim, é importante que elas sejam representadas de maneira que carreguem informações relevantes. Motivadas por algoritmos de aprendizado de máquina, palavras em PLN tendem a ser representadas por vetores numéricos. Uma das primeiras abordagens empregadas para tanto foi a de representação por meio de vetores *one-hot* – que representam cada palavra por um código binário. Esses vetores têm como número de dimensões a quantidade total de palavras que são representadas, pois cada uma possui identificação única. Consequen-

temente, os vetores gerados são altamente esparsos, sendo em maior parte preenchidos por zeros. Essa característica os torna ineficientes visto que são pouco representativos. Uma alternativa um pouco mais sofisticada aos vetores *one-hot* são as *bag-of-words*. Nesse tipo de representação, palavras iguais são agrupadas e suas posições correspondentes no vetor são seu número total de ocorrências – ou seja, a quantia de vezes que aparecem no texto processado. Ambas abordagens possuem uma grande deficiência em comum: geram vetores que não carregam conhecimento algum, sendo oriundos de simples contagem de palavras.

Com o desenvolvimento de técnicas mais sofisticadas de PLN e aprendizado de máquina, foi natural que surgissem representações mais efetivas de palavras. O trabalho de [Bengio et al. \(2003\)](#) foi um dos pioneiros em empregar redes neurais para aprender automaticamente representações vetoriais. Esse tipo de representação, formalmente denominado *representações distribuídas*, são atualmente mais conhecidas por *word embeddings* ou simplesmente *embeddings*. Seu uso é recente e vem obtendo sucesso em diversas tarefas de PLN, sendo utilizadas principalmente como entrada de arquiteturas neurais. De modo geral, *embeddings* são vetores reais distribuídos em um espaço multidimensional induzidos por meio de aprendizado não-supervisionado ([TURIAN; RATINOV; BENGIO, 2010](#)). Cada dimensão do vetor de uma palavra representa uma característica, de modo a capturar distribuidamente uma propriedade semântica, sintática ou morfológica dessa palavra ([COLLOBERT et al., 2011](#)).

O trabalho de [Collobert e Weston \(2008\)](#) foi um dos primeiros a introduzir o conceito de *word embedding* da maneira como é atualmente empregado. A abordagem propõe um modelo de língua baseado em uma rede neural que tenta prever a próxima palavra do contexto, dadas as anteriores. As representações das palavras são aprendidas por meio do ajuste de pesos da rede utilizando o algoritmo de *backpropagation* ([GOODFELLOW; COURVILLE, 2016](#)). Palavras que ocorrem com frequência em um mesmo contexto obtêm vetores similares, sendo que é o processo de treinamento da rede que aprende essa noção. Para efetuar o treinamento, é necessário apenas um *corpus* – que não precisa de anotações visto que somente a janela de contexto de cada palavra é utilizada.

O modelo de *embeddings* Word2Vec, proposto por [Mikolov et al. \(2013b\)](#), popularizou em grande escala esse tipo de representação. A técnica utilizada para geração do modelo segue o mesmo princípio do modelo de língua neural de [Collobert e Weston \(2008\)](#). Essa técnica foi dividida em dois modelos: *Continuous bag-of-words* (CBOW), ilustrado na Figura [Figura 6](#), e Skip-gram, ilustrado na Figura [Figura 6](#). O CBOW tem como objetivo prever a palavra do meio dada uma janela de contexto. Já o Skip-gram prediz as palavras da janela de contexto dada a palavra do meio como entrada. A representação vetorial é aprendida como consequência do treinamento da rede.

Um ponto negativo do Word2Vec é que a ordem das palavras dentro da janela de

Figura 6 – Modelos do *word embeddings* Word2Vec.

Fonte: Adaptada de Mikolov *et al.* (2013b).

contexto é desconsiderada. Desse modo, informações sintáticas não são representadas nos vetores. Para contornar esse problema, o trabalho de Ling *et al.* (2015) propôs uma versão modificada do Word2Vec, denominada Wang2Vec. A modificação representa a ordem das palavras por meio da concatenação das entradas em vez da soma, no caso do modelo CBOW, e como uma ponderação na saída do Skip-gram. A arquitetura do modelo é apresentada na Figura 7. Embora simples, a versão modificada é capaz de gerar vetores que carregam mais conhecimento sintático que o Word2Vec.

Figura 7 – Modelos de *word embeddings* Wang2Vec.

Fonte: Adaptada de Ling *et al.* (2015).

Como mencionado, o treinamento das *embeddings* depende somente de um corpus não-anotado. Nos modelos tradicionais, são geradas representações vetoriais apenas para palavras encontradas no corpus, visto que janelas de contexto em nível de palavra são essenciais para o treinamento. Essa característica pode ser problemática para aplicações cujo corpus utilizado não seja o mesmo que serviu de base para o treinamento das *embeddings*, pois pode haver palavras que não possuem vetores associados. O modelo proposto por Bojanowski *et al.* (2016), Joulin *et al.* (2016), denominado FastText, utiliza janelas de contexto em nível de caractere – utilizando n-gramas de caracteres de uma dada palavra em vez da palavra em si. Desse modo, é possível gerar vetores até mesmo palavras que não constam no vocabulário, pois apenas os caracteres que a compõe bastam para a geração. O FastText é baseado em uma adaptação do modelo Skip-gram do Word2Vec, sendo que

a saída da rede neural é calculada utilizando os vetores dos n-gramas de caracteres – que em seguida são somados aos de contexto para representar uma palavra w_t . A Figura 8 ilustra esse processo.

Figura 8 – Modelo de *word embeddings* FastText.

Fonte: Adaptada de Bojanowski *et al.* (2016).

Os vetores obtidos por meio dos modelos de *embeddings* são capazes de representar similaridades entre as palavras, sendo que palavras próximas no espaço vetorial (de acordo com alguma métrica de distância) tendem a ter significados similares. Um exemplo que ilustra essa capacidade de representação pode ser visto na Figura 9, que apresenta o espaço vetorial das *embeddings* reduzido para duas dimensões.

Figura 9 – Similaridades entre vetores de *word embeddings*.

Fonte: Cholleton (2016).

É possível notar três grupos que reúnem palavras com conceitos similares (música, computadores e comida). Em geral, a métrica de distância utilizada para calcular a similaridade entre *embeddings* é a *similaridade do cosseno*, que calcula o cosseno do ângulo

formado entre dois vetores – quanto maior seu valor, maior a similaridade. Esse tipo de agrupamento conceitual pode ser encontrado nas mais diversas formas, inclusive agrupando vetores que representam nome de países e que representam palavras de mesma classe gramatical. Outra característica interessante é a possibilidade de efetuar operações aritméticas entre os vetores. Um exemplo clássico do Word2Vec é que a operação $(\text{vetor}(\text{Rei}) - \text{vetor}(\text{Homem})) + \text{vetor}(\text{Mulher})$ resulta em um vetor extremamente próximo ao que representa a palavra *rainha* – mostrando que *word embeddings* são capazes de representar analogias entre conceitos.

3.1.2 Representação de similaridade entre palavras

Considerando as representações de palavras apresentadas, a similaridade entre elas pode ser medida com base em duas perspectivas. A primeira considera a palavra segundo sua representação em língua natural: ou seja, como uma sequência de letras (caracteres). Seguindo essa representação, medidas de similaridade de *string* e de similaridade fonética podem ser empregadas. A segunda perspectiva considera a palavra como um vetor numérico, ou *word embedding*. Nesse caso, medidas de distância em espaços vetoriais podem ser aplicadas. Nesta seção, serão apresentadas as principais medidas de similaridade que frequentemente são utilizadas em técnicas de normalização textual.

Ao se representar palavras como sequências de caracteres, é necessário utilizar medidas de similaridade de *strings* para compará-las. Uma das métricas mais empregadas para tal fim é a distância de edição, que consiste na contagem do número mínimos de edições para transformar uma *string* em outra. Tradicionalmente, são consideradas como edições as operações de inserção, substituição e remoção de letras. Uma modificação proposta por Damerau (1964), conhecida como distância de Damerau-Levenshtein, inclui a transposição (troca de letras adjacentes) como uma operação de edição. Essa medida foi proposta originalmente no contexto de correção ortográfica, sendo bem adequada para essa tarefa. A distância de edição entre *tbm* e *tambor*, por exemplo, é 4 (três inserções e uma substituição). Já a distância entre *tbm* e *também* é apenas 3 (inserção de *a*, de *m* e de *é*), mostrando que entre as duas essa é a correção mais provável. É possível inserir conhecimento específico de um idioma no cálculo dessa medida. Um exemplo é considerar adição de acento em uma letra como uma operação que tem custo menor que uma substituição convencional – facilitando assim a correção de erros causadas pela ausência de acentuação.

Outra medida amplamente utilizada é a *Longest Common Subsequence* (LCS), que representa a maior subsequência de caracteres em comum entre duas *strings*. Subsequências, diferente de *substrings*, não necessitam ser contíguas, os caracteres só precisam estar na mesma ordem em ambas *strings*. Entre as palavras *humano* e *chimpanzé*, por exemplo, a maior subsequência é *hman*. O cálculo dessa medida é um problema tradicional de

computação, sendo usualmente feito por meio de algoritmos de programação dinâmica (ULLMAN; AHO; HIRSCHBERG, 1976). Uma característica interessante da LCS é sua capacidade de representar o quanto uma palavra está contida em outra, tornando-a especialmente útil para lidar com abreviações. A palavra *enquanto*, por exemplo, é comumente abreviada como *eqt* em textos da *web*. A distância de edição entre elas é alta (5), indicando baixa probabilidade de ser a correção correta. No entanto, a LCS é a própria *string eqt*, portanto com o uso dessa medida é possível corrigir corretamente. A *Longest Common Subsequence Ratio* (LCSR) é uma ponderação da LCS, obtida dividindo-se o tamanho da maior subsequência pelo número de caracteres da maior palavra – resultando em um valor normalizado de 0 a 1, que a torna mais fácil de utilizar em conjunto com outras medidas.

O coeficiente de Dice, também conhecido por coeficiente de Sørensen-Dice, foi proposto originalmente por Sørensen (1948) como uma medida estatística para comparar duas amostras aleatórias. Sua adaptação para lidar com *strings* é calculada com base no número de bigramas de caracteres compartilhados entre duas palavras. Considerando X e Y como *strings* representadas por um conjunto de caracteres, o cálculo exato do coeficiente é computado como:

$$\mathcal{D} = \frac{2|X \cap Y|}{|X| + |Y|} \quad (3.1)$$

Seguindo a Equação 3.1, $\mathcal{D}$ indica o coeficiente de Dice e a operação de interseção é definida como o número de bigramas em comum entre as palavras. $|X|$ indica o tamanho em número de caracteres da *string*. Entre as palavras *experimento* e *expmnto*, por exemplo, o coeficiente é $\frac{2 \cdot (4)}{19} \approx 0,421$, pois as palavras têm quatro bigramas em comum (*ex*, *xp*, *nt* e *to*). Essa medida pode ser especialmente útil na normalização de ruídos que mantêm sílabas em comum com sua versão correta.

Outra abordagem para realizar comparação entre palavras é por meio da similaridade fonética. O algoritmo Soundex, proposto originalmente por Rusell (1918), tem como objetivo codificar palavras de acordo com os fonemas de sua pronúncia. Essa codificação faz com que palavras homófonas sejam representadas da mesma forma. Essa característica facilita a correção de erros motivados pela fala. A palavra *excesso*, por exemplo, é frequentemente escrita com a grafia incorreta *ecesso*, seguindo a maneira que é falada. Enquanto a distância de edição entre elas é 1, assim como um grande número de outras palavras que só tem uma letra diferente, a distância fonética é 0, pois são homófonas. Desse modo, o Soundex permite a correção de forma efetiva. Embora o algoritmo tenha sido originalmente projetado para o inglês, há adaptações disponíveis para outros idiomas. O trabalho de Duran, Avanço e Nunes (2015) propõe um corretor ortográfico baseado em fonética, que faz uso de uma versão modificada do Soundex com regras específicas para o português brasileiro.

Ao utilizar representações vetoriais de palavras, outras medidas de similaridade são necessárias. Em geral, *word embeddings* são distribuídas em um espaço de alta dimensionalidade. Como os vetores de *embeddings* são numéricos e contínuos, é possível compará-los utilizando métricas algébricas de distância. A distância euclidiana é uma das mais conhecidas, sendo calculada de acordo com a [Equação 3.2](#), sendo n o número de dimensões das *embeddings* e u e v dois vetores cuja distância está sendo calculada.

$$\mathcal{D} = \sqrt{\sum_{i=1}^n (u_i - v_i)^2} \quad (3.2)$$

Embora seja famosa, a distância euclidiana não é usualmente utilizada no contexto de aprendizado de máquina por não lidar satisfatoriamente com espaços de alta dimensionalidade ([MITCHELL, 1997](#)). A métrica mais comum para calcular a similaridade de *embeddings* é a distância do cosseno, apresentada na [Equação 3.3](#), sendo u e v vetores de n dimensões.

$$\mathcal{D} = \frac{\sum_{i=1}^n u_i \times v_i}{\sqrt{\sum_{i=1}^n (u_i)^2 \times \sum_{i=1}^n (v_i)^2}} \quad (3.3)$$

Essa medida calcula o cosseno do ângulo formado pelos vetores. Seu valor varia de 0 a 1 – com 0 indicando que são opostos e portanto não possuem similaridade e 1 que são idênticos.

Cada uma das medidas descritas tem suas particularidades e servem a um propósito. Combiná-las de modo a mensurar similaridade de maneira mais complexa é possível e pode apresentar bons resultados. Embora representações vetoriais sejam melhores para algoritmos de aprendizado de máquina, as medidas de similaridade de *strings* são úteis para normalização textual. Desse modo, é vantajoso explorar abordagens híbridas que sejam capazes de combinar o melhor de cada uma das representações apresentadas.

3.2 Abordagens de normalização

Uma vez representadas as palavras e as relações entre elas, é possível efetuar o processo de normalização. Ainda é necessário, entretanto, modelar detalhadamente como o processo ocorrerá. Não existe consenso na área sobre como o problema deve ser modelado, sendo que há várias propostas diferentes. No entanto, algumas abordagens são encontradas com mais frequência na literatura. É importante ressaltar que abordagens que modelam o problema da mesma maneira não necessariamente fazem uso das mesmas técnicas. Assim, o termo *abordagem* se refere ao modo como a tarefa de normalização é

modelada computacionalmente. Nesta seção, serão apresentadas as principais abordagens encontradas na literatura.

3.2.1 Baseadas em Léxico

É a abordagem mais simples de normalização. Consiste basicamente em utilizar léxicos (ou listas de palavras) anotados, contendo palavras ruidosas como entrada associadas às suas potenciais correções. É possível que cada ruído esteja associado a apenas uma correção, tornando o processo de normalização determinístico – ou seja, o ruído é sempre substituído pela mesma palavra. Caso existam diferentes candidatos à substituição, é necessário empregar alguma forma de desambiguação. Para tanto, medidas de similaridade entre palavras podem ser aplicadas.

A elaboração dos léxicos utilizados na normalização é geralmente feita manualmente. Portanto, é necessário compilar listas de palavras com ruídos e posteriormente corrigi-las. Esse processo deve ser feito com base em amostras de textos ruidosos. Um problema dessa abordagem é que somente são identificadas palavras que estão contidas nas amostras escolhidas para análise. Sendo assim, é comum que os léxicos elaborados fiquem restritos a um domínio específico. Outro ponto negativo é que podem existir diversas formas do mesmo ruído, muitas vezes contendo apenas pequenas variações, tornando difícil encontrar todas as variações em um conjunto restrito de textos. A criação manual de listas de palavras não emprega métodos automáticos de aprendizado, limitando demais sua aplicabilidade.

Embora restrita e não-escalável, a abordagem baseada em léxico pode ser efetiva em domínios bem delimitados. Se apenas textos de determinado assunto oriundos da mesma fonte, por exemplo, são processados pela aplicação, é possível que um léxico construído manualmente seja capaz de normalizar extensivamente os ruídos encontrados. Todavia, lidar com normalização de forma genérica não é facilmente realizável com essa abordagem. Consequentemente, é válido explorar abordagens que façam uso de algoritmos capazes de aprender correções com base em observações feitas sobre *córpus* anotados.

3.2.2 Baseadas em Noisy-channel

O modelo *noisy-channel* é uma metáfora de representação que pode ser aplicada em diversas tarefas de PLN. Em sua concepção original, proposta por [Shannon \(1956\)](#), o modelo tinha como objetivo representar a transmissão de dados. O processo era modelado com base na seguinte sequência de passos:

1. Os dados são emitidos a partir de um canal transmissor destinados a um canal receptor.

2. Ao decorrer da transmissão, os dados passam por um canal ruidoso que os corrompe.
3. O canal de destino recebe os dados corrompidos e não é capaz de decodificá-los corretamente.

O objetivo do *noisy-channel* é modelar cada um desses canais (emissor, receptor e canal ruidoso) de modo a conseguir fazer o processo contrário – obter os dados originais a partir dos corrompidos. Uma ligação telefônica é um bom exemplo para ilustrar o processo. Uma pessoa (emissor) faz uma ligação para outra (receptor). No entanto, as linhas de transmissão estão instáveis, fazendo com que a qualidade da chamada seja baixa (canal ruidoso), com muitos ruídos. Consequentemente, a pessoa não consegue entender corretamente o que está sendo dito – pois recebe dados corrompidos. O objetivo do *noisy-channel* é conseguir recuperar o que está sendo falado originalmente.

De maneira mais formal, o modelo *noisy-channel* visa a encontrar a sequência mais provável de palavras dada a observação de uma mensagem ruidosa. Por exemplo: uma sentença c de tamanho n é corrompida por um canal ruidoso gerando uma mensagem w de tamanho m . A sentença original pode ser recuperada por meio da distribuição de probabilidade *a posteriori* do canal receptor dado o canal emissor $P(w|c)$ e uma distribuição de probabilidade *a priori* do canal emissor $P(c)$. Sendo assim, o objetivo do *noisy-channel* é encontrar $\arg \max P(w|c)$. Aplicando a regra de Bayes e desconsiderando o denominador constante, o problema pode ser reescrito como $\arg \max P(c|w) \cdot P(w)$. A probabilidade $P(w)$ pode ser estimada por meio de observações no *corpus*. Já a probabilidade $P(c|w) \cdot P(w)$ não pode ser encontrada diretamente. No entanto, geralmente há um alinhamento que pode ser modelado entre a sentença original e a corrompida – permitindo assim estimar a probabilidade.

A modelagem do *noisy-channel* na prática se resume a encontrar esses alinhamentos. Para tanto, é necessário representar de alguma maneira o processo que transforma uma sequência normal em uma sequência ruidosa. Esse processo, também chamado de história gerativa, é geralmente modelado com engenharia de *features* – que é escolha de um conjunto de características que capaz de modelar a corrupção causada pelo canal ruidoso. Na normalização textual, a história gerativa deve representar o processo de correção de uma palavra ruidosa. Uma vez criada a história gerativa, o modelo pode estimar as probabilidades com base em observações sobre um *corpus*.

3.2.3 Baseadas em Redes Neurais

Abordagens baseadas em redes neurais vêm sendo utilizadas com sucesso em diversas tarefas de PLN, como análise de sentimentos (TAI; SOCHER; MANNING, 2016), simplificação textual (WANG *et al.*, 2016), análise sintática (FONSECA; ALUISIO, 2015)

e segmentação de sentenças (TREVISIO; SHULBY; ALUÍSIO, 2017). A tarefa de normalização segue a mesma tendência, com um grande número de trabalhos recentes utilizando redes neurais (MIN; MOTT, 2015; LEEMAN-MUNK; LESTER; COX, 2015; IKEDA; SHINDO; MATSUMOTO, 2016). Um atrativo para o uso dessas arquiteturas é a possibilidade de utilizar *word embeddings* como entrada, viabilizando assim o aprendizado sem engenharia de *features* manuais. Desse modo, é relevante analisar os fundamentos que formam a base dessa abordagem. Esta seção trata sobre os principais conceitos relacionados a redes neurais. Serão apresentadas as arquiteturas mais relevantes para a tarefa de normalização: *multilayer perceptrons* e redes recorrentes.

Multilayer Perceptron

Com uma arquitetura amplamente difundida, as *multilayer perceptrons* (MLPs) são possivelmente as rede neurais multi-camadas mais simples. Consistem de uma sequência de camadas compostas por unidades conhecidas como neurônios, os quais são responsáveis por calcular funções de ativação sobre os dados de entrada. A Figura 10 apresenta uma arquitetura simples de uma MLP com apenas uma camada oculta.

Figura 10 – Arquitetura de uma MLP.

Fonte: Cazala (2016).

A primeira camada de uma MLP é denominada *camada de entrada* e a última, *camada de saída*, sendo responsável por retornar o valor final computado pela rede. As camadas intermediárias são denominadas *camadas ocultas*. O funcionamento do fluxo de uma rede neural começa com a entrada da matriz de dados $X \in \mathbb{R}^{m \times n}$, em que m é o número de instâncias de treinamento e n é a quantidade de *features* associada a cada instância. As entradas são passadas para a próxima camada, na qual cada neurônio a_i visa a aprender uma informação associada a cada entrada anterior $\mathbf{x} \in \mathbf{X}$, de modo a

produzir uma ativação de acordo com a [Equação 3.4](#).

$$\mathbf{a} = f(\mathbf{z}) = f(\boldsymbol{\theta}^\top \mathbf{x}) \quad (3.4)$$

Sendo f a função de ativação, que comumente é uma função não-linear aplicada elemento a elemento sobre os dados. $\boldsymbol{\theta}$ representa os pesos associados entre as duas camadas. Exemplos comuns de funções de ativação são a função sigmoide ([Equação 3.5](#)), tangente hiperbólica ([Equação 3.6](#)), e *Rectified Linear Unit* (ReLU), ou unidade linear retificada ([Equação 3.7](#)).

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (3.5)$$

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (3.6)$$

$$\text{relu}(z) = \max(0, z) \quad (3.7)$$

O número de unidades de ativação na última camada da rede $y = h_\theta(X) = \mathbf{a}^{(L)}$ define a quantidade de etiquetas que o modelo classifica. Em um problema de classificação binária $y \in \{\mathcal{Y}_1, \mathcal{Y}_2\}$, a camada de saída possui apenas um neurônio com a função de ativação sigmoide. Em um problema multi-classe $y \in \{\mathcal{Y}_1, \dots, \mathcal{Y}_K\}$ é possível generalizar ao fazer com que camada de saída possua exatamente K unidades de ativação. A probabilidade de uma instância $\mathbf{x}$ ser da classe $\mathcal{Y}_k$ pode ser calculada com uma função *softmax*, descrita na [Equação 3.8](#).

$$P(\mathcal{Y}_k | \mathbf{x}) = h_\theta(\mathbf{x})_k = \frac{\exp(y_k)}{\sum_{j=1}^K \exp(y_j)} \quad (3.8)$$

Para encontrar o conjunto de pesos $\boldsymbol{\theta}$ ideais, é necessário definir uma função de custo que mensura o quão distante uma predição $\hat{y} = h_\theta(\mathbf{x})$ da rede está do valor real. Com base nisso, é possível aplicar métodos de gradiente descendente para minimizar a função de custo em relação aos pesos. Uma função frequentemente utilizada para esse fim é a *entropia cruzada* descrita na [Equação 3.9](#).

$$\mathcal{L}(\boldsymbol{\theta}) = - \sum_{k=1}^K y_k \log(\hat{y}_k) \quad (3.9)$$

Para minimizar a função de custo, o algoritmo *backpropagation* ([WERBOS, 1974](#)) é amplamente utilizado, já que ele reusa partes da computação dos gradientes de maneira eficaz. O algoritmo segue a direção inversa da usada para obter a saída $h(x)$ de modo a computar os gradientes de atualização dos pesos $\boldsymbol{\theta}$ em relação a função de custo $\boldsymbol{\theta} \mapsto \mathcal{L}$.

Esse comportamento é conhecido como gradiente descendente. A regra de atualização dos pesos num instante t é dada pela [Equação 3.10](#), sendo η a taxa de aprendizado, um valor geralmente pequeno que tem como objetivo definir o tamanho dos “passos” dados no gradiente, consequentemente controlando a velocidade de convergência da rede.

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla \mathcal{L}(\theta_t) \quad (3.10)$$

Uma técnica recente que vem sendo utilizada para evitar o *overfitting* no treinamento é o *dropout*, proposta por [Srivastava et al. \(2014\)](#). De modo geral, a técnica aleatoriamente “desliga” unidades neurais da rede durante o treinamento de modo a evitar que neurônios topologicamente próximos (ou conectados) aprendam de forma correlacionada – ou seja, dependendo um do outro. O uso do *dropout* acarreta, principalmente para redes neurais grandes, em diminuição considerável do *overfitting*.

Redes Neurais Recorrentes

Redes Neurais Recorrentes (RNNs, do inglês *recurrent neural networks*) são similares a MLPs em sua arquitetura, com a adição da possibilidade de conexões recorrentes entre camadas. A [Figura 11](#) apresenta um neurônio recorrente. Vários deles podem ser combinados para formar a arquitetura da rede, de modo análogo às MLPs.

Figura 11 – Neurônio de uma rede neural recorrente.

Fonte: [Olah \(2015\)](#).

Recorrências dão à rede a capacidade de lidar com sequências de dados, visto que informações já computadas podem ser consideradas na predição final, capturando assim relações temporais. Além disso, a sequência de dados pode ter um tamanho variável, permitindo que sentenças possam ser modeladas diretamente como sequências de palavras. De modo a adaptar o processo de aprendizado para aceitar sequências de tamanhos variáveis, os parâmetros da rede são compartilhados através do “tempo” – que avança a cada passo de execução da rede. Esse conceito, denominado *time delayed neural networks*, foi apresentado originalmente por [Lang, Waibel e Hinton \(1990\)](#).

Nas RNNs, cada instância de entrada de dados é considerada como um elemento temporal. Desse modo, um instante de tempo é o conjunto de passos de execução necessários para processar toda a sequência de entrada. A camada de saída de uma rede

recorrente também é temporal, de modo que a etiqueta mais provável para cada elemento da sequência pode ser obtida por meio de uma operação de *argmax* aplicada elementarmente na Equação 3.8. A fim de minimizar a função de custo, os pesos da RNN podem ser ajustados usando o algoritmo *Backpropagation Through Time* (BTT) (WERBOS, 1990), que é uma adaptação do *backpropagation* tradicional. O BTT realiza o ajuste dos pesos ao longo do tempo somando-se os gradientes da função de custo em cada instante t .

Em determinadas aplicações, é relevante considerar informações de um contexto futuro para realizar uma predição. Na tarefa de desambiguação lexical de sentidos, por exemplo, saber qual a palavra conseguinte a que se deseja desambiguar pode ser útil na determinação de seu sentido – portanto é interessante utilizar uma *feature* de informação do contexto à direita (futuro). Entretanto, RNNs convencionais processam a sequência de dados na ordem temporal, da esquerda para direita. O trabalho de Schuster e Paliwal (1997) propõe uma solução para esse problema. A abordagem proposta consiste em passar uma sequência de dados para uma RNN paralela que processa em ordem invertida (da direita para esquerda). Essa RNN é independente da original – a qual processa a entrada na ordem temporal. Para combinar os dois contextos, a saída de cada neurônio é combinada posteriormente. Arquiteturas que fazem uso dessa abordagem são denominadas Redes Neurais Recorrentes Bidirecionais (BRNN, do inglês *bidirectional recurrent neural networks*).

Uma maneira de visualizar as RNNs mais intuitivamente é desenrolando as recorrências. A Figura 12 apresenta essa forma de visualização. Nesse formato, a RNN pode ser vista como uma MLP com várias camadas ocultas – sendo que cada uma é na verdade um instante de tempo, representando o processamento de cada elemento da sequência.

Figura 12 – Unidade recorrente desenrolada.

Fonte: Olah (2015).

Apesar de possibilitar o ajuste dos pesos da rede e consequentemente o treinamento, o BTT apresenta dois problemas de aprendizado (GOODFELLOW; COURVILLE, 2016): (i) desvanecimento do gradiente (*vanishing gradient*), que ocorre quando o gradiente se torna muito pequeno ao decorrer dos passos de tempo e acaba perdendo a relevância em relação a unidades anteriores, fazendo com que os pesos não consigam ser ajustados satisfatoriamente e (ii) explosão do gradiente (*exploding gradient*), que é em linhas gerais

o oposto do desvanecimento, ocorrendo quando o gradiente atinge valores extremamente altos e impossibilita a convergência da rede. Uma possível solução para esses problemas é a utilização de células de memória especiais como unidades de ativação, como os neurônios do tipo *Long-Short Term Memory* (LSTM) (HOCHREITER; SCHMIDHUBER, 1997).

LSTM é uma unidade neural recorrente, sendo que a combinação de várias células forma uma rede de LSTMs. Nesse trabalho, o termo LSTM será utilizado para se referir genericamente à arquitetura neural composta por neurônios de LSTM – podendo ser formada por vários ou apenas um. Muitas variantes de LSTM já foram propostas na literatura, no entanto a arquitetura básica proposta por Greff *et al.* (2016) será utilizada na apresentação dos conceitos pertinentes. A Figura 13 ilustra as conexões de uma unidade de LSTM.

Figura 13 – Unidade de LSTM básica.

Fonte: Greff *et al.* (2016).

Conforme ilustrado na Figura 13, a arquitetura básica consiste de um conjunto de conexões recorrentes. As atualizações de acordo com as relações dos blocos da figura para um instante t são definidas como:

$$\begin{aligned}
 i_t &= \sigma(\theta_{ai} \cdot \mathbf{a}_t + \theta_{ui} \cdot \mathbf{u}_{t-1} + \mathbf{b}_i) \\
 f_t &= \sigma(\theta_{af} \cdot \mathbf{a}_t + \theta_{uf} \cdot \mathbf{u}_{t-1} + \mathbf{b}_f) \\
 o_t &= \sigma(\theta_{ao} \cdot \mathbf{a}_t + \theta_{uo} \cdot \mathbf{u}_{t-1} + \mathbf{b}_o) \\
 g_t &= \tanh(\theta_{ac} \cdot \mathbf{a}_t + \theta_{uc} \cdot \mathbf{u}_{t-1} + \mathbf{b}_c) \\
 c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
 u_t &= o_t \odot \tanh(c_t)
 \end{aligned}$$

Sendo $\mathbf{u}_t \in \mathbb{R}^\varphi$ uma unidade recorrente oculta, $i_t \in \mathbb{R}^\varphi$ o *gate* de entrada, $f_t \in \mathbb{R}^\varphi$ o *gate* de esquecimento, $o_t \in \mathbb{R}^\varphi$ o *gate* de saída, $g_t \in \mathbb{R}^\varphi$ é o *gate* de modulação da

entrada, e $c_t \in \mathbb{R}^\varphi$ a célula de memória, que consiste da soma de células de memórias pela multiplicadas elementarmente ($\odot$) por f_t com estados ocultos anteriores modulados pelo *gate* de entrada i_t . As matrizes θ_{ij} são pesos das conexões entre os *gates* i e j e $\mathbf{b}_i$ são vetores de viés para o *gate* i .

Ao contrário de uma RNN básica, a saída de uma unidade LSTM não depende apenas de estados anteriores, sendo que também são utilizadas informações codificadas internamente em sua célula de memória por meio dos *gates*. Os *gates* (ou portões) são definidos com funções sigmoide (σ) e podem ser vistos como reguladores que deixam passar $\sigma(x)$ de informação adiante. Já os *gates* que usam funções *tanh*, são análogos a funções de recompensa.

Com a motivação de simplificar o funcionamento de uma célula de LSTM, o trabalho de [Cho et al. \(2014\)](#) propôs uma unidade chamada *Gated Recurrent Unit* (GRU), na qual um único *gate* $\tilde{u}_t$ é responsável por controlar simultaneamente o fator de esquecimento f_t e a decisão de atualizar a célula interna c_t :

$$\begin{aligned} z_t &= \sigma(\theta_{az} \cdot \mathbf{a}_t + \theta_{uz} \cdot \mathbf{u}_{t-1} + \mathbf{b}_z) \\ r_t &= \sigma(\theta_{ar} \cdot \mathbf{a}_t + \theta_{ur} \cdot \mathbf{u}_{t-1} + \mathbf{b}_r) \\ \tilde{u}_t &= \tanh(\theta_{au} \cdot \mathbf{a}_t + \theta_{uu} \cdot (r_t \odot \mathbf{u}_{t-1}) + \mathbf{b}_u) \\ u_t &= (1 - z_t) \odot \mathbf{u}_{t-1} + z_t \odot \tilde{u}_t \end{aligned}$$

Sendo θ_{ij} pesos das conexões entre os *gates* i e j , e $\mathbf{b}_i$ vetores de viés para o *gate* i . Nessa formulação, a saída u_t corresponde a interpolação linear ponderada pelo *gate* de atualização z_t entre o que havia anteriormente armazenado na célula u_{t-1} e o atual candidato $\tilde{u}_t$. O novo candidato corresponde a uma regulação do que havia anteriormente $r_t \odot u_{t-1}$ com o que há de novo na entrada a_t . É importante ressaltar que uma unidade GRU possui uma matriz de pesos a menos do que uma unidade LSTM, deixando o treinamento via *backpropagation* mais rápido e eficiente.

3.2.4 Avaliação de Resultados

A tarefa de normalização pode ser abordada como um problema de classificação em múltiplas classes, considerando a palavra ruidosa como entrada, a saída do normalizador como predição do modelo e a palavra correta como saída esperada. Outra abordagem é representá-la como classificação binária: se a correção proposta pelo modelo é a correta, isso caracteriza um acerto; senão, um erro. Com base nessa modelagem, é possível utilizar métricas tradicionais de avaliação de classificadores para avaliar métodos de normalização. Diferentemente da classificação comum, uma prática frequente na tarefa de normalização é não gerar apenas uma saída no modelo – mas sim um conjunto de possíveis candidatos a substituir a palavra com ruído. Nesse caso, em vez de avaliar a predição como uma

entidade única, são avaliadas as *top-n* predições, sendo *n* o número total de predições consideradas. Na prática, *top-n* representa os *n* candidatos mais prováveis a substituir a palavra ruidosa. Esse conceito teve origem na avaliação de sistemas de recomendação como uma maneira mais adequada de avaliar as diferentes recomendações propostas (KARYPIS, 2001).

Em geral, a avaliação de classificadores é realizada comparando-se a saída do modelo com o resultado esperado. Na normalização textual, a saída do modelo é o candidato mais provável a corrigir o ruído ou um conjunto de *n* candidatos mais prováveis. O resultado esperado é a palavra correta, que foi corrigida por um anotador humano no caso de avaliação em *cópus* anotados. As medidas mais frequentes na literatura para avaliar classificadores são a F-Measure (F_β), a Cobertura (C), a Precisão (P) e a Acurácia (AC) (POWERS, 2011). Cada uma delas pode ser calculada a partir da matriz de confusão (Tabela 4) que consiste em uma tabela que discrimina os tipos de acertos e erros cometidos em uma tarefa de classificação (tradicionalmente binária).

Tabela 4 – Matriz de confusão

		Classificador	
		Sim	Não
Cópus	Sim	VP	FN
	Não	FP	VN

Fonte: Elaborada pelo autor.

Contextualizando a matriz de confusão na tarefa de normalização, Verdadeiro Positivo (VP) indica que o ruído foi corrigido corretamente; Falso Positivo (FP) indica que o modelo modificou determinada palavra que já estava correta; Verdadeira Negativo indica que o modelo corretamente não normalizou determinada palavra e Falso Negativo (FN) indica que o modelo não normalizou uma palavra que deveria ter sido corrigida. Com base nessas informações, é possível calcular cada uma das métricas mencionadas:

- **Precisão (P)**: indica a proporção de palavras normalizadas que de fato deveriam ter sido corrigidas:

$$P = \frac{VP}{VP + FP}$$

- **Cobertura (C)**: considerando o total de palavras que deveriam ter sido normalizadas, indica a proporção do total que efetivamente foi:

$$C = \frac{VP}{VP + FN}$$

- **F-Measure (F_β)**: medida que relaciona precisão e cobertura por meio de uma média harmônica desses valores:

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot C}{\beta^2 \cdot P + C}$$

Costuma-se utilizar $\beta = 1$, o que dá origem à medida conhecida por F1-Measure, F1-Score, ou por simplicidade, F_1 :

$$F_1 = 2 \cdot \frac{P \cdot C}{P + C}$$

- **Acurácia (AC)**: indica o total de normalizações corretas dado o montante de palavras analisadas:

$$AC = \frac{VP + VN}{VP + VN + FP + FN}$$

Uma métrica similar às citadas, utilizada em alguns trabalhos de normalização, é a taxa de alarme falso (FAR, do inglês *false alarm rate*), calculada como:

$$FAR = \frac{FP}{FP + VN}$$

Essa medida representa a taxa de correções que o normalizador efetuou erroneamente – portanto quanto menor seu valor, melhor o resultado. Outra forma de avaliar o processo de normalização é analisando a correção proposta em si, ou seja, comparando diretamente a palavra gerada pelo modelo à esperada. Uma medida que possibilita esse tipo de análise é a taxa de erro de palavra (WER, do inglês *word error rate*). A WER compara a palavra gerada pelo modelo à palavra efetivamente correta. A comparação é feita por meio da distância de edição, considerando as operação de substituição (S), remoção ou deleção (D) e inserção (I). A WER é calculada como:

$$WER = \frac{S + D + I}{S + D + Co}$$

Co indica o número de caracteres corretos, que não necessitam de operações de edição. A medida WER possibilita uma avaliação mais metódica dos erros cometidos na normalização, permitindo compreender exatamente quão distante a palavra gerada está da correta. A taxa de erro de caractere (CER, do inglês *character error rate*) é uma versão mais granular da WER, considerando as edições caractere a caractere e não somente da palavra como um todo.

TRABALHOS RELACIONADOS

Neste capítulo são apresentados os principais trabalhos encontrados na literatura sobre normalização textual que de alguma maneira se relacionam ao propósito desse projeto. As seções são divididas de acordo com abordagens para tratar a tarefa de normalização. É importante ressaltar que essa divisão se refere ao modo que o problema é modelado. Sendo assim, é possível que trabalhos que utilizam técnicas distintas estejam dispostos na mesma seção. Os trabalhos serão apresentados em ordem cronológica. A [Seção 4.4](#) traz um resumo da revisão da literatura, sintetizando brevemente as principais características dos trabalhos abordados.

4.1 Baseados em Léxico

Esta seção apresenta trabalhos que efetuam o processo de normalização, total ou parcialmente, por meio do uso de léxicos manualmente construídos. De modo geral, esses léxicos são listas de palavras fora do padrão seguidas por suas respectivas correções. Essa abordagem é muitas vezes limitada, pois o uso de listas elaboradas manualmente não é escalável. No entanto, é possível que em domínios restritos ou em aplicações específicas esses léxicos tenham a capacidade de cobrir a maior parte dos ruídos.

4.1.1 Trabalho de [Clark e Araki \(2011\)](#)

Este trabalho tem uma abordagem baseada em léxicos para normalização de textos da *web* escritos em língua inglesa. Os autores propõem um sistema, denominado *Casual English Conversion System* (CECS), para identificar e corrigir erros e irregularidades tipicamente encontrados no uso casual da língua inglesa, principalmente em redes sociais. O sistema considera oito categorias de ruídos, sendo elas:

1. **Abreviações (encurtamento):** palavras escritas de forma mais curta que suas

versões padrão de modo a poupar espaço, podendo incluir misturas de números e letras (como *gr8* para se referir a *great* (ótimo)).

2. **Abreviações (acrônimos)**: utilizados em nomes de entidades e como abreviações de sentenças ou expressões, como *IMHO* para se referir a *in my humble opinion* (em minha humilde opinião).
3. **Erros de digitação/ortográficos**: grafia errada de palavras existentes, causada por erro de digitação (ao se pressionar teclas próximas acidentalmente, por exemplo) ou por desconhecimento da forma correta.
4. **Omissão ou erros de pontuação**: além de ausência de pontos para delimitar sentenças, um erro comum na língua inglesa é a falta de apóstrofo (como em *dont* em vez de *don't*).
5. **Gírias contextuais**: palavras existentes no léxico da língua mas que são usadas como gírias com outro significado. Um exemplo é *lit*, que oficialmente é a forma no pretérito do verbo *to light* (acender), mas que informalmente é utilizada como sinônimo de *legal*. A resolução dessa categoria de erro pode ser considerada um problema de desambiguação de sentido.
6. **Wordplay**: “brincadeira” com a forma da palavra, quando a grafia incorreta é utilizada propositalmente. Repetição de letras para dar ênfase à palavra (como *chaaato* em vez de *chato*, em português) é um exemplo frequentemente encontrado na linguagem informal da Internet.
7. **Evasão de censura**: uso de pontuação ou de números para disfarçar palavras obscenas e xingamentos. Muitas vezes é utilizada para burlar filtros de linguagem que comumente são aplicados a comentários em fóruns de discussão ou em *chats*.
8. **Emoticons**: combinação de símbolos geralmente utilizados para representar sentimento ou expressões. Embora facilmente reconhecidos por humanos, não têm significado direto para aplicações de PLN – por isso normalizá-los.

Para analisar os ruídos e criar os léxicos de normalização, os autores compilaram um corpus de textos escritos em inglês casual. O corpus é composto de 1.043 entradas, extraídas de publicações no Twitter e de comentários em vídeos do YouTube. As anotações são formadas por quatro informações: **palavra errada** (o ruído encontrado); **palavra regular** (a versão corrigida, sendo reconhecida pelo léxico da língua); **categoria do ruído**, seguindo as supracitadas e **notas**, anotações que contêm informações sociais e culturais sobre o ruído. A anotação foi validada por meio de consulta a dicionários colaborativos de inglês casual disponíveis na *web*.

Uma vez elaborado o léxico, a normalização é realizada simplesmente consultando-o e substituindo as palavras ruidosas. No caso de gírias contextuais, a entrada no léxico é composta por fragmentos de sentenças. Para exemplificar esse processo, a palavra *rite* pode ser utilizada. Embora oficialmente signifique *ritual*, ela é informalmente empregada para se referir a *right* (correto) por serem foneticamente equivalentes. Nesse caso, também podem ser anotadas algumas palavras anteriores ou posteriores para auxiliar a compreender o contexto. Se *rite* é antecedido por *is* (é), por exemplo, a probabilidade de se referir a *right* é alta, visto que ser usada como “*é ritual*” faz pouco sentido. Para as demais categorias de ruído, a normalização é efetuada por consulta direta ao léxico.

A avaliação do sistema é feita de duas maneiras: considerando-o como uma ferramenta para auxiliar falantes não-nativos de inglês na compreensão de textos, e como uma etapa de pré-processamento para tradução automática. Para avaliar o primeiro uso, dez pessoas aprendendo inglês, entre 23 e 64 anos, ranquearam sua compreensão a respeito de 20 sentenças retiradas do Twitter (antes e após a normalização) de 1 (muito básico) a 5 (excelente). Para as sentenças originais, a média de compreensão foi de 2,89 pontos. Para as normalizadas, foi de 3,89 – mostrando que a normalização aumentou 1 ponto na média de compreensão. Para avaliar o sistema como uma etapa de pré-processamento, 100 sentenças do Twitter foram traduzidas de inglês para japonês antes e depois da normalização. Os erros na tradução foram computados manualmente, sendo consideradas duas categorias: palavras não traduzidas (PNT) e palavras erroneamente traduzidas (PET). O tamanho médio das sentenças era de 15,35 palavras. Após a normalização, o número médio de PNTs caiu de 3,34 para 0,86 – uma melhora significativa. O número de PETs não foi informado.

O sistema utiliza uma abordagem de normalização simples, com consultas a léxicos construídos manualmente. Embora a forma de avaliação seja interessante, foi conduzida em pequena escala e com metodologia que não foi totalmente esclarecida. Para compreender de fato a eficácia do sistema, devem ser realizados experimentos mais amplos e que testem efetivamente a taxa de correção.

4.1.2 Trabalho de *Duran, Avanço e Nunes (2015)*

Este trabalho, que é decorrente de *Avanço (2015)*, propõe um sistema de normalização para textos de CGU escritos em português brasileiro. O sistema é restrito ao domínio de *reviews* de produtos, sendo que sua construção é baseada no *córpus* descrito em *Hartmann et al. (2014)*. O *córpus* é composto por 85.910 *reviews*, 4.088.718 *tokens* e 837.866 *types*. Para o sistema de normalização proposto, são considerados ruídos: erros ortográficos, capitalização incorreta (em nomes próprios e após pontos), internetês, acrônimos e siglas, palavras aglutinadas e ausência de pontuação.

O sistema proposto, denominado UGCNormal, é composto por módulos que atuam

de maneira sequencial. A maior parte dos módulos faz uso de léxicos para efetuar a correção de ruídos. A Figura 14 apresenta o *pipeline* de normalização seguido pelo UGCNormal.

Figura 14 – *Pipeline* de normalização da ferramenta UGCNormal

Fonte: [Avanço \(2015\)](#).

Inicialmente, o sistema recebe como entrada um texto com ruídos. Em seguida, o segmentador proposto por [Condori e Pardo \(2015\)](#) é utilizado para delimitar o texto em sentenças, que então são enviadas ao tokenizador. O módulo de tokenização foi projetado especificamente para textos da *web*. Segundo [Avanço \(2015\)](#), o uso de um tokenizador tradicional acarreta perda de termos importantes, como unidades de medida (30mb, 7gb) e *emoticons* (:), :-/). Considerando essas características, o tokenizador do UGCNormal foi desenvolvido de modo a identificar *emoticons*, unidades de medida e URLs.

Com o texto tokenizado, o próximo passo é aplicar o corretor ortográfico (também denominado *spell-checker*, *spelling-checker* ou *speller*). Esse é o único módulo do sistema que efetua normalização sem utilizar léxicos. Como grande parte dos erros ortográficos presentes no corpus tem influência da língua falada, optou-se pelo desenvolvimento de um *spell-checker* orientado a corrigir erros motivados pela similaridade fonética. A estratégia proposta pelo sistema consiste em obter uma lista de candidatos à correção, recuperando as palavras do léxico que estão a uma distância de edição igual a 1 ou 2 da palavra errada. Em seguida são aplicados algoritmos de checagem de proximidade fonética. O método Soundex, originalmente proposto por [Rusell \(1918\)](#), é um dos algoritmos utilizados e gera códigos numéricos iguais para letras que possuem o mesmo ponto de articulação (considera aspectos fisiológicos e articulatórios da produção da fala) ([AVANÇO, 2015](#)). O outro algoritmo utilizado consiste de regras fonéticas formuladas especialmente para o português brasileiro.

Considerando que a lista de candidatos à correção é inicialmente composta por palavras com distância de edição 1 ou 2 da palavra errada, o primeiro filtro utilizado para ranquear os candidatos é a frequência de ocorrência da palavra candidata no Corpus Brasileiro¹. Quanto maior a frequência, mais provável que esta seja a substituição mais plausível. Em seguida, as regras fonéticas para o PB são utilizadas de modo a encontrar candidatos à correção com o mesmo código fonético da palavra errada. As palavras *exce-*

¹ <corpusbrasileiro.pucsp.br/cb/Acesso.html>

ção e *esseção*, por exemplo, são foneticamente equivalentes. Caso nenhum candidato seja encontrado desse modo, o algoritmo Soundex, que tem funcionamento similar, é aplicado. A [Tabela 5](#) apresenta sugestões de correção dadas por cada técnica de ranqueamento, sendo que a palavra sublinhada indica a sugestão correta.

Tabela 5 – Sugestões de correção dadas por cada técnica de ranqueamento

Palavra Original	Frequência	Soundex	Regras-PB
amarzenamento	<u>armazenamento</u>	NENHUMA	NENHUMA
espectativa	<u>expectativa</u>	<u>expectativa</u>	<u>expectativa</u>
keijo	meio	<u>queijo</u>	<u>queijo</u>
fasia	casa	fase	<u>fazia</u>
xave	deve	sabe	<u>chave</u>

Fonte: Adaptada de [Avanço \(2015\)](#).

Após a correção ortográfica, os módulos restantes indicados na [Figura 14](#) são aplicados. Cada um deles consiste em um léxico que efetua uma correção específica. O módulo “Acrônimos_Map” tem a função de identificar acrônimos no texto e transformá-los em caixa alta. O módulo “Internetês_Map” realiza a substituição de formas do internetês para as esperadas da normal culta, como *vc* para *você* e *tb* para *também*. Por fim, o módulo “Nome_Próprio_Map” realiza a capitalização dos nomes identificados. Cada um desses módulos possui dicionários construídos manualmente para realizar a normalização, que têm as seguintes características:

- 21.669 nomes próprias
- 432 formas de internetês
- 248 palavras estrangeiras
- 18 siglas para unidades de medida
- 156 tipos de acrônimos genéricos

O UGCNormal foi avaliado de forma intrínseca e extrínseca. Para efetuar a avaliação intrínseca, foram utilizadas amostras contendo 60 *reviews* extraídas dos *websites* Buscapé e Mercado Livre. Ambas as amostras foram anotadas por especialistas, que identificaram e corrigiram os seguintes tipos de erros: erros ortográficos independentes de contexto (como *exelente* e *pesimo*), erros ortográficos dependentes de contexto (como *pratico* em vez de *prático* e *esta* em vez de *está*), internetês (como *vc* e *blz*), uso inapropriado de capitalização (nomes próprios, acrônimos e início de sentença), palavras aglutinadas

(como *muitobom* em vez de *muito bom*) e pontuação (apenas ponto final). A [Tabela 6](#) apresenta os resultados obtidos na avaliação intrínseca, seguindo o formato $X/Y = Z$, sendo X o total de correções efetuadas para cada categoria de erro, Y o total de erros anotados na amostra e Z o valor de cobertura obtido.

Tabela 6 – Resultado da avaliação intrínseca do UGCNormal

Tipo de erro/ruído	Buscapé	Mercado Livre
Ortográfico Não-contextual	50 / 56 = 0,89	87 / 108 = 0,80
Ortográfico Contextual	15 / 39 = 0,38	24 / 76 = 0,31
Internetês	4 / 6 = 0,67	15 / 25 = 0,60
Caixa (nomes próprios)	11 / 12 = 0,92	13 / 19 = 0,68
Caixa (início de sentença)	14 / 14 = 1,00	7 / 12 = 0,58
Palavras Aglutinadas	0 / 2 = 0	2 / 6 = 0,33
Pontuação	44 / 47 = 0,94	58 / 79 = 0,73

Fonte: Adaptada de [Avanço \(2015\)](#).

Embora, as amostras tenham o mesmo número de sentenças, o córpus do Mercado Livre apresentou maior número de ruídos. Devido a esse fato, a taxa de correção nessa amostra foi menor para todos os tipos de erro. De modo geral, o sistema obteve bons resultados para correção ortográfica não-contextual, comprovando a eficácia do *spell-checker* fonético. No entanto, a correção de erros ortográficos contextuais não foi satisfatória, pois não são utilizados métodos que levam em conta o contexto. Os resultados para correção de internetês foram intermediários, mostrando que a dependência de léxicos construídos manualmente é negativa.

A avaliação extrínseca do sistema foi realizada analisando o impacto da normalização na tarefa de etiquetagem morfosintática. O *tagger* MXPOST ([RATNAPARKHI, 1996](#)) foi utilizado para etiquetar uma amostra de 10 *reviews*. A acurácia obtida foi de 0,9135, sendo que em textos jornalísticos esse valor geralmente é próximo de 0,97. Em seguida, as *reviews* foram normalizadas utilizando o UGCNormal e o *tagger* foi novamente aplicado. Com os textos normalizados, a acurácia obtida foi de 0,9339 – totalizando uma melhora de 0,0204. O teste estatístico de Wilcoxon (*Signed-rank*) foi conduzido a fim de verificar a chance de a melhora na acurácia do *tagger* ter sido obtida ao acaso. Ao nível de significância de 5%, a hipótese de a melhora não ser significativa foi rejeitada, com *p-value* obtido igual a 0,02249.

O normalizador proposto por [Duran, Avanço e Nunes \(2015\)](#) traz contribuições relevantes à área. Seus pontos que mais merecem destaque são o fato de ser construído para o português brasileiro e prover um sistema completo de normalização – desde o pré-processamento até a saída do texto normalizado. Apesar de ser robusto, ainda é altamente

limitado. O uso de listas anotadas manualmente para efetuar a normalização impede que o sistema funcione de forma adaptativa e flexível. O fato de as listas terem sido criadas considerando um domínio específico também é negativo, visto que o funcionamento do normalizador pode ser prejudicado quando aplicado a outras modalidades de CGU. Para verificar essa possibilidade, é necessário efetuar testes mais amplos em outras amostras de textos.

4.1.3 Trabalho de *Stiilpen e Merschmann (2016)*

Esse trabalho propõe uma metodologia completa de pré-processamento para lidar com textos da *web* em português brasileiro. A metodologia foi construída com o objetivo de preparar textos oriundos principalmente de redes sociais, *blogs* e fóruns para serem utilizados por aplicações de mineração textual e de PLN. A avaliação do sistema é realizada considerando a tarefa de categorização de textos e de classificação de polaridade. A metodologia proposta divide a etapa pré-processamento em diversos módulos, sendo um deles o de normalização textual. São considerados como ruídos erros ortográficos, capitalização inválida de nomes próprios, internetês, palavras obscenas, pontuação incorreta e acrônimos. A [Figura 15](#) apresenta o *pipeline* de execução da metodologia.

Figura 15 – *Pipeline* de execução da metodologia proposta por [Stiilpen e Merschmann \(2016\)](#).

Fonte: Adaptada de [Stiilpen e Merschmann \(2016\)](#).

O sistema recebe como entrada textos da *web*. O módulo Detector de Idioma é responsável por filtrar os textos de modo que somente os escritos em língua portuguesa sejam considerados pelo sistema. O Normalizador Textual trata os ruídos mencionados utilizando o corretor ortográfico LanguageTool², o corretor gramatical CoGrOO³ e um conjunto de léxicos, tanto públicos quanto proprietários (desenvolvidos pelos autores). Não é especificado qual foi o procedimento utilizado para compilação de cada um. As diferentes categorias de ruídos são normalizadas com base em léxicos específicos, sendo que em sua totalidade o sistema emprega léxicos contendo:

- Gírias e expressões de internetês mais frequentes na *web*, com 500 entradas.

² <languagetool.org>

³ <cogroo.sourceforge.net/>

- Lista de palavras obscenas, com 226 entradas. Os autores não especificam o que são consideradas palavras obscenas, nem pelo que são substituídas.
- Nomes de organizações e de marcas, com 10.000 entradas.
- Nomes próprios de pessoas e locais, com 8.000 entradas.
- Palavras reconhecidas pela língua, por meio dos dicionários VERO⁴, Aspell⁵, LIWC⁶ e UNITEX-PB⁷.

O tokenizador utilizado pelo sistema é projetado para lidar com textos da *web*, sendo capaz de identificar *emoticos* e URLs. O Extrator de *Features* é responsável por extrair características morfológicas e sintáticas do textos para posteriormente serem utilizadas pelas aplicações. O Contextualizador tem como objetivo expandir as informações contidas no texto por meio do uso de dados estruturados e semi-estruturados da Wikipédia, WordNets e dicionários semânticos. O último módulo, denominado DTR (*Data Transformation and Reduction*), aplica algoritmos de discretização e de seleção de atributos nos dados extraídos do texto. Por fim, a saída do sistema pode ser utilizada por aplicações de mineração textual.

A metodologia é avaliada extrinsecamente em duas tarefas: categorização de textos e classificação de polaridades. Não é realizada avaliação intrínseca, ou seja, o pré-processamento não é avaliado por si só. A tarefa de categorização de textos consiste em classificar 600 *tweets* que falam sobre cerveja em uma das seis categorias: saúde e bem-estar; conhecimento cervejeiro; economia; ambiente; cultura/comportamento e consumo responsável. A tarefa de classificação de polaridades consiste em classificar 1630 *reviews* sobre aplicativos de celular como positivas ou negativas. Ambas tarefas são executadas antes e depois de cada etapa do *pipeline* de pré-processamento. Somente os resultados finais (após a execução de todas as etapas) serão descritos, por serem os mais relevantes. A Tabela 7 apresenta os resultados obtidos para as tarefas de categorização textual (CT) e de classificação de polaridades (CP), antes e depois da metodologia de pré-processamento (PP) ser aplicada.

Os resultados indicam que o pré-processamento acarretou em uma melhora significativa no desempenho das tarefas. O ganho na medida F1 foi de aproximadamente 10% na categorização textual e de 3,5% na classificação de polaridades. Desse modo, é possível concluir que a etapa de pré-processamento – incluindo a normalização textual – é de grande valia ao lidar com textos da *web*. Embora a metodologia baseada em léxicos seja simples, os resultados são promissores. No entanto, como os léxicos utilizados não são

⁴ <pt-br.libreoffice.org/projetos/vero/>

⁵ <ftp.gnu.org/gnu/aspell/dict/0index.html>

⁶ <143.107.183.175:21380/portlex/index.php/pt/projetos/liwc>

⁷ <nilc.icmc.usp.br/nilc/projects/unitex-pb/web/dicionarios.html>

Tabela 7 – Resultados obtidos pela metodologia proposta por [Stiilpen e Merschmann \(2016\)](#)

Tarefa	Precisão (%)	Cobertura (%)	F1 (%)
CT	73,57	70,23	71,21
CT + PP	82,64	80,95	81,40
CP	92,05	92,04	92,04
CP + PP	95,52	95,51	95,51

Fonte: Elaborada pelo autor.

descritos em detalhe, é difícil saber se a abordagem é escalável e se pode ser aplicada a outros tipos de texto de CGU. A ausência de uma avaliação intrínseca é outro ponto negativo do trabalho, pois impossibilita que se faça uma análise da eficácia da normalização textual efetuada pelo sistema.

4.2 Baseados em *Noisy-channel*

Esta seção apresenta trabalhos que utilizam a metáfora do *noisy-channel* para efetuar normalização textual. Grande parte das pesquisas iniciais na área eram baseadas nessa modelagem. O principal desafio de técnicas que utilizam *noisy-channel* é estimar a probabilidade do canal ruidoso. Para tanto, é necessário modelar como palavras reconhecidas pelo padrão se tornam ruidosas. Esse processo geralmente envolve o uso de *features* extraídas manualmente, muitas vezes utilizando conhecimento linguístico. Embora recentemente tenha perdido espaço para abordagens alternativas, o *noisy-channel* foi fundamental para o desenvolvimento da normalização textual, sendo utilizado em trabalhos altamente relevantes à área.

4.2.1 Trabalho de [Brill e Moore \(2000\)](#)

Trabalho de grande importância histórica por ser um dos primeiros a aplicar *noisy-channel* na tarefa de normalização. Embora a abordagem proposta pelo trabalho seja simples de acordo com os padrões atuais, ela foi de grande influência para dar início a esse nicho de pesquisa. Apesar de sua simplicidade, optou-se por apresentar esse trabalho devido ao seu contexto histórico.

A abordagem proposta é basicamente uma generalização da distância de Damerau-Levenshtein ([DAMERAU, 1964](#)). Considerando um alfabeto Σ , o modelo permite operações de edição da forma $\alpha \rightarrow \beta$, com $\alpha, \beta \in \Sigma^*$. $P(\alpha \rightarrow \beta)$ é a probabilidade de um usuário escrever α quando na verdade pretendia escrever β . O modelo pode ser descrito conforme as seguintes notações: seja $Part(w)$ o conjunto de todas as maneiras possíveis de particionar a *string* w em *substrings* adjacentes. Para uma partição específica $R \in Part(w)$, com

$|R| = j$ (R consiste de j segmentos contíguos), seja R_i o i -ésimo segmento. Segundo o modelo:

$$P(s|w) = \sum_{R \in \text{Part}(w)} P(R|w) \cdot \sum_{T \in \text{Part}(s)} \prod_{i=1}^{|R|} P(T_i|R_i)$$

Sendo que um par específico de alinhamentos para s e w induz um conjunto de operações de edição que permite derivar s a partir de w .

O treinamento do modelo necessita de pares formados por palavras incorretas s_i e suas versões corrigidas w_i . Para formar os pares, os caracteres de s_i são alinhados aos de w_i de modo a minimizar a distância de edição entre as palavras. Uma vez gerados os pares, é possível calcular a probabilidade de cada substituição $\alpha \rightarrow \beta$ como:

$$\frac{\text{contagem}(\alpha \rightarrow \beta)}{\text{contagem}(\alpha) \cdot \text{contagem}(\alpha \rightarrow \beta)}$$

Estimar $\text{contagem}(\alpha)$ é difícil, pois seu valor deveria ser correspondente à distribuição dos erros ortográficos na língua como um todo, não somente no *cópus*. No entanto, os resultados empíricos mostraram que esse valor não é tão relevante.

O modelo foi avaliado em um *cópus* contendo 10.000 erros ortográficos comuns na língua inglesa. 80% do *cópus* foi utilizado para treinamento (estimativa das probabilidades) e 20% para avaliação. O melhor resultado obtido, sem nenhuma janela adicional de contexto, foi de 87% de acurácia. É importante ressaltar que os erros considerados não são característicos de textos gerados por usuários. Por serem comuns na língua, são em grande parte causados por poucas operações de edição e mantêm relação significativa com a palavra correta. Portanto, é de se esperar que esses resultados não se repitam em ruídos encontradas em CGU, visto que estes são altamente desordenados e têm pouca estruturação definida.

4.2.2 Trabalho de Choudhury et al. (2007)

Trabalho focado na normalização de textos oriundos de troca de mensagens, como SMSs e *chats*. A linguagem empregada nesse tipo de comunicação é denominada *texting language* (TL) e apresenta elevado número de palavras fora do padrão, incluindo erros ortográficos, gramaticais, ausência de pontuação e muitas abreviações. O processo de normalização é modelado seguindo a metáfora *noisy-channel*. Considera-se que uma palavra correta é comprimida ou corrompida enquanto transmitida por meio de um canal ruidoso hipotético, se transformada em uma palavra de TL. Cadeias Ocultas de Markov (HMMs, do inglês *Hidden Markov Models*) são utilizadas para estimar as probabilidades necessárias no modelo *noisy-channel*.

Formalmente, o *noisy-channel* proposto é modelado como se segue. Considera-se $S = s_1, s_2, \dots, s_l$ uma sentença na forma padrão e s_i *tokens* incluindo palavras, pontuação

e abreviações. Quando S é transmitida através do canal ruidoso, ela é transformada em $T = t_1, t_2, \dots, t_l$, sendo T a sentença correspondente escrita em TL e t_i a versão corrompida do token s_i . O *noisy-channel* é então caracterizado pela probabilidade condicional $P(T|S)$. De modo a decodificar T em sua forma padrão $\delta(T)$, sendo δ o modelo do decodificador, é necessário determinar $P(S|T)$, que pode ser expressada como:

$$\delta(T) = \arg \max P(T|S) \cdot P(S)$$

Assumindo que a corrupção causada em t_i depende somente de s_i e não de seu contexto, a equação pode ser reescrita como:

$$\delta(T) = \arg \max \left[\prod_{i=1}^l P(t_i|s_i) \cdot P(S) \right]$$

Sendo assim, o modelo decodificador pode ser obtido com base no modelo de palavra $P(t_i|s_i)$ e no de língua $P(S)$. Portanto, para efetuar a normalização, basta estimar as probabilidades para cada modelo. O modelo de língua pode ser estimado com simples probabilidades obtidas por meio da observação do *cópus*.

O objetivo principal do modelo de palavra é representar o uso intencional de abreviações e não-intencional de erros de digitação encontrado em textos de TL. Para tanto, são utilizadas transições em cadeias de Markov representando caminhos grafêmicos e fonéticos que são percorridos dando origem à corrupção de palavras corretas. As transições no caminho grafêmico são efetuadas sobre os caracteres que constituem a palavra. Como abreviações fazem amplo uso de deleções, a transição vazia ϵ é permitida. De modo geral, as transições grafêmicas representam a probabilidade de ocorrência de cada cadeia de caracteres – sendo que o objetivo do modelo é aprender cadeias que resultam em palavras corretas. As transições fonéticas utilizam um algoritmo de conversão de grafemas para fonemas de modo a representar a pronúncia das palavras. Essa característica é útil para captar erros motivados pela fala. Atuando de modo análogo ao fonêmico, as transições fonéticas visam a capturar substituições de caracteres que ocorrem ao escrever a palavra da mesma maneira que ela é pronunciada.

A construção do HMM de cada palavra ocorre com base em treinamento feito sobre um *cópus* anotado. Esse *cópus* é constituído de 1.000 mensagens de texto (SMSs) escritas em inglês. As mensagens foram manualmente convertidas para forma padrão, sendo cada palavra corrigida. Os textos originais foram alinhados em nível de palavra aos corrigidos, resultando em um *cópus* paralelo com 20.000 pares de palavras. Esses textos foram utilizados tanto para treinamento do modelo quanto para avaliação. Para avaliação, utilizou-se um subconjunto de 1.228 *tokens* extraídos do *cópus*. Os resultados obtidos variaram de 89% a 97% de acurácia, dependendo da configuração do modelo. No entanto, a maior parte dos *tokens* pertencentes ao subconjunto já estavam na forma padrão, portanto não necessitava de normalização. Desse modo, a metodologia de avaliação não

é capaz de retratar realisticamente a eficácia do processo de normalização efetuado – impossibilitando a realização de uma análise ampla sobre o trabalho.

4.2.3 Trabalho de *Cook e Stevenson (2009)*

Apresenta um modelo não-supervisionado de normalização de textos de SMSs. É basicamente uma extensão da abordagem de *Choudhury et al. (2007)*. A mesma modelagem *noisy-channel* é utilizada. O diferencial dessa proposta é a forma de construção do modelo de palavra. Além de representar as transições grafêmicas e fonéticas, esse modelo também inclui transições específicas para lidar com outras características dos ruídos encontrados em mensagens de texto.

A primeira modelagem adicional é a de variações estilísticas, que são erros cometidos propositalmente de modo a refletir características pessoais dos usuários. Essa modelagem é calculada com uma versão probabilística da distância de edição, inspirada no trabalho de *Brill e Moore (2000)*. Nessa versão da medida, são permitidas operações de edição com dois caracteres. São estimadas duas probabilidades: (i) $P(g_t|g_s, pos)$, que é a probabilidade do grafema g_t ser escrito em vez de sua versão correta g_s na posição pos (que pode ser no início, meio ou fim da palavra) e (ii) $P(h_t|p_s, h_s)$, que é a probabilidade do grafema ruidoso h_t ser utilizado dados as formas corretas do fonema p_s e do grafema h_s na posição pos . A estimativa dessa probabilidade ajuda a captar erros motivados principalmente pela fala.

Uma modelagem focada em captar abreviações também é empregada. Para tanto, sua probabilidade é estimada de duas maneiras. Se o *token* t_i é uma subsequência de sua versão correta s_i , $P(t_i|s_i, subsequencia) = c$, sendo c uma constante que pondera a importância desse tipo de correção. Se t_i não é uma subsequência de s_i , $P(t_i|s_i, subsequencia) = 0$. Uma modelagem análoga a essa é empregada para representar “recortes de prefixo” (*prefix clipping*, em inglês) – um tipo de ruído, frequente na língua inglesa, no qual o prefixo da palavra é utilizado para se referir a ela como um todo.

O modelo é avaliado no corpus utilizado por *Choudhury et al. (2007)*. A avaliação é feita considerando acurácia *top-1*, *top-10* e *top-20*. Os resultados obtidos, seguindo essa ordem, foram de 59,4%, 83,8% e 87,8%. Os resultados foram em média 0,5% inferiores aos de *Choudhury et al. (2007)*. No entanto, essa abordagem é não-supervisionada e não necessita de treinamento com base em palavras alinhadas. As probabilidades são estimadas apenas usando as modelagens específicas mencionadas. Desse modo, a acurácia obtida é satisfatória e mostra que abordagens não-supervisionadas também obtêm sucesso na tarefa de normalização.

4.3 Baseados em Redes Neurais

Esta seção apresenta trabalhos que utilizam redes neurais para efetuar normalização textual. Esse tipo de arquitetura, principalmente com técnicas de *deep learning*, vem sendo utilizado com sucesso em diversas tarefas de PLN. Para normalização textual, seu uso é recente. Um fato que incentivou o uso de redes neurais nesse contexto foi a *shared task* para normalização lexical de *tweets*⁸, apresentada no *Workshop on Noisy User-generated Text* (WNUT) em 2015. Para execução da tarefa de avaliação conjunta, um corpus anotado de *tweets* foi disponibilizado. Os sistema participantes deveriam ser capazes de identificar as palavras fora do padrão e corrigi-las. Duas categorias de sistemas foram permitidas: os **não-restritos** (que podem fazer uso de quaisquer ferramentas e dados disponíveis publicamente) e os **restritos** (que podem somente utilizar ferramentas prontas, sem léxicos anotados e dados adicionais). A avaliação dos sistemas participantes foi efetuada por meio das métricas de precisão, cobertura e F1.

4.3.1 Trabalho de *Min e Mott (2015)*

Participante da *shared task* do WNUT 2015, foi o segundo classificado na competição na categoria de sistemas restritos. A abordagem proposta é dividida em três etapas. A primeira é responsável por filtrar *tokens* específicos do domínio do Twitter, como *hashtags*, URLs e nome de usuários. Esse tipo de informação é ignorado pois não acrescenta informações relevantes e dificulta o processo de aprendizado. A segunda etapa efetua a normalização baseada em dicionário – que é criado em tempo de execução com base nos *tweets* anotados. A última, implementa a normalização baseada na arquitetura neural proposta.

O primeiro passo realizado pelo sistema é um simples pré-processamento do texto de entrada. Primeiramente, todas as palavras são reescritas somente com letras minúsculas. Em seguida, *tokens* que se referem a URLs, *hashtags* e nome de usuários são identificados e deixados como estão para não interferirem nas demais etapas. Após o pré-processamento, um dicionário de normalização é criado utilizando as amostras anotadas. Cada erro encontrado no corpus é inserido no dicionário como uma entrada, sendo que cada uma de suas correções é também inserida como possível normalização. Caso haja uma única correção no corpus para determinado erro, a normalização sempre consistirá em um mapeamento direto – ou seja, esse erro sempre será corrigido com a mesma palavra. No caso de múltiplas correções, todas são adicionadas ao dicionário e a normalização será feita na próxima etapa. Correções desse tipo indicam que é necessário considerar informações contextuais para decidir qual a substituição mais adequada para o erro.

Palavras ambíguas (que possuem mais de uma correção possível no dicionário) ou

⁸ <noisy-text.github.io/2015/norm-shared-task.html>

fora do vocabulário (não contidas no dicionário) são normalizadas na última etapa, na qual se utiliza a arquitetura neural. O sistema modela a normalização textual como uma sequência de operações de edição (inserção, remoção e substituição) que transforma determinada palavra ruidosa em sua versão correta. Desse modo, a rede neural deve aprender as sequências de edição. Os dados de entrada da rede são codificados da seguinte forma: cada caractere é mapeado para um inteiro (de 0 a 66), visto que existem 67 caracteres distintos no corpúsculo. Uma palavra é então representada como a concatenação dos vetores *one-hot* de cada um dos caracteres que as compõem. A etiqueta morfossintática da palavra também é incluída no vetor, sendo que o *tagger* CMU Twitter (OWOPUTI *et al.*, 2013) é a ferramenta utilizada para etiquetamento. De modo a representar o contexto, o vetor final de entrada da rede é formado pelo vetor da palavra a ser normalizada (e sua respectiva *tag*), concatenado ao da palavra que a antecede e ao que a precede.

A saída da rede é representada como a sequência de operações de edição em nível de caractere que transformam a palavra ruidosa em sua versão corrigida. Para cada exemplo de treinamento, a distância de edição é utilizada para mapear cada transformação realizada para corrigir a palavra. Na correção de *vc* para *você*, por exemplo, a saída da rede seria modelada como “*nenhuma, inserir_o, nenhuma, inserir_ê*”. Cada caractere recebe uma instrução a respeito de qual operação de edição deve ser aplicada. Caso não deva ser alterado, a operação é *nenhuma*. Para reduzir o tamanho da saída, as operações somente são representadas até a última que deve ser efetivamente aplicada – ou seja, operações do tipo *nenhuma* só são descritas quando existe alguma operação de outro tipo após elas. Na normalização de *exesso* para *excesso*, por exemplo, a saída é codificada como “*nenhuma, nenhuma, inserir_c*”, sendo que o resto é omitido.

Uma vez codificados os dados de entrada e de saída, o treinamento da rede pode ser efetuado. A arquitetura neural utilizada é uma LSTM com 256 unidades ocultas, 25% de taxa de *dropout* e entropia cruzada multiclasse como função de custo. O treinamento foi feito utilizando *batches* de tamanho 16. O sistema foi avaliado com três configurações diferentes: utilizando apenas a rede neural, sem o dicionário de normalização (V1) ; utilizando todas as etapas sem informação contextual (V2) e utilizando todas as etapas com informação contextual (V3). A Tabela 8 apresenta os resultados obtidos (precisão, cobertura e F1) em cada uma das configurações.

É importante ressaltar que na avaliação final da *task* o sistema obteve, com sua melhor configuração, 81,75% de F1. Os resultados indicam que a rede neural funcionou melhor quando empregada em conjunto com o dicionário de normalização. Isso se deve ao fato de que o dicionário consegue corrigir com mapeamento direto uma parcela considerável de erros. Quando utilizada sem o dicionário, a rede neural tem de lidar com um número maior de correções – o que dificulta seu treinamento. O uso de informação contextual acarretou em uma melhora bem pequena nos resultados, mostrando que na

Tabela 8 – Resultados obtidos pela arquitetura proposta por Min e Mott (2015)

Configuração	Precisão (%)	Cobertura (%)	F1 (%)
V1	87,22	66,0	75,10
V2	89,19	70,62	78,75
V3	89,40	71,11	79,19

Fonte: Elaborada pelo autor.

maioria das vezes apenas a palavra incorreta é o bastante para efetuar a normalização. De modo geral, a arquitetura neural proposta é interessante por codificar os dados de maneira simples, sem uso de *features* complexas. Apenas os caracteres da palavra e sua etiqueta morfofossintática foram utilizados, tornando essa abordagem escalável.

4.3.2 Trabalho de Leeman-Munk, Lester e Cox (2015)

Participante da *shared task* do WNUT 2015, foi o terceiro classificado na competição na categoria de sistemas restritos. O modelo proposto consiste de três componentes: o Normalizador, que efetua a correção de palavras ruidosas, o *Flagger* (sinalizador), que indica quais palavras devem ser normalizadas e o *Conformer* (corretor), que tem como objetivo amenizar erros cometidos na etapa de normalização. O sistema recebe o texto de entrada, realiza o pré-processamento e o envia concomitantemente para o Normalizador e para o *Flagger*. O Normalizador encontra um candidato para correção visando a substituir a palavra incorreta e o *Flagger* determina se o candidato ou a palavra original serão utilizados. A saída do Normalizador é enviada ao *Conformer*, que a adapta conforme a lista do vocabulário aceito. Por fim, a saída de todos os componentes é enviada a um módulo de tomada de decisão que define se a normalização será ou não aplicada.

O elemento central que forma a base do Normalizador e do *Flagger* é uma rede neural *feed-forward* profunda. No Normalizador, a rede neural é modelada como um *denoising autoencoder* – arquitetura cujo objetivo é reconstruir entradas ruidosas da maneira que originalmente eram. Ela recebe como entrada a sequência de caracteres da palavra representada em um vetor numérico de dimensão fixa. A primeira camada da rede é responsável por aprender essa representação de acordo com os exemplos de treino, sendo que o vetor da palavra é a concatenação do vetor de cada um de seus caracteres. A saída da rede é um vetor de tamanho $w.v$, sendo w o número de caracteres da palavra e v o total de caracteres existentes no vocabulário. Desse modo, para cada caractere da palavra de entrada, a saída representa uma distribuição de probabilidade sobre todos os caracteres do vocabulário, determinando se o original deve ou não ser substituído e qual o substituto mais provável.

O *Flagger* determina o que deve e o que não deve ser normalizado, visto que apenas

9% do total de *tokens* do *cópus* necessita de normalização. Sua estrutura geral é a mesma do Normalizador, com exceção da última camada. Em vez de gerar uma distribuição de probabilidades sobre os caracteres, a saída da rede é modelada com apenas dois neurônios – um sinalizando que a normalização deve ser efetuada e o outro que não. O *Conformer* atua como uma etapa de regularização, garantindo que as correções propostas pela rede neural sejam de fato válidas. Para tanto, um dicionário é construído com todas as palavras corretas no *cópus*. Se a correção proposta pela rede não está contida no dicionário, ela é então substituída pela mais próxima (pela distância de edição) que consta no dicionário. Esse processo garante que somente palavras válidas sejam utilizadas para correção.

Os experimentos foram conduzidos com diversas configurações de arquitetura de rede. Apenas a melhor será apresentada. O Normalizador é projetado com duas camadas ocultas, contendo 2.000 neurônios cada, e 50% de taxa de *dropout*. O *Flagger* possui duas camadas ocultas, com 1.000 neurônios cada, e 75% de taxa de *dropout*. *Tokens* de URLs, *hashtags* e nomes de usuário foram desconsiderados, pois a rede neural teve dificuldade em processá-los corretamente. Na melhor configuração da rede, os resultados obtidos foram 90,12% de precisão, 74,37% de cobertura e 81,49% de F1, deixando o sistema na terceira colocação da competição com apenas 0,26% a menos de F1 que o segundo colocado. Os resultados indicam que até uma arquitetura neural relativamente simples, como uma *feed-forward*, é capaz de efetuar normalização textual satisfatoriamente. A abordagem proposta utiliza apenas os caracteres da palavra, não necessitando de nenhuma *feature* específica – fato que a torna facilmente replicável, podendo ser avaliada em outros domínios.

4.3.3 Trabalho de Ikeda, Shindo e Matsumoto (2016)

Abordagem que utiliza redes neurais recorrentes para normalização de textos escritos em língua japonesa. Não é feita uma categorização de erros nem uma especificação de o que se considera como ruído. A arquitetura neural utilizada é um modelo *encoder-decoder* (codificador-decodificador), que vem sendo utilizado com sucesso em tarefas de tradução automática (BAHDANAU; CHO; BENGIO, 2014; SUTSKEVER; VINYALS; LE, 2014; LUONG; PHAM; MANNING, 2015). Esse modelo consiste de duas redes neurais: uma responsável pela codificação e outra pela decodificação. O *encoder* mapeia a palavra de entrada para um vetor numérico de dimensão fixa e o *decoder* gera a sentença ou a palavra de saída. O modelo atua em nível de caractere, ou seja, uma palavra é representada como a concatenação dos vetores de seus caracteres. Diferente da língua inglesa e portuguesa, nas quais o conjunto de caracteres utilizados é pequeno, o japonês apresenta mais de 3.500 símbolos em seu sistema de escrita. Essa característica aumenta consideravelmente a complexidade do problema. A Figura 16 ilustra o modelo neural utilizado.

O *encoder* mapeia a sentença de entrada para um vetor h de dimensão fixa utilizando uma rede neural recorrente bidirecional GRU. A rede possui duas camadas paralelas

Figura 16 – Modelo *encoder-decoder* para normalização.

Fonte: Ikeda, Shindo e Matsumoto (2016).

em duas direções: uma processando a sentença a partir do primeiro caractere, gerando o vetor $\vec{h}$, e a outra a partir do último, gerando $\overleftarrow{h}$. O vetor codificado h é a concatenação do gerado por cada direção. De modo geral, o principal objetivo do *encoder* é gerar uma representação robusta das palavras, permitindo que posteriormente sejam decodificadas corretamente mesmo sendo modificadas devido a ruídos. Essa característica torna esse modelo adequado para tratar de normalização.

O *decoder* recebe como entrada a representação codificada de determinada palavra e tem como objetivo reconstruí-la de modo a obter uma palavra válida pertencente ao vocabulário. A arquitetura utilizada em sua construção é uma GRU com mecanismo de atenção, que decide quais partes da sequência de caracteres de entrada são mais relevantes para o aprendizado da rede (BAHDANAU; CHO; BENGIO, 2014). A saída do *decoder* é uma distribuição de probabilidades para cada caractere do vocabulário, indicando quais são os mais prováveis para substituição.

Para efetuar o treinamento desse tipo de arquitetura, é necessário um grande conjunto de dados anotados. No entanto, não há corpus com anotações de ruídos para o japonês grandes o suficiente. Para contornar esse problema, os autores propõem a criação de um corpus sintético. Para tanto, utilizam um conjunto de regras para inserir ruído em palavras corretas contidas em um corpus já existente de textos escritos em japonês contemporâneo. As regras foram criadas com base em observações feitas sobre amostras anotadas de *tweets* – portanto retratam de fato a natureza dos ruídos encontrados nesse tipo de textos. As amostras de *tweets* foram extraídas de um corpus com 1.831 sentenças anotadas, sendo que cada palavra com ruídos foi identificada e corrigida.

Os experimentos de avaliação são conduzidos com três treinamentos diferentes: somente com o corpus de *tweets*, somente com o corpus sintético e com ambos combinados.

O sistema proposto (EncDec) é comparado com uma abordagem baseada em regras (as utilizadas no corpus sintético) e com um CRF. A métrica de avaliação utilizada é o CER, calculado como a distância de edição entre a palavra predita pelo sistema e a forma correta, normalizada pelo total de caracteres. Essa medida indica quão próxima a previsão está da realidade, sendo que quanto menor o valor, menor o total de erros. A [Tabela 9](#) apresenta os resultados obtidos, sendo que a coluna *nenhuma operação* indica que nenhum método de normalização foi aplicado.

Tabela 9 – Resultados obtidos pelo modelo proposto por [Ikeda, Shindo e Matsumoto \(2016\)](#)

Método	CER (%)
Nenhuma operação	17,15
Regras	15,07
CRF (tweets)	13,71
CRF (sintético)	13,63
CRF (combinado)	13,43
EncDec (tweets)	87,07
EncDec (sintético)	13,54
EncDec (combinado)	13,43

Fonte: Adaptada de [Ikeda, Shindo e Matsumoto \(2016\)](#).

Os melhores resultados foram obtidos por meio do CRF e do modelo *encoder-decoder*, ambos treinados no corpus reunindo os dados manualmente anotados e os sintéticos. A redução no total de erros foi de 3,72% em relação a não efetuar normalização. É interessante notar que o modelo EncDec atingiu erro de 87,07% quando treinado somente nos dados anotados – mostrando que a quantidade de dados é de fato essencial para o aprendizado da rede neural. Como haviam poucos exemplos não-sintéticos, a arquitetura neural não foi capaz de aprender satisfatoriamente quando treinada somente neles, fazendo com que a taxa de erro obtida fosse exorbitantemente alta. De modo geral, a abordagem de criação de dados sintéticos foi proveitosa, melhorando os resultados obtidos. Desse modo, essa proposta é potencialmente útil para a realização desse projeto, visto que também há poucos dados anotados para o português brasileiro.

4.4 Síntese dos Trabalhos Discutidos

De modo a resumir as principais características dos trabalhos discutidos, a [Tabela 10](#) apresenta uma síntese de cada um. Apenas o melhor resultado de cada trabalho é mostrado.

Tabela 10 – Síntese dos trabalho de normalização textual apresentados

Trabalho	Tipo de Texto	Técnicas	Avaliação	Resultados
Brill e Moore (2000)	-	<i>Noisy-channel</i> com medidas de similaridade	Córpus de 10.000 erros ortográficos	87% de acurácia
Choudhury <i>et al.</i> (2007)	SMS	<i>Noisy-channel</i> com HMM	Córpus de 1.000 SMSs	89% de acurácia
Cook e Stevenson (2009)	SMS	<i>Noisy-channel</i> não-supervisionado	Córpus de 1.000 SMSs	87,8% de acurácia
Clark e Araki (2011)	De redes sociais	Léxicos	Somente extrínseca	-
Duran, Avanço e Nunes (2015)	<i>Reviews</i> de produtos	Léxicos	Córpus de 60 <i>reviews</i>	63% de cobertura média
Min e Mott (2015)	<i>Tweets</i>	Redes neurais (LSTM)	Córpus da <i>task</i> do WNUT	81,75% de F1
Leeman-Munk, Lester e Cox (2015)	<i>Tweets</i>	Redes neurais (MLP)	Córpus da <i>task</i> do WNUT	81,49% de F1
Ikeda, Shindo e Matsumoto (2016)	<i>Tweets</i>	Redes neurais (GRU)	Córpus de <i>tweets</i> com 1.831 sentenças	13,43% de CER
Stiilpen e Merschmann (2016)	<i>Tweets</i>	Léxicos	Somente extrínseca	-

Fonte: Elaborada pelo autor.

CÓRPUS DE CGU

Para investigar métodos de normalização flexíveis e capazes de lidar com diferentes tipos de textos, é fundamental realizar uma ampla análise da linguagem do CGU. Para tanto, diferentes tipos de ruídos presentes em diferentes modalidades de CGU devem ser coletados e investigados. Córpus de textos de CGU são fontes importantes de conhecimento e têm o potencial de fornecer todas as informações necessárias para efetuar essa análise. A próxima seção apresenta os córpus de ruídos que serviram como base para o desenvolvimento das técnicas de normalização.

Um córpus de ruídos é uma coleção de textos, geralmente advindos da *web*, que contém informações sobre palavras não reconhecidas pela língua. Um córpus anotado deve prover cada ruído na forma de uma tripla: a palavra em si (ou versão ruidosa), sua categoria e sua versão correta (ou normalizada). Por exemplo, na sentença *isso é uma ecessão*, a anotação do ruído deve ser formada por *ecessão* (o ruído em si), *erro ortográfico* (a categoria do ruído) e *exceção* (sua versão normalizada). Infelizmente, são poucos os recursos disponíveis para a língua portuguesa. Os córpus existentes são focados no domínio de *reviews* de produtos e não são abrangentes, já que se tratam apenas de textos com as mesmas características. Outro obstáculo é a falta de consenso na categorização de ruídos – os córpus seguem padrões de anotação ligeiramente diferentes. Com base nessas lacunas, optou-se por realizar a anotação de um novo córpus de ruídos, com textos de diferentes domínios e fontes, e com uma categorização que busca ser menos subjetiva. Este capítulo descreve primeiramente os dois córpus já existentes para o português brasileiro e, por fim, apresenta o novo córpus, resultado deste trabalho.

5.1 Córpus Buscapé

Córpus de *reviews* de produtos extraídas do Buscapé, apresentado em [Hartmann et al. \(2014\)](#). A anotação foi baseada em um total de 85.910 *reviews*, contendo 4.097.905

tokens e 68.633 palavras únicas. Inicialmente, uma tentativa de correção dos ruídos por meio de corretores ortográficos (como o Aspell¹) foi realizada. No entanto, a taxa de correção obtida foi baixa – apenas 42,55% dos *tokens* foram normalizados corretamente pelo Aspell. Nesse contexto, optou-se por efetuar uma investigação mais ampla dos ruídos, dando início, assim, à anotação do cópus. Os ruídos mais frequentes encontrados nas *reviews* (com frequência mínima de 3) foram incluídos na anotação. A categorização de ruídos utilizada foi descrita detalhadamente na seção [Seção 2.3](#). Ao todo, quatro pares de anotadores anotaram 5.775 *tokens*. A tarefa de anotação contou com duas etapas de treinamento. Na primeira, 100 *tokens* foram anotados individualmente por cada anotador e em seguida a anotação foi revisada em grupo. Na segunda etapa, o mesmo processo foi seguido para anotar mais 75 *tokens*. Embora possua um número considerável de ruídos anotados e uma categorização definida, o cópus ainda assim apresenta deficiências. Os principais pontos negativos a serem destacados são:

- Não existe categoria específica para erros ortográficos contextuais (palavras contidas no léxico, mas incorretas no contexto em que se encontram), portanto eles foram anotados simplesmente como erros ortográficos comuns. Na perspectiva da normalização, esse ponto é uma falha pois a correção de erros ortográficos contextuais faz uso de técnicas específicas, diferentes das demais. A ausência dessa categoria impede que essas técnicas sejam avaliadas.
- Palavras corretas, mas não contidas no léxico considerado como padrão para a anotação, são categorizadas como erros ortográficos. Essa escolha pode acarretar problemas para a avaliação de um normalizador, pois pode forçar a correção de palavras que já estavam corretas.
- A categoria *internetês* é totalmente subjetiva, tornando difícil compreender quando um ruído pertence a ela. A categoria não possui uma definição completa, mas uma de suas características apresentadas é a tentativa de escrever a palavra da maneira que ela é pronunciada – portanto tornando o *internetês* como um ruído que visa a exprimir características próprias do autor. Embora isso possa ser claro em alguns casos, geralmente a linha entre o erro intencional para exprimir características do autor e o erro ortográfico, por exemplo, é muito tênue. Na descrição da categoria, o ruído *produitu* é apresentado como um exemplo de *internetês* para designar *produto*. No entanto, não é claro saber porque não é apenas um erro ortográfico.
- A categoria de unidades de medida é extremamente específica, aparecendo em apenas 0,45% dos *tokens*. Desse modo, ela aumenta desnecessariamente a complexidade da anotação e acrescenta pouca informação ao cópus anotado.

¹ aspell.net

- Existe uma grande quantia de *tokens* sem categoria (14.5%). Ruídos com essa anotação são diversificados e não seguem padrões, tornando assim difícil normalizá-los efetivamente.

Apesar de prover informações interessantes para uma análise linguística de ruídos em CGU, o *córpus Buscapé* possui empecilhos para ser usado como base de desenvolvimento e avaliação de técnicas de normalização.

5.2 *Córpus UGCNormal*

Utilizado no trabalho de [Duran, Avanço e Nunes \(2015\)](#), é um *córpus* de *review* de produtos que tem anotação com o foco em servir como base para avaliação de técnicas de normalização. O *córpus* é composto por 60 *reviews* extraídas do *site* Mercado Livre e 60 extraídas do *site* Buscapé. Ambas foram anotadas por um especialista, sendo que cada ruído foi dividido em uma das seguintes categorias: erros ortográficos não-contextuais, erros ortográficos contextuais, internetês, uso inapropriado de caixa (nomes próprios, acrônimos e início de sentença), palavras aglutinadas (como *muitobom* em vez de *muito bom*) e ausência de pontuação (apenas ponto final). A distribuição de cada categoria pode ser vista na [Tabela 11](#).

Tabela 11 – Distribuição de categorias do *Córpus UGCNormal*.

Tipo de ruído	Buscapé	Mercado Livre	Total
Ortográfico Não-contextual	56	108	164
Ortográfico Contextual	39	76	115
Internetês	6	25	31
Caixa (nomes próprios)	12	19	31
Caixa (início de sentença)	14	12	26
Palavras aglutinadas	2	6	8
Pontuação	47	79	126

Fonte: Adaptada de [Avanço \(2015\)](#).

O *córpus UGCNormal* trouxe melhorias em relação ao *Buscapé*, dividindo os ruídos em categorias que facilitam a avaliação de técnicas de normalização. No entanto, algumas limitações ainda persistem. A categoria *internetês*, por exemplo, continuou seguindo o mesmo esquema de anotação do *córpus Buscapé* – portanto apresentando a mesma subjetividade. As duas categorias relacionadas à pontuação (*Pontuação* e *Caixa (início de sentença)*) não estão diretamente relacionadas às técnicas de normalização, visto que no

trabalho de [Avanço \(2015\)](#) a segmentação de sentenças é realizada por uma ferramenta externa. Nesse contexto, essas categorias aumentam a complexidade da anotação e agregam pouca informação. A categoria de palavras aglutinadas sofre do mesmo problema – sua frequência é muito baixa e a correção não é feita pelo normalizador. Apesar de essas características serem negativas, o maior problema do *córpus* é seu tamanho: há poucos ruídos anotados. A categoria de nomes próprios e a de internetês, por exemplo, contam apenas com 31 *tokens* cada. Essa baixa quantidade dificulta a análise da capacidade efetiva de correção do normalizador.

5.3 **Córpus TUB-UGC**

Pautando-se nas deficiências presentes nos *córpus* apresentados e no fato de ambos conterem apenas *reviews* de produtos, optou-se por criar um novo *córpus* de ruídos para o desenvolvimento deste trabalho. Considerando-se que um dos objetivos do projeto é tratar ruídos de forma independente de domínio, é necessário diversificar a fonte dos textos. Para garantir que sejam considerados ruídos de diversas origens, o *córpus* é composto por amostras de textos advindos de três modalidades de CGU: *reviews* de produtos, discussões em fóruns e publicações no Twitter. Por esse motivo o nome TUB-UGC foi escolhido: **T**witter, **U**OL e **B**uscapé, todos textos de **UGC** (*user-generated content*). As amostras de *reviews* são oriundas do *córpus* de [Hartmann et al. \(2014\)](#) e já são utilizadas na avaliação do UGCNormal, então somente foram anotados mais exemplos de ruídos. A amostra de discussão em fóruns foi extraída de tópicos publicados no Fórum da UOL, nas categorias Televisão², Esporte³, Jogos⁴ e Vale Tudo⁵, que trata de assuntos variados. A amostra de publicações no Twitter consiste de *tweets* escritos em português extraídos de junho a dezembro de 2016, sem filtros específicos de busca.

Coletou-se um grande número de textos de cada modalidade – milhões de *tweets* e dezenas de milhares de *posts* no fórum da UOL e de *reviews* do Buscapé. Após a coleta, cada texto foi segmentado em sentenças, utilizando-se a ferramenta proposta por [Condori e Pardo \(2015\)](#), e posteriormente *tokenizado*. *Tweets* foram todos considerados como uma sentença, por terem tamanho máximo de 140 caracteres. É importante ressaltar que o segmentador delimitou, como uma única sentença, alguns textos longos (como em *posts* da UOL) em que o autor não usou pontuação alguma – fazendo com que houvesse sentenças com um número muito alto de *tokens*. Depois do pré-processamento, o total de sentenças coletadas foi de 294.711 do Buscapé, 68.000 do Fórum da UOL e 1.954.000 do Twitter. Apenas uma parcela do total de sentenças foi utilizada, sendo posteriormente

² <forum.televisao.uol.com.br>

³ <forum.esporte.uol.com.br>

⁴ <forum.jogos.uol.com.br>

⁵ <forum.jogos.uol.com.br/vale-tudo_f_57>

distribuída aleatoriamente entre os anotadores. A [Tabela 12](#) traz estatísticas sobre as diferentes amostras de CGU. A coluna *Fonte* indica a modalidade do texto, *Maior Sentença* indica a sentença com o maior número de *tokens*, *Tamanho Médio* indica o número médio de *tokens* por sentença e *Desvio Padrão* é o desvio padrão em relação à média.

Tabela 12 – Estatísticas sobre amostras de CGU.

Fonte	Maior Sentença	Tamanho Médio	Desvio Padrão
Buscapé	387	11.86	12.33
UOL	518	16.67	16.11
Twitter	47	8.37	6.01

Fonte: Elaborada pelo autor.

O número elevado de *tokens* na maior sentença de cada modalidade é devido à inconsistência da ferramenta de segmentação – que também acarreta desvio padrão alto. Como os *tweets* têm limite máximo de 140 caracteres, o número médio de *tokens* em cada um não está muito abaixo das demais modalidades. Isso se deve principalmente ao fato de *tweets* frequentemente conterem *emojis* (que contam como um *token* mas são um único caractere), *emoticons* e caracteres soltos. Essas estatísticas, embora simples, já indicam que diferentes modalidades de CGU apresentam comportamentos distintos em relação a desvios da língua.

O processo de Anotação

A anotação seguiu o procedimento dos demais cópus – cada ruído foi normalizado (quando pertinente) e categorizado. Para facilitar a anotação, os ruídos foram pré-categorizados utilizando a ferramenta UGCNormal. Caso determinada palavra não estivesse contida no léxico padrão da língua, ela recebia a categoria dada pelo UGCNormal. Caso a ferramenta não conseguisse identificá-la, a palavra era marcada como ruído e o anotador definia a categoria. Note que, mesmo que a categoria fosse pré-determinada, o anotador deveria confirmar a marcação – portanto toda escolha feita pelo UGCNormal foi revisada. A correção do ruído era necessária em casos de desvios não contidos em léxicos do UGCNormal, como erros ortográficos e abreviações. A [Figura 17](#) ilustra a *interface* de anotação.

Marcações em amarelo indicam que a palavra é ruidosa e recebeu uma categoria pelo UGCNormal. Nesses casos, o anotador deveria ainda assim confirmar se a categoria estava correta. Marcações em vermelho indicam que não houve sugestão de categoria por parte do UGCNormal e, portanto, o anotador deve indicar por si só. A [Figura 18](#) ilustra o processo de seleção de categoria e de inserção da correção.

Figura 17 – *Interface de anotação do córpus TUB-UGC.*

Fonte: Elaborada pelo autor.

Figura 18 – *Interface da seleção de categorias e inserção de correção.*

Fonte: Elaborada pelo autor.

O sistema Brat⁶ foi escolhido para efetuar a anotação, sendo que a versão utilizada pode ser visualizada em <http://nilc.icmc.usp.br/brat/index.xhtml#/Ruidos/>. A categorização utilizada na anotação do córpus TUB-UGC é baseada na dos córpus Buscapé

⁶ brat.nlplab.org

e UGCNormal, mas foi planejada de modo a amenizar as deficiências das categorizações anteriores. Abaixo, estão listadas as diretrizes seguidas pelos anotadores para categorizar os ruídos:

- **Erro Ortográfico (O)**: grafia incorreta de palavras existentes no dicionário que resulta em uma palavra fora do padrão. Em geral, um erro ortográfico corresponde a um erro de digitação – com troca de letras, ausência de acentos etc. Alguns exemplos são: **nao** em vez de **não**, **ecesso** em vez de **excesso**, **desing** em vez de **design** e **tdos** em vez de **todos**. Erros ortográficos **devem ser corrigidos** pelo anotador.
- **Erro Ortográfico Contextual (OC)**: ocorre quando uma palavra válida (presente no dicionário) é utilizada incorretamente no lugar de outra também válida. Em geral é causado por ausência de um único acento. Alguns exemplos são **esta** em vez de **está**, **porem** em vez de **porém** e **noticia** em vez de **notícia**. Para corrigi-lo, é necessário considerar contexto, pois ambas as formas da palavra (com e sem acento) existem. Devem ser corrigidos quando a versão atual da palavra está errada no contexto.
- **Sigla (SI)**: é uma sequência de letras com o objetivo de abreviar palavras. É importante notar que a expansão da sigla deve **sempre** resultar em mais de uma palavra. Alguns exemplos são **ONU** para **Organização das Nações Unidas**, **BBB** para **Big Brother Brasil** e **RJ** para **Rio de Janeiro**. Embora siglas geralmente correspondam a nomes próprios (outra categoria de ruído), como os exemplos anteriores, a prioridade é marcá-las como SI. Siglas só precisam ser corrigidas no caso de não terem sido previamente categorizadas como tal.
- **Abreviação (AB)**: versão reduzida de uma palavra. Diferentemente das siglas, as abreviações representam uma **única** palavra. Alguns exemplos são **pq** para **porque**, **tbm** para **também**, **ctz** para **certeza**, **s** para **sim** e **n** para **não**. Note que pode haver diferentes abreviações para a mesma palavra, como **qd**, **qnd** e **qndo** para **quando**. Abreviações **devem** ser corrigidas para sua forma expandida.
- **Internetês (IN)**: gírias e expressões usualmente empregadas na linguagem da Internet. Ruídos são categorizados como IN somente em situações específicas, tais como:
 1. Aumentativos/diminutivos: **mozão** para **amor** e **coisica** para **coisa**.
 2. Repetição de caracteres: **lindooo** para **lindo** e **muuuuito** para **muito**.
 3. Gírias e expressões que podem ser transformadas em uma palavra válida: **topzera** para **top**, **ain** para **ai**, **eh** para **é** e **nois** para **nós**.

Ruídos nessa categoria **devem ser corrigidos**.

- **Estrangeirismo (EN)**: palavras de outros idiomas. Não são corrigidas.
- **Nome Próprio (NP)**: nome de pessoa, lugar, entidade etc. Ruídos desse tipo não são corrigidos, apenas categorizados.
- **Sem Categoria (SC)**: palavras que não ocorrem nos léxicos e que não se enquadram em nenhuma das categorias acima. Palavras aglutinadas (ex. **meunome**) ou separadas (ex. **ba ra to**) encaixam-se nessa categoria.
- **Já corrigido**: palavras que receberam a classe Ruído, provavelmente por incompletude de algum léxico usado na pré-anotação, mas que estão corretas e devem permanecer como estão. Logo, não precisam de qualquer correção.
- **Lixo**: são *tokens* que não podem ser transformadas em palavras reconhecidas pela língua. Risadas (como **kkk** e **hahaha**) e sequências aleatórias de caracteres são exemplos.

Pode-se notar que essa categorização é capaz de preencher lacunas que existiam nas outras. A categoria internetês foi reduzida a casos específicos e bem definidos, diminuindo assim sua subjetividade. A categoria *Já corrigido* foi criada para lidar com palavras corretas não contidas no léxico, que no caso do *córpus* Buscapé eram consideradas como erros ortográficos. A categoria *Lixo* foi criada para eliminar *tokens* sem sentido e, conseqüentemente, reduzir a quantia de marcações *Sem categoria* – que existiam em grande número no *córpus* Buscapé. Além disso, categorias relacionadas a ferramentas externas (como as de pontuação) foram removidas, por não serem de suma importância à avaliação de técnicas de normalização.

A anotação foi dividida em três etapas. A primeira teve como objetivo a definição das categorias e elaboração do manual do anotador (disponível no [Apêndice A](#)), seguida pela anotação, por três anotadores, de 100 sentenças cada um. Após uma reunião com esses anotadores, as categorias foram mais bem-definidas e a anotação pode efetivamente começar. Ao todo, cinco anotadores participaram do processo e receberam treinamento prévio. Uma reunião com todos os anotadores para esclarecimento de dúvidas e para anotação conjunta de algumas amostras também foi realizada. A segunda etapa contou com a anotação de, em média, 150 sentenças por anotador. Ao término dela, uma análise sobre a distribuição das categorias dos ruídos anotados até então foi realizada e as amostras da terceira etapa foram escolhidas de modo a tentar balancear o *córpus*. Ao fim do processo, cada anotador etiquetou em média 300 sentenças. 80 sentenças foram anotadas em comum por todos para realizar o cálculo de medidas de concordância.

No total, foram anotados 12.405 *tokens*, sendo 3699 palavras únicas. A [Figura 19](#) apresenta a distribuição dos ruídos no *córpus*, por categoria.

Figura 19 – Distribuição de ruídos por categoria no *córpus TUB-UGC*.

Fonte: Elaborada pelo autor.

Cada fatia da [Figura 19](#) representa uma categoria de ruído. Os valores de proporção da categoria em relação ao total (%) e o número total de *tokens* pertencentes a ela também são apresentados. É importante ressaltar que uma mesma palavra pode ser classificada em diferentes categorias, dependendo do contexto em que ela aparece. A palavra *sky*, por exemplo, pode tanto ser um nome próprio (a empresa Sky) como um estrangeirismo, referindo-se à palavra inglesa correspondente a *céu*. Outro ponto importante é que os números se referem à quantidade de ***tokens*** – ou seja, incluem palavras repetidas. Os ruídos mais frequentes do *córpus* apresentam um elevado número de ocorrências. A [Figura 20](#) apresenta a frequência dos dez ruídos mais comuns.

Pode-se notar que abreviações predominam nas palavras mais frequentes, pois são muito utilizadas para poupar tempo de digitação. Para analisar a distribuição de categorias das palavras únicas (ou seja, desconsiderando a frequência), é necessário determinar apenas uma categoria para ela. Caso a palavra tenha mais de uma categoria possível, aquela com maior número de anotações é escolhida. Por exemplo: se *sky* foi anotada cinco vezes como estrangeirismo e duas vezes como nome próprio, estrangeirismo é escolhido como sua categoria predominante. A [Figura 21](#) apresenta a distribuição de categorias para palavras únicas no *córpus*, considerando a categoria predominante.

É possível notar que ao analisar palavras únicas, houve significativa mudança na distribuição de categorias. Nomes próprios continuam com a maior frequência, pois há uma grande diversidade de nomes de pessoas, entidades, marcas, etc. e, portanto, a repetição

Figura 20 – Frequência dos dez ruídos mais comuns no *córpus* TUB-UGC.

Fonte: Elaborada pelo autor.

Figura 21 – Distribuição de ruídos por categoria para palavras únicas no *córpus* TUB-UGC.

Fonte: Elaborada pelo autor.

de nomes idênticos é menor. No entanto, abreviações passaram de uma distribuição de 23,3% do total para apenas 3,92%. Essa mudança se deve principalmente ao fato de que um grande número de abreviações é utilizado com muita frequência, como visto na

Figura 20. As demais categorias tiveram pouca alteração. Outra análise interessante é observar a distribuição de categorias nas diferentes modalidades de CGU (UOL, Twitter e Buscapé). A Figura 22 apresenta a distribuição de categorias dos *tokens* anotados em *posts* no fórum da UOL.

Figura 22 – Distribuição de ruídos por categoria em *posts* no fórum da UOL.

Fonte: Elaborada pelo autor.

Embora nomes próprios continuem com a maior distribuição, sua proporção é consideravelmente maior nos textos da UOL – 12 pontos percentuais acima da distribuição geral. Essa característica pode ser explicada pela natureza dos fóruns de discussão. Em geral, tópicos de discussão se referem a determinadas entidades (como carros, programas televisivos e jogos), portanto é natural que esses nomes próprios, bem como outros pertinentes à discussão, sejam utilizados nos *posts*. Abreviações tiveram frequência menor que na distribuição geral, sendo que erros ortográficos assumiram a segunda colocação. A Figura 23 apresenta a distribuição de categorias dos *tokens* anotados em *reviews* de produtos do Buscapé.

Apesar de *reviews* de produtos em geral conterem diversos nomes próprios (como de marcas, empresas, lojas e produtos em si), essa categoria foi apenas a segunda mais frequente nos textos do Buscapé. No entanto, sua frequência ainda assim foi ligeiramente superior a da distribuição geral. A categoria de ruído mais presente nas *reviews* foi a de erros ortográficos – indicando que os autores cometem vários erros de digitação ou que desconhecem a grafia correta das palavras. Siglas tiveram um aumento considerável de frequência (cerca de 75%), possivelmente devido a nome de marcas e de características de produtos (como televisores de *LED*, por exemplo). As demais categorias apresentaram

Figura 23 – Distribuição de ruídos por categoria em *reviews* do Buscapé.

Fonte: Elaborada pelo autor.

comportamento similar ao geral. Por fim, a figura [Figura 24](#) apresenta a distribuição de categorias dos *tokens* anotados em *tweets*.

Figura 24 – Distribuição de ruídos por categoria em *tweets*.

Fonte: Elaborada pelo autor.

A categoria predominantes nos *tweets* foi a de abreviações, possivelmente devido à

restrição ao número máximo de caracteres nos textos. Outro ponto importante é que *tokens* marcadas como lixo tiveram frequência relativamente alta. Isso se deve principalmente a dois fatos: primeiramente, apesar de a coleta de *tweets* ter sido realizada de modo a considerar apenas aqueles escritos em português, alguns, escritos totalmente em outros idiomas, foram coletados. Nesse caso, os anotadores foram instruídos a marcar todos os *tokens* como lixo. Por fim, outra característica que contribui para o aumento dessa categoria é o uso frequente de risadas (*kkkk*, *huehueh*) no Twitter – que também são categorizadas como lixo. As demais categorias apresentaram comportamento bem similar ao geral.

A diferença entre as modalidades de CGU pode também ser vista ao analisar os ruídos mais frequentes de cada uma. A [Tabela 13](#) apresenta os dez ruídos com maior número de ocorrências para cada modalidade de CGU, ordenados de maneira decrescente de acordo com a frequência.

Tabela 13 – Os 10 ruídos mais frequentes em cada modalidade de CGU.

UOL	Buscapé	Twitter
q	tv	q
bbb	nao	n
vc	camera	vc
nao	hd	to
voce	facil	pq
pq	samsung	ta
diana	lg	p
ja	ja	nao
fernando	android	mt
dilma	otimo	eh

Fonte: Elaborada pelo autor.

Pode-se notar que os ruídos mais frequentes também seguem a distribuição geral da modalidade a qual pertencem. Em textos da UOL, três dos ruídos mais frequentes são nomes próprios – indicando que, de fato, ocorrem muitas vezes nos textos. Já no Buscapé, há várias palavras relacionadas a produtos: como *tv* e *samsung*. Outro ponto interessante é a presença de palavras sem acentuação, corroborando o número elevado de erros ortográficos nesse tipo de texto. Por fim, a maior parte dos ruídos presentes no Twitter são relacionados à linguagem empregada na *web*, contendo abreviações e grafias alternativas (como *eh* em vez de *é*).

Para avaliar a concordância, duas medidas foram utilizadas: o Kappa de Fleiss e a

concordância absoluta. O Kappa de Fleiss é uma medida estatística usada para calcular a concordância entre um número fixo de anotadores (FLEISS; COHEN, 1973). Seu valor varia de -1 a 1, sendo que valores abaixo de 0 indicam concordância muito ruim e acima de 0.81, concordância quase perfeita. Diferente do Kappa de Cohen (COHEN, 1960), o de Fleiss pode ser aplicado a qualquer número de anotadores e não a apenas um par. A concordância absoluta indica quantas vezes a maioria dos anotadores concorda com a etiqueta – ou seja, no caso de 5 anotadores, sempre que pelo menos 3 concordarem há concordância absoluta. No cópus TUB-UGC, a concordância foi avaliada separadamente para categorização e para correção dos ruídos. A [Tabela 14](#) apresenta os valores de concordância obtidos.

Tabela 14 – Valores de concordância obtidos na anotação.

Medida	Kappa de Fleiss	Concordância Absoluta
Categorização	0,70	75,36%
Correção	0,83	88,36%

Fonte: Elaborada pelo autor.

Para categorização, esse intervalo de valores indica concordância substancial. Já para correção, os valores indicam concordância quase perfeita (CARLETTA, 1996).

MÉTODOS DE NORMALIZAÇÃO TEXTUAL

Neste capítulo, são apresentados métodos de normalização textual investigados neste trabalho. As técnicas desenvolvidas têm como objetivo melhorar a qualidade de textos oriundos da *web*, consequentemente aumentando o desempenho de tarefas de PLN quando aplicadas em CGU. Métodos de normalização podem ser divididos em três etapas: identificação de ruídos, geração de candidatos e ranqueamento. A primeira etapa é responsável por identificar quais palavras de determinado texto necessitam passar pelo processo de normalização. A segunda, dada uma palavra ruidosa, busca encontrar palavras reconhecidas pela língua candidatas a substituí-la. Em seguida, a etapa de ranqueamento tem como objetivo encontrar o candidato mais adequado para normalização entre as sugestões geradas. A apresentação dos métodos seguirá essa divisão, ou seja, técnicas de identificação de ruídos, geração de candidatos e de ranqueamentos são desenvolvidas e descritas separadamente.

A [Seção 6.1](#) introduz o ponto de vista sob o qual a tarefa de normalização será tratada neste trabalho, apresentando o modo de lidar com cada categoria de ruído. A [Seção 6.2](#) descreve duas abordagens para identificação de ruídos e analisa as vantagens e as desvantagens de cada uma. A seção [Seção 6.3](#) trata sobre métodos de geração de palavras candidatas à substituição de ruídos. Analogamente, a [Seção 6.4](#) apresenta técnicas para ranquear os candidatos gerados de modo a selecionar os mais adequados para a normalização. Por fim, a [Seção 6.5](#) introduz a ferramenta de normalização desenvolvida neste trabalho, que realiza o processo completo de normalização – desde o pré-processamento até efetivamente a correção de ruídos.

6.1 Considerações Iniciais

Como mencionado, a ferramenta UGCNormal, proposta por [Duran, Avanço e Nunes \(2015\)](#), parece ser o único normalizador de conteúdo gerado por usuário disponível

para o português brasileiro. Desse modo, é natural considerá-lo como referência para se propor novas técnicas de normalização. Nesse contexto, há lacunas não preenchidas pelo UGCNormal que motivam a investigação de novos métodos de normalização. Uma das principais lacunas é a restrição de domínio: UGCNormal foi projetado com foco em *reviews* de produtos, desconsiderando as demais modalidades de CGU. Outro ponto a ser melhorado é que as técnicas de normalização empregadas são baseadas em léxico. Ou seja, a identificação e a correção de ruídos é efetuada com base em dicionários criados manualmente. Essa característica é altamente limitante, visto que o uso de dicionários não é escalável e depende totalmente do domínio. Pautando-se principalmente nesses pontos limitantes, a proposta dos métodos de normalização que serão explorados nesse trabalho visa a atender aos seguintes princípios:

- **Independência de domínio:** o processo de normalização é apto a lidar com diversos tipos de CGU, como *reviews* de produtos, publicações em redes sociais e discussões em fóruns. Para tanto, serão empregadas técnicas capazes de aprender automaticamente a correção dos ruídos encontrados em diferentes tipos de textos – eliminando a necessidade de utilizar léxicos elaborados manualmente.
- **Flexibilidade e adaptabilidade:** as definições de o que são palavras ruidosas e de quando normalizá-las dependem da aplicação. Assim, o normalizador não determina por si só a normalização. Na prática, a identificação de ruídos pode ser baseada tanto em léxicos que podem ser alterados conforme a necessidade da aplicação quanto em algoritmos de aprendizado de máquina que podem ser treinados para modalidades textuais específicas. O léxico da língua padrão pode ser considerado como base, mas não há necessidade de segui-lo estritamente. Uma vez identificadas as palavras fora do padrão, o sistema fará sugestões de palavras correspondentes para normalizá-las. A aplicação deve ter a possibilidade de determinar se as sugestões serão ou não acatadas, fazendo com que o normalizador não atue de forma determinística. É válido ressaltar que, embora léxicos sejam empregados na identificação de ruídos, a correção ocorre automaticamente em decorrência de algoritmos de aprendizado, portanto as deficiências do UGCNormal não serão repetidas.
- **Capacidade de lidar com diferentes tipos de ruído:** além de lidar com diversas categorias de CGU, o processo de normalização deve ser apto a tratar diferentes tipos de ruído – incluindo erros ortográficos contextuais ou não, gírias da *web* e abreviações. Para tanto, é ainda mais relevante descartar o uso de léxicos para correção, de modo a evitar categorização de erros. Para garantir a capacidade de lidar genericamente com diversos ruídos, é necessário que a correção ocorra indiscriminadamente, sem considerar categorias específicas.

Além desses princípios, é necessário deixar claro quais tipos de ruídos serão considerados. Todas as categorias presentes no corpus TUB-UGC, descritas detalhadamente na [Seção 5.3](#), serão consideradas de alguma maneira pelos métodos de normalização. No entanto, é importante ressaltar que o significado de *normalizar* pode ser diferente para cada categoria de ruído. Desse modo, ruídos de determinados tipos não precisam necessariamente passar pelas três etapas de normalização. Com o intuito de deixar claro todo o processo de normalização, o tratamento específico aplicado a cada categoria de ruído será descrito.

As categorias de *Erros Ortográficos*, *Abreviações* e *Internetês* devem passar pelas três etapas. Métodos para normalizar ruídos pertencentes a essas categorias seguem os mesmos princípios, visto que devem inicialmente identificar a palavra fora do padrão, gerar possíveis correções para ela e, por fim, escolher o melhor substituto. Portanto, cada um desses passos pode ser avaliado individualmente ao tratar esses tipos de ruído. Erros ortográficos contextuais devem passar por um processo similar, no entanto, são processados por técnicas distintas – visto que sua normalização depende exclusivamente de informações do contexto.

Siglas e *Nomes Próprios* representam ruídos que somente precisam ser identificados. Sua correção é, na prática, apenas estética, pois só há mudança de capitalização – *samsung* deve ser normalizado para *Samsung* e *hdmi* para *HDMI*, por exemplo. Esse tipo de correção é útil para ferramentas de PLN que necessitam de capitalização consistente. Neste trabalho, optou-se por não expandir siglas automaticamente, devido principalmente ao fato de que acrônimos e siglas também são comumente encontrados em textos na forma padrão. *Organização das Nações Unidas*, por exemplo, é frequentemente referida como ONU, sendo que a forma abreviada é considerada padrão. Desse modo, é difícil categorizá-los como ruídos que efetivamente necessitam de correção.

Ruídos dos tipos *Estrangeirismo* e *Já Corrigido* também devem ser apenas identificados. Não é necessário efetuar nenhuma outra ação para normalizá-los, visto que já se encontram em formas canônicas. No caso dos estrangeirismos, as técnicas desenvolvidas permitem removê-los do texto caso isso seja adequado às necessidades da aplicação. Palavras marcadas como *lixo* são identificadas e, por padrão, removidas, pois são em geral *tokens* que não podem ser transformados em palavras reconhecidas pela língua padrão. Por fim, ruídos *Sem Categoria* são identificados como um ruído qualquer e passam pelas três etapas de normalização. Como palavras desse tipo não se enquadram em nenhuma das categorias bem definidas, não é possível processá-las de maneira específica.

Resumidamente, o processo de normalização pode ser definido de acordo com os seguintes passos:

1. A amostra de CGU a ser normalizada é pré-processada de modo a deixá-la adequada

para ser tratada por técnicas posteriores. O pré-processamento inclui tokenização, segmentação de sentenças, remoção de repetição excessiva de caracteres, entre outros procedimentos. A remoção de caracteres repetidos tem como objetivo facilitar o processamento da palavra pelos métodos de normalização. Em geral, sequência são limitadas a três repetições – portanto *muuuuuuito* seria transformado em *muuuito*. Para realizar a segmentação de sentenças, utilizam-se ferramentas já disponíveis, visto que a tarefa por si só é altamente complexa e está além do escopo deste trabalho.

2. Uma vez delimitadas as sentenças, palavras fora do padrão são identificadas.
3. Nesse momento, Nomes Próprios e Siglas podem ser capitalizados. *Tokens* categorizados como Lixo são removidos. Estrangeirismos e palavras já corrigidas não são consideradas nas etapas posteriores, ou seja, seu processamento acaba na identificação.
4. Para as demais categorias, são gerados candidatos para corrigir cada palavra fora do padrão nas sentenças.
5. Após a geração de candidatos, cada um é avaliado e ranqueado levando-se em consideração o método empregado. Em seguida, o melhor candidato segundo as técnicas de ranqueamento propostas substitui a palavra fora do padrão. A lista de candidatos poderia ainda passar por ranqueamentos posteriores de acordo com a necessidade da aplicação.
6. Por fim, após normalizar todas as sentenças, o novo texto é gerado.

Os passos do processo de normalização, descritos anteriormente, estão contidos nos métodos que serão apresentados nas seções seguintes.

6.2 Métodos de Identificação de Ruídos

O primeiro passo efetivo do processo de normalização consiste em determinar o que será normalizado – ou seja, identificar se determinada palavra está escrita na forma padrão ou se é um ruído. Para tanto, primeiramente deve-se estabelecer o que é a forma padrão. Em princípio, a norma culta da língua é adotada como o padrão. Nesse caso, é possível utilizar um léxico da língua para identificar palavras ruidosas. Desse modo, se uma palavra não estiver contida no léxico, ela é considerada um ruído. No entanto, essa abordagem é limitada. O principal ponto negativo é que um léxico - por melhor que seja – não compreende todas as palavras que devem ser reconhecidas. É o caso das classes abertas, como nomes próprios, siglas, gírias, etc. Uma maneira de contornar esse problema é adotar um conjunto de léxicos para ditar o padrão, ao invés de apenas um. Ou seja, além

do léxico da língua, léxicos de nomes próprios, de siglas, entre outros, são combinados de modo a identificar ruídos. A combinação de léxicos também permite flexibilizar a definição de o que é padrão de acordo com as necessidades de aplicações específicas. Para alterar quais palavras são consideradas válidas, basta adicioná-las a algum léxico.

No entanto, a abordagem baseada em léxicos é totalmente dependente das listas de palavras. Embora dicionários não sejam recursos caros ou de difícil acesso, ainda assim aumentam os requisitos do processo de normalização. Além disso, manter dicionários que identifiquem corretamente ruídos em diferentes domínios e em diferentes textos de CGU é difícil. Nesse contexto, é relevante investigar meios de identificar ruídos sem necessidade de recursos. Métodos de aprendizado de máquina têm a capacidade de realizar esse tipo de tarefa. Uma alternativa é treinar algoritmos visando a identificar automaticamente palavras fora do padrão. A vantagem dessa abordagem é que, uma vez realizado o treinamento, não se necessita mais de recursos externos. Em contrapartida, não é possível alterar o que é considerado padrão após o treinamento – diminuindo assim a flexibilidade da identificação. Considerando que as duas abordagens (por meio de léxicos ou automática) possuem vantagens e desvantagens, ambas são investigadas neste trabalho, sendo apresentadas nas próximas seções.

Identificação de Ruídos Baseada em Léxicos

Trata-se da abordagem mais simples para identificação de ruídos. Para cada palavra da sentença, basta verificar se ela está contida em algum dos léxicos. Caso não esteja, a palavra é possivelmente um ruído e deve ser analisada pelos métodos de normalização. Como ponto de partida, os léxicos da ferramenta UGCNormal foram utilizados. Ao todo, seis listas de palavras foram usadas, contendo: nomes próprios, siglas, abreviações, internetês, estrangeirismos e um dicionário da língua portuguesa. Essas listas foram empregadas na pré-anotação do corpus TUB-UGC, conforme descrito na [Seção 5.3](#). Após a anotação, qualquer palavra identificada como ruído por um anotador foi adicionada ao léxico correspondente de sua categoria, se não tivesse sido identificada previamente pelo UGCNormal. Por exemplo: se o nome próprio *Maria* tivesse sido pré-categorizado pelos léxicos do UGCNormal, não era necessário acrescentá-lo ao léxico de nomes próprios. Já se a sigla *BBB* não tivesse sido identificada, ela era então expandida pelo anotador para *Big Brother Brasil* e então inserida no léxico de siglas. Consequentemente, ao fim da anotação todos os léxicos foram atualizados e expandidos. Desse modo, é difícil avaliar a identificação por meio de léxicos, pois todos os ruídos do corpus estão contidos nos léxicos, então a taxa de identificação correta seria de 100% – visto que os ruídos foram extraídos a partir do próprio corpus. No entanto, uma análise interessante que pode fornecer informações importantes pode ser feita ao comparar os léxicos do UGCNormal aos expandidos ao fim da anotação. A [Tabela 15](#) apresenta o número de palavras em cada um dos léxicos

do UGCNormal, a porcentagem de ruídos (incluindo repetições) no corpus TUB-UGC que cada léxico foi capaz de pré-categorizar e, por fim, o total de palavras adicionadas a cada um – ou seja, o total de palavras únicas que foram encontradas no corpus e que não estavam nos léxicos da categoria correspondente.

Tabela 15 – Avaliação dos léxicos utilizados na identificação de ruídos.

Léxico	Número de Palavras	Cobertura do Corpus	Palavras Adicionadas
Abreviações	89	68,54%	206
Siglas	158	28,22%	123
Nomes Próprios	9.405	49,27%	775
Internetês	401	34,27%	371
Estrangeirismos	247	27,30%	217

Fonte: Elaborada pelo autor.

Pode-se notar que, em geral, a cobertura alcançada pelos léxicos do UGCNormal foi baixa. Como exceção de abreviações, categoria na qual um número pequeno de ruídos (*q*, *vc*, *tbn*) se repete muitas vezes, os léxicos não conseguiram pré-categorizar corretamente nem metade dos ruídos. Isso se deve principalmente à diferença de domínios, pois os léxicos foram originalmente projetados para tratar apenas *reviews* de produtos. Outro ponto interessante é que a quantidade de palavras adicionadas permite observar o enriquecimento dos léxicos. Em quatro dos cinco léxicos foram adicionadas quase o dobro de palavras em relação ao tamanho original. Além da quantidade, os ruídos inseridos são de domínios diferentes, trazendo mais variedade de palavras fora do padrão e, consequentemente, informações mais genéricas com menos dependência de domínio.

Identificação de Ruídos Baseada em Aprendizado de Máquina

Para identificar ruídos utilizando aprendizado de máquina, é necessário primeiramente modelar o problema antes de considerar quais métodos empregar. Considerando que o objetivo da identificação é atribuir uma categoria para cada palavra de uma sentença, este problema pode ser modelado como uma tarefa de predição estruturada. Dada um sentença, o algoritmo deve ser capaz de gerar uma sequência de mesmo tamanho, contendo uma *tag* para cada palavra correspondente. Não é necessário, entretanto, utilizar todas as categorias de ruídos. Como a identificação visa a deixar explícito como tratar cada palavra ruidosa, um novo conjunto de *tags* foi proposto, sendo elas:

- **OK**: indica que não é necessário aplicar nenhuma ação de normalização na palavra. Se enquadram nessa categoria palavras pertencentes ao léxico, anotadas como já corrigidas e estrangeirismos.

- **NP**: indica que a palavra é um nome próprio e deve ter a letra inicial capitalizada.
- **SI**: indica que a palavra é uma sigla e deve ter todas as letras capitalizadas.
- **LIXO**: *tokens* que não acrescentam informação à sentença e devem ser removidos.
- **NORM**: ruídos que podem ser erros ortográficos, abreviações ou internetês e que devem passar por todas as etapas do processo de normalização.

Com estas cinco categorias, é possível mapear todo tipo de ação a ser feito por um normalizador. Optou-se por utilizar redes neurais como método de aprendizado de máquina para aplicar à tarefa. Como apresentado na [Subseção 3.2.3](#), esse tipo de abordagem vem sendo utilizada em diversas tarefas de PLN e é capaz de obter resultados altamente satisfatórios. Desse modo, a identificação de ruídos foi modelada como um problema de aprendizado supervisionado. Portanto, são necessários dados anotados para efetuar o treinamento das redes neurais. Considerando que o número de sentenças anotadas não é suficiente para treinar algoritmos de aprendizado de máquina, optou-se por adotar uma abordagem de supervisão distante, inspirada no trabalho de [Go, Bhayani e Huang \(2009\)](#). Essa abordagem consiste basicamente em atribuir classes aos dados sem efetivamente haver anotação. No caso desta tarefa, os léxicos de normalização descritos na [Seção 6.2](#) foram utilizados para anotar uma amostra de 1 milhão de *tweets* **não** contidos no corpus TUB-UGC. Para cada sentença, todas as palavras receberam uma *tag* de acordo com o léxico que as continha. Palavras pertencentes ao léxico da língua, ao de *já corrigidas* e ao de estrangeirismos foram mapeadas para *OK*. Palavras contidas no léxicos de nomes próprios e no de siglas foram mapeadas, respectivamente, para *NP* e para *SI*. O léxico de *lixo* foi utilizado para atribuir a etiqueta *LIXO*. O restante das palavras foi marcada como *NORM* – indicando que deveriam ser processadas por todas as demais etapas de normalização. A supervisão distante permite gerar de maneira pouco custosa um grande número de sentenças anotadas. Isso permite o uso de redes neurais profundas para a realização desta tarefa.

A arquitetura escolhida para efetuar a identificação de ruídos foi uma rede neural recorrente convolucional (RCNN, do inglês *Recurrent Convolutional Neural Network*). Mais detalhes sobre seu funcionamento podem ser encontrados na [Subseção 3.2.3](#). Esse tipo de arquitetura já foi utilizado com sucesso para segmentação de sentenças ([TILK; ALUMÃE, 2016](#); [CHE et al., 2016](#)) e também para classificação de textos em geral ([LAI et al., 2015](#)), inclusive em trabalhos para o português brasileiro, como em ([TREVISIO; SHULBY; ALUÍSIO, 2017](#); [CORREA et al., 2017](#)). A arquitetura da RCNN utilizada, baseada no trabalho de [Treviso, Shulby e Aluísio \(2017\)](#), é ilustrada na [Figura 25](#).

A entrada do modelo é uma matriz de $\phi \times d$, sendo ϕ o tamanho da sentença e d , o número de dimensões das *embeddings*. O modelo Word2Vec com 500 dimensões foi

Figura 25 – Arquitetura da RCNN para identificação de ruídos.

Fonte: Treviso, Shulby e Aluísio (2017).

utilizado, sendo que o processo de indução das *word embeddings* usadas é descrito detalhadamente na Seção 6.3. Em seguida, a camada convolucional recebe a entrada e é responsável por extrair n_f novas *features* a partir de uma janela deslizante de tamanho h_c , que corresponde ao tamanho do filtro aplicado ao vetor concatenado $[e_1, \dots, e_{h_c}]$, que indica uma região de h_c palavras vizinhas (KIM, 2014). A camada convolucional produz novas *features* u_t para cada t -ésima palavra ao aplicar um filtro compartilhado para uma janela de h_c *embeddings* numa sentença de tamanho ϕ . A camada de convolução é capaz de combinar a representação das *embeddings* de modo a gerar novas *features* que trazem informações a respeito do contexto local. Com o intuito de passar para a próxima camada somente as *features* mais relevantes, uma operação de *max-pooling* é aplicada. Essa operação simplesmente seleciona os n maiores valores de cada região dos filtros.

As novas *features* extraídas são passadas por uma camada recorrente bidirecional. Como mencionado na Subseção 3.2.3, esse tipo de camada é capaz de armazenar informações passadas ao conectar estados ocultos anteriores com o estado oculto atual em um instante t , e também informações futuras ao efetuar o mesmo processo com a sentença invertida. Com o uso de uma camada bidirecional que trata *features* obtidas por meio de convolução, a arquitetura neural fica apta a explorar o princípio de que informações de palavras próximas têm grande influência, enquanto considera que palavras longes à esquerda (o passado da dimensão temporal) e à direita (o futuro da dimensão temporal) também podem ter algum impacto na classificação.

Após a camada BRNN, a estratégia de regularização conhecida como *dropout*, que tenta prevenir co-adaptação entre unidades ocultas durante o *forward-backpropagation*, é utilizada. Essa estratégia, proposta por Srivastava *et al.* (2014), basicamente ignora neurônios com o propósito de reduzir a chance de superajustar o modelo, ou seja, a chance de haver *overfitting*. Por fim, a última camada recebe a saída da BRNN após o *dropout* a cada instante t , e então a passa por uma camada totalmente conectada (MLP), de modo a produzir predições usando a função de ativação *softmax*. O treinamento foi realizado em

um conjunto de 1 milhão de *tweets* anotados por meio de supervisão distante. Os hiper-parâmetros utilizados no treinamento foram definidos seguindo o trabalho de Treviso, Shulby e Aluísio (2017), sendo que os principais estão listados na Tabela 16.

Tabela 16 – Principais hiper-parâmetros de treinamento da arquitetura RCNN.

Hiper-parâmetro	Valor
Tamanho das <i>embeddings</i>	500
Número de filtros	100
Tamanho do filtro	7
Tamanho do <i>max-pooling</i>	3
Unidades recorrentes	100
Taxa de <i>dropout</i>	0,5
Taxa de aprendizado	0,001

Fonte: Adaptada de Treviso, Shulby e Aluísio (2017).

A avaliação da arquitetura pode ser realizada de duas maneiras. Primeiramente, é possível analisar seu desempenho no próprio conjunto de treino, utilizando a abordagem de validação cruzada (KOHAVI *et al.*, 1995). Para tanto, os dados são divididos em um determinado número n de *folds*, treinado utilizando-se $n - 1$ deles e testado em apenas um. Isso se repete de modo a que a arquitetura seja treinada e testada em todos os *folds*. Por fim, a média do resultado obtido em cada um fornece uma avaliação mais precisa. Cinco *folds* foram usados nos experimentos conduzidos. Utilizar essa abordagem permite verificar a capacidade da arquitetura em aprender a desempenhar corretamente a tarefa. Se os resultados são ruins na validação cruzada nos dados de treinamento, há indícios de que o modelo utilizado não é adequado para a tarefa e, conseqüentemente, não se sairá bem ao classificar os dados de teste – que é, de fato, seu objetivo. A Tabela 17 apresenta as classificações realizadas pela arquitetura nos dados de treino utilizando validação cruzada. As linhas e as colunas representam as possíveis *tags*. Assim, a linha indica qual a *tag* correta, seguida pelo número de predições para cada categoria nas colunas. Por exemplo: a célula correspondente a (NP, NP) é o número de vezes em que a *tag* foi corretamente predita. Já a célula (NP, SI) indica quantas vezes NP foi erroneamente predita como SI.

Calculando-se a média da precisão por categoria, obtém-se precisão total de **99,81%**. Embora o valor seja extremamente alto, esse resultado não é totalmente inesperado. Considerando que a anotação foi feita por meio de léxicos, há pouca ambigüidade entre *tags* – em geral, cada palavra é sempre mapeada para a mesma categoria. Desse modo, a identificação dos ruídos torna-se simples para uma rede neural aprender. Apesar disso, é possível notar que a arquitetura não aprendeu somente a mapear os léxicos, pois ela efetuou predições incorretas. Essa característica indica que o modelo foi capaz de aprender informações que de fato demonstram o que faz uma palavra se tornar um ruído. Por fim, é fundamental avaliar o desempenho da arquitetura nos dados de teste – ou seja, no *corpus*

Tabela 17 – Predições realizadas pela RCNN no conjunto de treino.

<i>Tag</i> correta	<i>Tag</i> predita				
	OK	NORM	SI	LIXO	NP
OK	1882173	2167	0	3	87
NORM	947	212956	0	24	137
SI	6	35	9676	0	14
LIXO	14	260	0	25157	5
NP	59	432	0	0	453124

Fonte: Elaborada pelo autor.

TUB-UGC. Para tanto, ela foi treinada com a amostra total de *tweets* anotados e depois testada nos textos do *corpus*. É importante ressaltar que o *corpus* contém sentenças de diferentes modalidades de CGU, não apenas *tweets*. Portanto, a identificação de ruídos nesse caso tem o potencial de ser desafiadora devido à presença de tipos de textos que não foram vistos durante o treinamento. A Tabela 18 apresenta as classificações realizadas pela arquitetura no conjunto de textos do *corpus* TUB-UGC.

Tabela 18 – Predições realizadas pela RCNN no conjunto de teste.

<i>Tag</i> correta	<i>Tag</i> predita				
	OK	NORM	SI	LIXO	NP
OK	38476	634	34	18	282
NORM	171	4955	61	100	99
SI	11	96	427	0	8
LIXO	29	438	24	172	22
NP	52	811	75	4	1970

Fonte: Elaborada pelo autor.

A precisão média obtida foi de **93,64%**. O valor ainda se manteve alto, porém é possível notar que a quantidade de erros foi bem maior. Na categoria *lixo*, a arquitetura não foi capaz de classificar corretamente a maioria dos ruídos. É provável que no *corpus* estejam contidos muitos *tokens* considerados como lixo, que não estavam presente no conjunto de treino. Apesar disso, pode-se concluir que métodos de aprendizado de máquina são capazes de identificar ruídos com precisão comparável à obtida por meio de léxicos. Uma análise mais completa pode ser feita ao testar o modelo e os léxicos de ruídos em amostras de textos não vistos anteriormente. Nesse caso, poderia haver palavras não inseridas nos léxicos, permitindo assim avaliá-los efetivamente.

De modo geral, a identificação de ruídos baseada em aprendizado de máquina

é efetiva. Entretanto, na prática, é relevante discutir se há vantagens em utilizar tal abordagem. Apesar de não necessitar de léxicos após o treinamento, o uso de arquiteturas neurais é computacionalmente custoso e lento. Consultar uma lista de palavras é bem mais rápido e simples do que efetuar predições em uma rede neural recorrente. Outro ponto negativo é a falta de flexibilidade. Para alterar o que é definido como ruído na abordagem baseada em léxicos, basta modificar a lista de palavras – removendo ou adicionando novos termos de acordo com a necessidade. Já em uma rede neural, realizar alterações implica em treinar novamente a arquitetura por completo, que também é um processo custoso. Assim, embora o uso de arquiteturas neurais para identificação de ruídos seja interessante sob o ponto de vista de pesquisa, na prática, é mais viável utilizar léxicos.

6.3 Métodos de Geração de Candidatos

Uma vez identificadas as palavras ruidosas que devem ser normalizadas, é possível gerar candidatos para corrigi-las. Esse processo visa a reduzir a complexidade da escolha da palavra correta, que é realizada na etapa de ranqueamento. Embora seja possível efetuar toda a normalização em um só passo – identificando o ruído e corrigindo-o em seguida –, gerar candidatos para depois ranqueá-los possui algumas vantagens. Utilizar uma etapa intermediária flexibiliza a normalização, possibilitando empregar métodos diferentes para gerar e para ranquear possíveis correções. A geração de candidatos também reduz o número total de palavras a serem avaliadas pelo ranqueamento: em vez de ser necessário analisar todo o léxico buscando correções, basta processar os candidatos gerados.

De modo geral, a etapa de geração de candidatos deve atender a dois requisitos: ser capaz de conter a versão correta da palavra dentre os candidatos gerados e gerar um número limitado de palavras candidatas. O primeiro requisito é necessário para garantir que os candidatos gerados sejam de fato possíveis correções. Para avaliá-lo, basta verificar se a versão normalizada do ruído está contida no total de candidatos. O segundo requisito visa a assegurar que a geração de candidatos está efetivamente reduzindo o total de palavras a serem avaliadas posteriormente. Se o número de candidatos não fosse considerado, bastaria adicionar todo o léxico da língua como potencial correção e se obteria cobertura total. No entanto, essa abordagem vai contra os princípios de flexibilização da normalização. Uma maneira de avaliar os dois requisitos em conjunto é por meio do potencial de correção. Por exemplo: se um método *A* de geração obtém 80% de cobertura com um total de 100 candidatos, e outro método *B* obtém o mesmo valor com apenas 10, é intuitivo dizer que *B* é superior. Numericamente, essa relação pode ser expressa dividindo-se o valor da cobertura pelo total de candidatos – obtendo-se, assim, a probabilidade de cada candidato ser o correto. No caso do método *A*, esse valor é de $80\%/100 = 0,8$, indicando que cada candidato tem 0,8% de probabilidade de ser o correto. Já para *B*, o valor é de $80\%/10 = 8\%$, mostrando que os candidatos gerados por esse método têm maior chance

de estarem corretos. Essa medida, aqui denominada *potencial de correção*, será utilizada na avaliação dos métodos de geração de candidatos. A cobertura e o número de candidatos também serão avaliados.

Geração por Similaridade Lexical

Um método intuitivo de gerar candidatos é por meio de medidas de similaridade lexical. Essas medidas, apresentadas com mais detalhes na [Subseção 3.1.2](#), são capazes de representar e avaliar a similaridade entre palavras. Ou seja, o uso delas permite calcular numericamente a semelhança entre um ruído e sua possível correção. Utilizá-las para gerar candidatos para normalização é um processo simples. Primeiramente, deve-se calcular a similaridade entre o ruído a ser normalizado e todas as palavras do léxico. Em seguida, todas as palavras que contam com similaridade acima de um limiar definido são consideradas como possíveis correções, sendo por fim adicionadas à lista de candidatos. Neste trabalho, foram desenvolvidos métodos utilizando três medidas diferentes: distância de edição, *longest common subsequence* (LCS) (ULLMAN; AHO; HIRSCHBERG, 1976) e uma adaptação da métrica de similaridade lexical, proposta por Hassan e Menezes (2013), aqui denominada *simlex*.

Para gerar candidatos utilizando a distância de edição, três valores de limiar foram utilizados: 1, 2 e 3, cada um representando o número máximo de edições necessárias para transformar o ruído em uma possível correção. A [Tabela 19](#) apresenta o valor de cobertura (em porcentagem) obtido para cada categoria de ruído. A [Tabela 20](#) apresenta o número médio de candidatos gerados pelo método e seu potencial de correção (em porcentagem).

Tabela 19 – Cobertura da geração de candidatos por distância de edição.

	Distância Máxima		
	1	2	3
Erro Ortográfico	79,69	88,17	92,84
Abreviação	2,56	52,75	74,59
Internetês	10,23	28,40	43,20

Fonte: Elaborada pelo autor.

É possível notar que quanto maior o valor do limiar, maior o número de candidatos e consequentemente melhor a cobertura. No entanto, o potencial de correção cai drasticamente de acordo com o total de candidatos. Quanto menor esse valor, maior o número de palavras adicionadas indevidamente como candidatas – indicando que o ranqueamento será mais difícil. A distância de edição teve bons resultados para correção de erros ortográficos, os quais em geral são pequenas alterações de caracteres. Porém, para as demais categorias, a cobertura obtida foi menor. Para abreviações, isso se deve principalmente

Tabela 20 – Número médio de candidatos e potencial de correção para distância de edição.

	Distância Máxima					
	1		2		3	
	Candidatos	Potencial	Candidatos	Potencial	Candidatos	Potencial
Erro Ortográfico	8,15	9,78	165,90	0,53	1449,55	0,06
Abreviação	6,49	0,39	138,29	0,38	1123,62	0,07
Internetês	4,52	2,26	99,55	0,28	968,93	0,04

Fonte: Elaborada pelo autor.

ao fato de que a versão abreviada de uma palavra tende a ter bem menos letras que a original, aumentando assim a distância de edição. Para o internetês, a diversidade dos ruídos presentes nessa categoria dificulta alinhá-los à versão correta por meio de apenas similaridade lexical.

Diferentemente da distância de edição, a LCS indica o tamanho da maior subsequência em comum entre duas cadeias de caracteres. Desse modo, essa medida é capaz de representar quanto uma palavra está contida em outra. Com base nessa propriedade, a escolha de utilizar LCS foi pautada principalmente em melhorar a cobertura de candidatos para normalizar abreviações. É importante ressaltar que a LCS depende do tamanho de cada palavra, pois mede subsequências em comum. Para eliminar essa dependência, foi utilizada a razão entre a LCS e o tamanho da maior palavra – normalizando assim seu valor entre 0 e 1. Outra alteração realizada no cálculo foi o uso do complemento da medida em vez da medida em si, apenas visando a manter a consistência na apresentação de resultados. Enquanto uma distância de edição 0 indica que duas palavras são idênticas, LCS 0 indica que não há semelhança entre elas, pois quanto maior o valor, maior a similaridade. Portanto, o complemento foi utilizado para inverter o funcionamento da medida. Como limiar, foram utilizados os valores 0.2, 0.5 e 0.8. A [Tabela 21](#) apresenta o valor de cobertura (em porcentagem) obtido para cada categoria de ruído. A [Tabela 22](#) apresenta o número médio de candidatos gerado pelo método e seu potencial de correção (em porcentagem).

A cobertura obtida utilizando LCS com o limiar máximo foi superior à distância de edição em todas as categorias. No entanto, o número de candidatos gerados foi extremamente elevado – reduzindo muito o potencial de correção. Apesar disso, o total de candidatos ainda foi consideravelmente menor que o tamanho total do léxico, indicando que cumpriu seu papel como etapa intermediária visando a facilitar o ranqueamento posterior. A última medida avaliada, *simlex*, é uma combinação da LCS com a distância de

Tabela 21 – Cobertura da geração de candidatos por LCS.

	Distância Máxima		
	0,2	0,5	0,8
Erro Ortográfico	46,41	88,62	93,56
Abreviação	0,39	30,01	95,67
Internetês	5,50	36,03	55,88

Fonte: Elaborada pelo autor.

Tabela 22 – Número médio de candidatos e potencial de correção para LCS.

	Distância Máxima					
	0,2		0,5		0,8	
	Candidatos	Potencial	Candidatos	Potencial	Candidatos	Potencial
Erro Ortográfico	4,06	0,11	2476,81	0,03	125299	0
Abreviação	0,10	3,9	84,60	0,35	13996	0,06
Internetês	1,42	3,86	1433,51	0,03	90302	0

Fonte: Elaborada pelo autor.

edição. Baseando-se em [Hassan e Menezes \(2013\)](#), a medida foi calculada como:

$$\text{simlex}(w_1, w_2) = \begin{cases} \frac{LCSR(w_1, w_2)}{MED(w_1, w_2)}, & \text{se } MED(w_1, w_2) > 0 \\ LCSR(w_1, w_2), & \text{senão} \end{cases} \quad (6.1)$$

onde w_1 e w_2 são as palavras as quais se deseja calcular a similaridade e $MED(w_1, w_2)$ é a distância de edição modificada entre elas, calculada como

$$MED(w_1, w_2) = ED(w_1, w_2) - DS(w_1, w_2)$$

$DS(w_1, w_2)$ representa a *simetria diacrítica* entre w_1 e w_2 – uma medida que indica o número de caracteres de uma palavra alinhados com versões deles mesmos que contenham diacríticos (ou seja, acentos) na outra palavra. A simetria diacrítica entre *maçã* e *maca*, por exemplo, é 2, pois *ç* está alinhado com *c* e *ã* com *a*. Essa modificação é utilizada devido ao fato de que muitos erros ortográficos em português ocorrem em decorrência de um único diacrítico errado, portanto esse cálculo dá mais peso a esse tipo de ruído. *LCSR* refere-se a *Longest Common Subsequence Ratio*, ou taxa da maior subsequência em comum, sendo a normalização da LCS pelo tamanho da maior palavra, como mencionado. A [Tabela 23](#) apresenta o valor de cobertura (em porcentagem) obtido para cada categoria de ruído. A [Tabela 24](#) apresenta o número médio de candidatos gerado pelo método e seu potencial de correção (em porcentagem).

Tabela 23 – Cobertura da geração de candidatos por *simlex*.

	Distância Máxima		
	0,2	0,5	0,8
Erro Ortográfico	78,69	82,64	88,03
Abreviação	0,40	2,6	22,70
Internetês	5,03	13,89	31,60

Fonte: Elaborada pelo autor.

Tabela 24 – Número médio de candidatos e potencial de correção para *simlex*.

	Distância Máxima					
	0,2		0,5		0,8	
	Candidatos	Potencial	Candidatos	Potencial	Candidatos	Potencial
Erro Ortográfico	2,92	27,90	12,50	6,61	204,69	0,43
Abreviação	0,07	5,71	5,24	0,49	36,64	0,62
Internetês	0,87	5,78	5,49	2,53	97,87	0,32

Fonte: Elaborada pelo autor.

Com exceção de erros ortográficos, *simlex* obteve cobertura baixa. Entretanto, uma característica interessante sobre essa medida é o número pequeno de candidatos totais. Isso indica que a escolha dos candidatos foi conservadora – a medida é portanto mais restritiva. No contexto de geração de candidatos, essa propriedade não é totalmente desejável pois limita a cobertura. No entanto, pode ser muito útil para avaliação e ranqueamento, por conseguir atribuir uma similaridade mais criteriosa. De modo geral, os resultados indicam que gerar candidatos utilizando apenas medidas de similaridade lexical não é o bastante. Embora tenha sido possível obter cobertura e potencial de correção satisfatórios na categoria de erros ortográficos, para abreviações e internetês os resultados foram insuficientes. Desse modo, é relevante investigar outras técnicas para realizar a geração de candidatos.

Geração por Word Embeddings

Considerando que os resultados obtidos por métodos baseados em similaridade lexical não foram satisfatórios, é necessário explorar outras abordagens. Nesse contexto, utilizar *word embeddings*, apresentadas com mais detalhes na [Subseção 3.1.1](#), é promissor devido a sua capacidade de representar informações semânticas, sintáticas e até mesmo contextuais. Sendo assim, a ideia fundamental deste método de geração de candidatos é utilizar o potencial das representações vetoriais de palavras e gerar léxicos de normalização aprendidos automaticamente de maneira não-supervisionada. O objetivo é que os léxicos

aprendidos contenham como entrada palavras ruidosas seguidas por listas de candidatos a normalizá-las. Este método foi inspirado no trabalho de Sridhard (2015), sendo que uma versão expandida, modificada e adaptada para o português foi inicialmente apresentada em Bertaglia e Nunes (2016). Ainda assim, o método foi revisado para melhor se adequar a este trabalho. Sua primeira versão será utilizada para ilustrar o método. Em seguida, as modificações feitas serão apresentadas e avaliadas no corpus TUB-UGC.

Para o funcionamento do método, foram treinados modelos Word2Vec (MIKOLOV *et al.*, 2013) de *embeddings*, construídos com base em textos ruidosos de conteúdo gerado por usuário. Foram utilizadas duas fontes de textos. A primeira é um conjunto de *reviews* de produtos extraídas do corpus proposto por Hartmann *et al.* (2014). A segunda é uma grande quantia de *tweets* escritos em português extraídos do Twitter no período de junho a outubro de 2016 utilizando um *crawler* desenvolvido pelos autores. Ambos conjuntos de textos tiveram de ser pré-processados de modo a prepará-los para processamentos posteriores. Os *tweets* foram tokenizados utilizando a ferramenta Twokenizer¹, um tokenizador projetado especificamente para lidar com publicações no Twitter. Nome de usuários, *hashtags* e *hyperlinks* foram removidos por não acrescentarem informações relevantes à criação de *embeddings*. As *reviews* de produtos foram tokenizadas utilizando um tokenizador simples de palavras (*word tokenizer*) e marcações pré-definidas contidas nas *reviews*, como “O que gostei sobre esse produto:”, foram removidas. Em seguida, as sentenças foram segmentadas utilizando a ferramenta proposta por Condori e Pardo (2015). Após a segmentação, o total de sentenças obtidas foi de 6.8 milhões para os *tweets* e 20 milhões para as *reviews*. É importante ressaltar que o elevado número de sentenças de *reviews* se deve ao modo de funcionamento do segmentador. Por utilizar *features* como quebra de linha, muitas vezes palavras individuais são consideradas como sentenças – fato que acarreta um aumento considerável da quantidade total obtida.

Após o pré-processamento dos textos, as *embeddings* foram treinadas utilizando a ferramenta Gensim². Foram criados modelos *skip-gram* (Skip) e *continuous bag-of-words* (Cbow) com 100, 300 e 500 dimensões. Uma janela de contexto de tamanho 5 foi utilizada e o processo de amostragem hierárquica foi empregado durante o treinamento para reduzir o tamanho do vocabulário, de modo que somente palavras com mais de dez ocorrências no corpus foram consideradas.

Com os modelos de *embeddings* criados, é possível aplicar medidas de similaridade para encontrar candidatos para normalizar palavras fora do padrão. A primeira medida utilizada foi a similaridade do cosseno – métrica que calcula o cosseno do ângulo formado entre dois vetores de modo a determinar quão similares são (MIKOLOV; YIH; ZWEIG, 2013). A Figura 26 ilustra os 25 vizinhos mais próximos (com base na similaridade do

¹ <cs.cmu.edu/~ark/TweetNLP/#pos>

² <radimrehurek.com/gensim/index.html>

cosseno) da palavra *você* no modelo de *embeddings* obtido.

Figura 26 – 25 vizinhos mais próximos da palavra *você* no modelo de *embeddings* Word2Vec.

Fonte: Elaborada pelo autor.

A versão canônica da palavra se encontra no centro da [Figura 26](#). Ao seu redor, há diversas variações ruidosas encontradas no modelo de *embedding*. O objetivo é encontrar a forma canônica partindo de uma versão ruidosa. Para tanto, é necessário considerar a similaridade lexical entre as palavras. Essa medida foi computada com base na *simlex*, apresentada na [Seção 6.3](#).

Além do conjunto de *word embeddings*, o método também necessita da lista de palavras que são considerada canônicas pela aplicação. Nos experimentos realizados, o léxico da língua portuguesa UNITEX-PB³ foi utilizado. Palavras com frequência menor de 100 no Corpus Brasileiro⁴ foram removidas. Essa etapa de poda remove palavras que têm probabilidade ínfima de estarem presentes em textos de CGU e consequentemente diminui o tamanho do léxico – reduzindo assim consideravelmente o tempo de execução do método.

A etapa seguinte consiste em encontrar os K vizinhos mais próximos no espaço vetorial das *embeddings* de cada palavra canônica, sem considerar na vizinhança palavras contidas no léxico. O valor de K foi definido como 25, seguindo os experimentos de Sridhard (2015). Esse passo gera uma lista de possíveis corrupções (ou seja, versões com ruído) de

³ nilc.icmc.usp.br/nilc/projects/unitex-pb/web/dicionarios.html

⁴ <corpusbrasileiro.pucsp.br/cb/Acesso.html>

cada palavra canônica w_c . Considerando o espaço vetorial das *embeddings* V , o léxico de palavras padrão L e w_c , cada palavra $\in L$, a lista de versões ruidosas é obtida como:

$$\text{versões ruidosas}[w_c] = \forall_{w \in V, w \notin L} \max_k \text{similaridade cosseno}(w_c, w) \quad (6.2)$$

Após a obtenção da lista de versões ruidosas, é necessário invertê-la de modo que seja possível encontrar a palavra canônica dada uma ruidosa – ou seja, a lista deve ser indexada pela palavra com ruído e deve conter possíveis candidatos de correção. Para efetuar esse procedimento, a lista de versões ruidosas de cada palavra do léxico é percorrida. Em seguida, uma nova estrutura é criada para armazenar cada palavra ruidosa w_n como índice de uma lista de palavras canônicas que contém w_n em sua lista de versões ruidosas. Após esse passo, uma tupla na forma $(w_c, \text{score}(w_n, w_c))$ é armazenada como possível candidato de correção a w_n . A pontuação (*score*) é calculada como:

$$\text{score}(w_n, w_c) = n \times \text{simlex}(w_n, w_c) + (1 - n) \times \text{similaridade cosseno}(w_n, w_c) \quad (6.3)$$

A similaridade no espaço vetorial (calculada pelo cosseno) é considerada na pontuação devido ao fato de que algumas correções possuem baixa similaridade lexical. Isso ocorre comumente em abreviações: *d+*, por exemplo, é frequentemente utilizada para abreviar *demais*. Como a subsequência comum de caracteres entre as duas é pequena, o valor da similaridade lexical é baixo. Considerar a similaridade do cosseno ameniza esse problema. O termo n da equação é um valor entre 0 e 1 que indica o peso (ou importância) dado à similaridade lexical. Nos experimentos realizados, n foi definido como 0,8. O léxico aprendido é armazenado como estrutura *hash*, indexada por w_n , na qual cada entrada contém uma lista de prováveis candidatos a correção para substituir a respectiva palavra ruidosa. Para determinar o melhor candidato, basta computar $\max(\text{score}(w_n, w_c))$.

Considerando que somente os K vizinhos mais próximos da palavra canônica são adicionados ao léxico, algumas palavras ruidosas de baixa frequência podem não constar nos K vizinhos e conseqüentemente serem desconsideradas. Nesse caso, a abordagem proposta até o momento não é capaz de encontrar nenhum candidato de correção. Para contornar esse problema, uma etapa de expansão foi adicionada ao método. Se determinada palavra ruidosa w_n não está contida no léxico aprendido, ou seja, $w_n \notin T$, a palavra canônica $\in L$ mais lexicalmente similar a w_n é adicionada como candidato de correção. Desse modo, a etapa de expansão pode ser definida como:

$$\text{Se } w_n \notin T : T[w_n] \Leftarrow \max_{w_c \in L} \text{simlex}(w_n, w_c)$$

Apesar de ser simples, a etapa de expansão melhorou os resultados gerais obtidos pelo sistema. É importante notar que essa versão do método utilizava um modelo de língua para tratar erros ortográficos contextuais. Na versão atual, essa abordagem não é mais utilizada por ser pouco efetiva. Métodos específicos para tratar erros contextuais foram

desenvolvidos e serão descritos na [Seção 6.4](#). No trabalho de [Bertaglia e Nunes \(2016\)](#), o método foi avaliado no corpus UGCNormal. Os primeiros experimentos conduzidos tiveram como objetivo determinar qual o melhor modelo de *word embedding* para a tarefa de normalização. Foram avaliados os modelos *skip-gram* e *continuous bag-of-words* (Cbow) do Word2Vec com 100, 300 e 500 dimensões. Para cada configuração de dimensão, os modelos foram treinados em dois conjuntos diferentes de dados: o primeiro, denominado Ruidoso, segue exatamente o padrão já descrito. O segundo, denominado Híbrido, inclui uma amostra adicional de 38 milhões de sentenças extraídas da Wikipédia. A [Tabela 25](#) apresenta a cobertura obtida utilizando cada modelo de *embeddings* na correção de erros ortográficos (O) e na de internetês (I) (com a configuração básica do método).

Tabela 25 – Resultados para correção de ruídos obtidos de cada modelo Word2Vec.

Modelo	Dimensões					
	100		300		500	
	O	I	O	I	O	I
Cbow Híbrido	67.2%	64.5%	72.3%	64.5%	72.1%	64.5%
Cbow Ruidoso	70.7%	54.8%	74.7%	64.5%	76.0%	64.5%
Skip-gram Híbrido	63.8%	77.4%	74.2%	71.0%	76.0%	71.0%
Skip-gram Ruidoso	77.3%	64.5%	78.6%	67.7%	78.6%	77.4%

Fonte: Elaborada pelo autor.

A abordagem apresentada é não-supervisionada e faz uso de medidas de similaridade lexical e de modelos vetoriais de representação de palavras. O método não necessita de conjuntos de regras nem de léxicos específicos, sendo apenas requerida uma lista de palavras consideradas canônicas. Portanto, a abordagem é facilmente expansível e adaptável conforme as necessidades de cada aplicação específica. Com base nessas características, optou-se por continuar utilizando-a neste trabalho. A principal diferença na versão atual é que a etapa de expansão foi incorporada ao método – ou seja, não é mais realizada em separado. Em vez disso, palavras lexicalmente similares são adicionadas como candidatas de correção ao mesmo tempo que as similares no espaço de *embeddings*. Outra diferença é no modelo de *embeddings* utilizado. Optou-se por investigar o uso de outros modelos além do Word2Vec, visando a analisar o impacto do espaço vetorial na geração de candidatos. Para tanto, os modelos FastText ([BOJANOWSKI et al., 2016](#)) e Wang2Vec ([LING et al., 2015](#)), ambos descritos na [Subseção 3.1.1](#), foram avaliados. A modelagem do Word2Vec prioriza a captura de informações semânticas, a do FastText, de morfológicas e a do Wang2Vec, de sintáticas. Para ilustrar a diferença entre os modelos, a [Tabela 26](#) apresenta as palavras mais próximas à palavra ruidosa *simplismente* (simplesmente) no espaço vetorial, dispostas em ordem decrescente de acordo com a similaridade.

É possível notar que a versão correta da palavra (*simplesmente*), encontra-se entre

Tabela 26 – Palavras mais próximas à *simplesmente* em diferentes modelos de *embeddings*.

Word2Vec	Wang2Vec	FastText
<u>simplesmente</u>	<u>simplesmente</u>	simplsmente
nouis	realmente	<u>simplesmente</u>
jungsis	literalmente	simplis
personifica	definitivamente	felismente
estou	realmnt	simplesment

Fonte: Elaborada pelo autor.

as mais próximas em todos os modelos. No Word2Vec, as palavras mais próximas não possuem relação óbvia com a original, sendo que provavelmente obtiveram representações similares por ocorrerem em contextos semelhantes. Já no modelo Wang2Vec, é clara a captura de informações sintáticas: todas as palavras próximas são advérbios. Analogamente, o FastText demonstra forte indícios morfológicos, gerando palavras próximas com muitos caracteres em comum. Intuitivamente, pode-se perceber as nuances de cada representação. Desse modo, é relevante considerar todas neste trabalho.

Para treinar as novas *embeddings*, uma amostra maior, contendo cerca de 40 milhões de *tweets*, foi utilizada. A tokenização foi um pouco diferente: números foram substituídos pela *tag* *NUMBER*, nomes de usuários por *USERNAME*, *hashtags* por *HASHTAG* e URLs por *URL*. Entidades relacionadas a números (telefones, datas, valores monetários etc) também foram substituídas por *NUMBER*. O uso dessas *tags* permite a obtenção de uma representação vetorial para esse tipo de informação, que no pré-processamento utilizado anteriormente era simplesmente descartada. Repetições consecutivas de caracteres superiores a 3 foram limitadas a apenas 3 – *legaaaal*, por exemplo, é substituído por *legaaal*. Sinais de pontuação foram removidos, por não fornecerem informações relevantes aos métodos de normalização (que são apenas lexicais). Uma modificação da ferramenta Tweetokenize⁵ foi utilizada para tokenizar o texto. Foram adicionadas expressões regulares para reconhecer formato de datas e telefones brasileiros, bem como para lidar com caracteres acentuados. A versão modificada para o português brasileiro está disponível publicamente para uso em github.com/tfcbertaglia/enelvo/tree/master/enelvo/preprocessing/tokenizer. O Tweetokenize foi escolhido por ser mais fácil de adaptar que outras ferramentas. Seguindo os experimentos anteriores, todos os modelos foram treinados com 500 dimensões utilizando a arquitetura *skip-gram*. Após o treinamento das novas *embeddings*, o método de aprendizado de léxicos foi utilizado com base em cada modelo. Para ilustrar o resultado obtido após a execução desta técnica de geração, a Tabela 27 apresenta os cinco primeiros candidatos gerados por cada modelo de *word embeddings* para o ruído *pq* (porque), dispostos em ordem decrescente de acordo com a pontuação.

⁵ github.com/jaredks/tweetokenize

Tabela 27 – Correções aprendidas pelo método de geração de candidatos por *embeddings*.

Word2Vec	Wang2Vec	FastText
porque - 0,20	porque - 0,23	pqu - 0,63
que - 0,14	porquê - 0,18	q - 0,53
pois - 0,13	pois - 0,16	porque - 0,21
porquê - 0,13	afinal - 0,09	pois - 0,17
porém - 0,11	-	tipo - 0,16

Fonte: Elaborada pelo autor.

Todos os léxicos de correção foram capazes de encontrar a versão normalizada da palavra. No entanto, o modelo FastText pontuou-a somente em terceiro lugar. Outro ponto interessante é que duas palavras fora do padrão (*q* e *pqu*) acabaram sendo geradas como candidatos a correção. Isso se deve possivelmente a inconsistências no léxico da língua utilizado. Apesar de o método gerar os candidatos e pontuá-los concomitantemente, nesta seção serão apresentados resultados relacionados apenas à geração. A avaliação dos candidatos gerados a partir de cada modelo de *embedding* será feita utilizando a cobertura, o número médio de candidatos e o potencial de correção. A [Tabela 28](#) apresenta a cobertura (em porcentagem) obtida pela geração de candidatos a partir de cada modelo diferente de *embeddings*.

Tabela 28 – Cobertura da geração de candidatos por *word embeddings*.

	Modelo de Embedding		
	Word2Vec	Wang2Vec	FastText
Erro Ortográfico	84,48	87,52	86,21
Abreviação	91,73	84,77	83,55
Internetês	81,42	87,68	80,41

Fonte: Elaborada pelo autor.

O valor de cobertura obtido foi alto para todas as categorias de ruídos. Exceto por erros ortográficos, a geração de candidatos por meio de *word embeddings* superou por uma vasta margem os resultados obtidos pela geração por similaridade lexical. Para melhor avaliar o método, a [Tabela 29](#) apresenta o número médio de candidatos gerados e o potencial de correção obtido (em porcentagem).

Pode-se notar que, diferentemente da geração por similaridade lexical, o número de candidatos gerados foi baixo para todas as categorias. Esse fato, aliado à cobertura alta, resulta em um potencial de correção razoável. Consequentemente, o ranqueamento dos candidatos gerados é pouco trabalhoso, pois todos são de fato possíveis normalizações do ruído. Os resultados indicam que *word embeddings* podem ser utilizadas de maneira

Tabela 29 – Número médio de candidatos e potencial de correção para geração por *embeddings*.

	Modelo de Embedding					
	Word2Vec		Wang2Vec		FastText	
	Candidatos	Potencial	Candidatos	Potencial	Candidatos	Potencial
Erro Ortográfico	6,65	12,70	4,86	18,02	4,70	18,35
Abreviação	5,39	17,01	6,50	13,04	6,40	13,05
Internetês	8,17	9,97	7,83	11,20	6,00	13,40

Fonte: Elaborada pelo autor.

efetiva na normalização, sendo capazes de capturar satisfatoriamente a relação entre um ruído e sua versão normalizada.

6.4 Métodos de Ranqueamento de Candidatos

Após obter palavras candidatas a normalizar determinado ruído, é necessário selecionar qual de fato é a mais pertinente para substituí-lo. Nesse contexto, métodos de ranqueamento de candidatos são responsáveis por atribuir uma pontuação a cada palavra candidata. Ao término desse processo, é possível escolher o candidato (ou vários) mais provável para normalização. É importante ressaltar que o ranqueamento não seleciona necessariamente apenas o candidato de maior pontuação, mas sim gera uma lista ordenada contendo determinado número de candidatos. Desse modo, ainda há espaço para processamentos posteriores – que podem ser realizados especificamente de acordo com a necessidade da aplicação. Visando a levar em consideração essa característica, os métodos de ranqueamento serão avaliados quanto a cobertura de seus *top* – *n* candidatos. Ou seja, se a versão correta do ruído está contida na lista dos *n* candidatos de maior pontuação, considera-se que o método é capaz de normalizá-lo. Neste trabalho, serão considerados como valores de *n* 1, 2 e 3. Com *n* = 1, a avaliação diz respeito a um único candidato, ou seja, corresponde a definitivamente corrigir o ruído e substituí-lo por uma palavra considerada correta. Os métodos de ranqueamento seguem as mesmas abordagens que os de geração: são baseados em similaridade lexical e em *word embeddings*. Ambos serão detalhados nas próximas seções.

Ranqueamento por Similaridade Lexical

Conforme apresentado na [Seção 6.3](#), três medidas de similaridade lexical foram utilizada para gerar candidatos: distância de edição, LCS e *simlex*. No entanto, como discutido anteriormente, *simlex* apresentou cobertura baixa na geração e portanto candidatos gerados a partir dela não serão considerados no ranqueamento. Apesar disso, sua natureza restritiva a torna pertinente para ser utilizada como medida de pontuação. Desse modo, *simlex* será empregada como a pontuação no ranqueamento por similaridade lexical. Ou seja: para ranquear cada candidato *c* a normalizar *p*, basta calcular a *simlex* entre *c* e *p*

e ordenar todos os candidatos de acordo com o valor calculado. De modo a fornecer um meio de comparação, a frequência de cada candidato também será utilizada como medida de ranqueamento. Nesse caso, cada candidato c é pontuado de acordo com sua frequência de ocorrência no total de amostras de textos de CGU coletadas para criação do corpus TUB-UGC. Apesar de simples, essa medida é capaz de dar importância a palavras que ocorrem frequentemente e portanto efetuar correções pertinentes em determinadas situações. O ranqueamento será avaliado separadamente para candidatos gerados por meio da distância de edição e pela LCS, considerando os $top-1$, $top-5$ e $top-10$ candidatos de maior pontuação. Somente serão ranqueados candidatos oriundos do método geração com maior cobertura de cada medida de similaridade – ou seja, o com maior limiar. A [Tabela 30](#) apresenta a cobertura (em porcentagem) obtida utilizando-se a pontuação por *simlex* e por frequência para cada categoria de ruído em candidatos gerados por distância de edição ≤ 3 .

Tabela 30 – Cobertura obtida no ranqueamento de candidatos gerados por distância de edição.

Distância de Edição – 3						
	top-1		top-5		top-10	
	Simlex	Frequência	Simlex	Frequência	Simlex	Frequência
Erro Ortográfico	72,90	7,80	82,23	21,21	85,04	38,75
Abreviação	1,22	0,98	26,04	22,14	36,66	29,82
Internetês	5,49	11,14	14,50	21,83	22,13	24,42

Fonte: Elaborada pelo autor.

Os resultados obtidos para erros ortográficos foram satisfatórios, com a *simlex* sendo capaz de corrigir 72,90% dos ruídos com o candidato de maior pontuação e 85,04% com os dez maiores. Nessa categoria, a medida foi bem superior à frequência. A cobertura atingida nas demais categorias foi baixa, mostrando que o método não foi capaz de normalizar adequadamente abreviações e internetês. No entanto, esse comportamento já era esperado, devido ao fato de a cobertura da geração de candidatos ser insuficiente. Nesse contexto, pode-se considerar que o ranqueamento cumpriu razoavelmente bem seu papel, pois dentre os casos os quais poderia corrigir corretamente (considerando todos os candidatos possíveis), foi capaz de acertar uma parcela considerável. A [Tabela 31](#) apresenta a cobertura (em porcentagem) obtida utilizando a pontuação por *simlex* e por frequência para cada categoria de ruído em candidatos gerados pelo complemento da razão da LCS $\leq 0,8$.

Houve apenas uma ligeira diferença entre a cobertura obtida por candidatos gerados por distância de edição e por LCS. Com exceção de erros ortográficos, é possível considerar que ambas as medidas proveem candidatos de baixa similaridade efetiva, que dificultam a execução de um ranqueamento satisfatório. De modo geral, pode-se concluir que métodos baseados em similaridade lexical não são capazes, por si só, de normalizar

Tabela 31 – Cobertura obtida no ranqueamento de candidatos gerados por complemento da razão da LCS.

CLCSR – 0,8						
	top-1		top-5		top-10	
	Simlex	Frequência	Simlex	Frequência	Simlex	Frequência
Erro Ortográfico	72,84	5,89	82,50	13,46	85,08	14,09
Abreviação	1,22	31,27	26,04	46,22	36,93	46,97
Internetês	5,49	1,52	14,96	6,41	22,74	8,54

Fonte: Elaborada pelo autor.

internetês e abreviações. Por outro lado, apresentam bons resultados ao lidar com erros ortográficos – nos quais o ruído tem alta similaridade lexical com sua versão correta.

Ranqueamento por Word Embeddings

O ranqueamento por *word embeddings* faz parte da técnica apresentada na [Seção 6.3](#). Como descrito, o método gera candidatos e os ranqueia simultaneamente. Porém, visando a manter a organização do texto, a avaliação da etapa de ranqueamento será realizada nesta seção. Para tanto, o mesmo procedimento usado na avaliação dos métodos por similaridade lexical será empregado. A cobertura será computada com base nos *top – 1*, *top – 5* e *top – 10* candidatos com maior pontuação. Os léxicos de correção gerados por cada modelo de *embedding* (Word2Vec, Wang2Vec e FastText) serão avaliados separadamente. A [Tabela 32](#) apresenta a cobertura (em porcentagem) obtida utilizando o ranqueamento por meio de *word embeddings* para cada categoria de ruído em candidatos gerados pelo modelo Word2Vec.

Tabela 32 – Cobertura obtida no ranqueamento de candidatos gerados pelo modelo Word2Vec.

Word2Vec			
	top-1	top-5	top-10
Erro Ortográfico	81,02	84,39	84,48
Abreviação	75,41	91,73	91,73
Internetês	74,52	80,96	81,42

Fonte: Elaborada pelo autor.

A cobertura obtida foi satisfatória em todas as categorias. É interessante notar que houve pouca variação de cobertura entre os diferentes números de candidatos considerados. Isso se deve principalmente ao fato de que o modelo de *word embeddings* foi capaz de captar similaridades complexas entre palavras – gerando, conseqüentemente, candidatos que facilitam a etapa de ranqueamento. O modelo de Word2Vec obteve o melhor resultado na normalização de abreviações, sendo capaz de corrigir corretamente 75,41% delas com

o candidato de maior pontuação e 91,73% com os cinco maiores. Não houve mudança entre o *top-5* e o *top-10* pois os métodos baseados em *embeddings* têm um baixo número médio de candidatos, sendo que muitas vezes não há todos os 10 para modificar a cobertura do *top-10*. A [Tabela 33](#) apresenta a cobertura (em porcentagem) obtida utilizando o ranqueamento por meio de *word embeddings* para cada categoria de ruído em candidatos gerados pelo modelo Wang2Vec.

Tabela 33 – Cobertura obtida no ranqueamento de candidatos gerados pelo modelo Wang2Vec.

	Wang2Vec		
	top-1	top-5	top-10
Erro Ortográfico	86,57	87,52	87,52
Abreviação	77,92	84,77	84,77
Internetês	80,09	87,44	87,67

Fonte: Elaborada pelo autor.

Em geral, os resultados obtidos pelo modelo Wang2Vec foram bem similares aos do Word2Vec. No entanto, considerando apenas a cobertura *top-1*, os valores obtidos foram consideravelmente superiores. Considerando que este modelo de *embedding* dá ênfase à sintaxe, é possível que esse tipo de informação seja de grande importância para a normalização textual – ou seja, é preciso ir além da similaridade lexical. A [Tabela 34](#) apresenta a cobertura (em porcentagem) obtida utilizando o ranqueamento por meio de *word embeddings* para cada categoria de ruído em candidatos gerados pelo modelo FastText.

Tabela 34 – Cobertura obtida no ranqueamento de candidatos gerados pelo modelo FastText.

	FastText		
	top-1	top-5	top-10
Erro Ortográfico	84,85	86,21	86,21
Abreviação	69,49	83,55	83,55
Internetês	72,13	80,41	80,41

Fonte: Elaborada pelo autor.

O modelo FastText não se destacou em nenhuma categoria, obtendo resultados condizentes aos demais modelos. Em média, sua cobertura é a pior das três. Por ser um modelo de *word embeddings* que enfatiza informações morfológicas, é possível que a ligeira perda de desempenho seja devida a sua característica de representar palavras com muito caracteres em comum como próximas no espaço. Como visto nos métodos por similaridade lexical, esse tipo de semelhança por si só pode ser ruim para a normalização. A combinação dos candidatos gerados pelos diferentes modelos de *embeddings* de modo a

tentar aumentar a cobertura do ranqueamento foi testada empiricamente, mas não atingiu resultados relevantes que justificassem o aumento no número médio de candidatos a serem avaliados. De modo geral, o uso de *word embeddings* proveu resultados altamente satisfatórios tanto para geração de candidatos quanto para ranqueamento. *Embeddings* possuem outras vantagens, além do valor superior de cobertura obtido. Primeiramente, os modelos treinados podem ser atualizados com textos novos – possibilitando assim o recálculo da representação das palavras. Isso é especialmente útil para lidar com texto de novos domínios, não considerados previamente, e para acompanhar a mudança temporal da língua. Considerando que expressões de internetês e abreviações surgem continuamente na *web*, retrainar as *embeddings* pode ser essencial para manter a eficácia dos métodos de normalização. Outra vantagem desse tipo de representação é no desempenho em se calcular similaridades. Diferentemente de medidas lexicais, as quais requerem algoritmos custosos de programação dinâmica para serem calculadas, a similaridade entre *embeddings* é apenas a distância entre dois vetores no espaço, que pode ser calculada eficientemente como produtos matriciais – operações que podem ser altamente otimizadas computacionalmente.

Correção de Erros Ortográficos Contextuais

A normalização de erros ortográficos contextuais deve ser tratada de modo particular. Até o momento, a correção de todos os ruídos foi realizada apenas lexicalmente – utilizando somente informações sobre a palavra fora do padrão e sobre seus possíveis substitutos. No entanto, erros ortográficos contextuais são palavras reconhecidas pela língua e só são consideradas ruidosas devido ao contexto em que se encontram. Desse modo, é necessário desenvolver técnicas capazes de considerar o contexto no qual a palavra está inserida. Tais métodos enquadram-se em métodos de ranqueamento pois não é necessário gerar candidatos a substituí-los, visto que para normalizar um erro contextual basta definir se ele deve ser acentuado ou não. Na prática, cada ruído contextual possui dois candidatos – sua versão com e sem acento.

Recentemente, o uso métodos baseados em redes neurais para representação de contexto em PLN vem crescendo. Trabalhos como (CHO *et al.*, 2014; IKEDA; SHINDO; MATSUMOTO, 2016) utilizam arquiteturas neurais profundas com o objetivo de obter representações contextuais de palavras, sentenças e até mesmo documentos inteiros. Seguindo essa abordagem, propõe-se uma arquitetura que combina um *autoencoder* para representação do contexto de sentenças e uma rede MLP para classificar a presença de acento na palavra. A Figura 27 apresenta a estrutura geral da arquitetura proposta.

O funcionamento da arquitetura é simples. Para o treinamento, são necessárias sentenças que contêm palavras que existem com ou sem acentos (como *esta* e *está*, *vicio* e *vício*, etc.). É importante ressaltar que as sentenças devem ser oriundas de textos redi-

Figura 27 – Arquitetura neural para correção de erros ortográficos contextuais.

Fonte: Elaborada pelo autor.

gidos em língua padrão, visando a garantir que a acentuação da palavra esteja de fato correta no contexto em que se encontra. Em seguida, a palavra é **retirada** da sentença – dividindo a entrada em duas: a palavra a ser avaliada e o fragmento de sentença a que ela pertence. Esse passo é necessário pois se deseja obter uma representação de contexto que não dependa da palavra avaliada. Em seguida, o *encoder*, que é uma BRNN, recebe o fragmento de sentença como entrada e gera uma representação para ele. A representação obtida, que simboliza o contexto, é então combinada à representação da palavra avaliada (por meio da concatenação dos vetores das duas representações). Após a combinação, o *dropout* é utilizado com o objetivo de evitar *overfitting*. A representação combinada é por fim passada por uma camada totalmente conectada (MLP) de modo a produzir previsões usando a função de ativação *softmax*. A previsão é binária, indicando se a palavra avaliada contém, ou não, acento. Para exemplificar, considere a sentença *hoje o dia está frio*. No treinamento, as duas entradas seriam *está* e *hoje o dia frio*. O fragmento seria processado pelo *encoder*, que gera como saída uma representação intermediária. Em seguida, a representação é concatenada à *embedding* de *está* e o vetor resultante é processado pela camada totalmente conectada. Por fim, a saída esperada é 1 (com acento), então os pesos são ajustados de acordo.

Como são necessários textos na forma padrão para o treinamento, um conjunto de 1 milhão de sentenças extraídas da Wikipédia foi utilizado. Foram coletadas somente sentenças que continham candidatos a erros ortográficos contextuais. Para evitar ambiguidade contextual, sentenças que continham versões com e sem acento da mesma palavra foram removidas (por exemplo em *vou esperar até que você ate o nó*). O treinamento foi realizado utilizando *embeddings* do modelo Word2Vec, obtidas por meio do repositório de *embeddings* do NILC⁶. Os hiper-parâmetros utilizados no treinamento foram definidos com base na literatura da área, sendo que os principais estão listados na Tabela 35.

⁶ <nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc>

Tabela 35 – Principais hiper-parâmetros de treinamento da arquitetura *encoder*.

Hiper-parâmetro	Valor
Tamanho das embeddings	500
Unidades recorrentes	500
Taxa de dropout	0,5
Taxa de aprendizado	0,001

Fonte: Elaborada pelo autor.

Para avaliar a capacidade de aprendizado da arquitetura, seu desempenho foi testado no conjunto de treino (textos da Wikipédia) por meio de validação cruzada com 5 *folds*. Para avaliar as predições em textos de CGU, a arquitetura foi utilizada para corrigir erros ortográficos contextuais do córpus TUB-UGC. Os resultados de ambos constam na Tabela 36, com valor em porcentagem de cobertura precisão e F1.

Tabela 36 – Avaliação da arquitetura *encoder* para correção de erros ortográficos contextuais.

	Treino	Teste
Cobertura	88,20	70,67
Precisão	98,87	91,98
F1	88,00	79,93

Fonte: Elaborada pelo autor.

O valor de F1 obtido no conjunto de treino foi alto, indicando que a arquitetura é capaz de aprender quando acentuar ou não uma palavra. No entanto, os resultados foram consideravelmente inferiores no conjunto de teste. Há vários motivos que justificam este comportamento. Os textos utilizados no treinamento são escritos na língua padrão, com poucos ruídos. Já, no teste, são todas amostras de CGU – altamente ruidosas e com várias palavras não reconhecidas. Essa mudança de domínio dificulta o funcionamento da rede neural. Outro ponto é que as *embeddings* utilizadas foram induzidas a partir de córpus também escritos na língua padrão, portanto as representações geradas não se adequam bem ao vocabulário dos textos de CGU. Ainda assim, a F1 obtida (**79,93%**) indica uma capacidade considerável de correção. A ferramenta UGCNormal, por exemplo, foi capaz de corrigir corretamente apenas **34,5%** dos erros ortográficos contextuais contidos no córpus de *reviews* de produtos na qual foi avaliada. Uma maneira de comparar o método é utilizando um *baseline* – alguma técnica simples que também realize a tarefa. Optou-se por utilizar o *baseline* da frequência: a palavra avaliada foi sempre substituída por sua versão com maior número de ocorrências no córpus. *Esta*, por exemplo, foi sempre acentuada, já que *está* tem frequência maior. A Tabela 37 apresenta os valores (em porcentagem) da

cobertura, precisão e F1 obtidos pelo *baseline* na correção erros ortográficos contextuais.

Tabela 37 – Avaliação do *baseline* de frequência para correção de erros ortográficos contextuais.

	Valor (%)
Cobertura	64,40
Precisão	91,30
F1	75,50

Fonte: Elaborada pelo autor.

Pode-se notar que os valores obtidos pelo *baseline* foram inferiores, porém próximos, aos atingidos pela arquitetura neural. Para confirmar a diferença entre os resultados, o teste de Wilcoxon (*Signed-Rank*) foi realizado. Ao nível de significância de 5%, a hipótese da melhora de desempenho não ser significativa foi rejeitada (*p-value* obtido igual a 0.00512) (GEHAN, 1965). No entanto, é difícil tirar conclusões a partir desse resultado. O *baseline* de frequência obteve sucesso no corpus pois a maioria das palavras avaliadas deveria ser de fato acentuada, entretanto não é possível afirmar que essa característica predomina em todo texto de CGU sem conduzir mais experimentos. Já a arquitetura neural, pelo menos em textos bem formados, mostrou ser robusta e capaz de identificar erros contextuais em uma grande diversidade de sentenças. Consequentemente, pode-se concluir que erros ortográficos contextuais podem ser satisfatoriamente identificados e corrigidos pela arquitetura proposta.

6.5 O Normalizador Enelvo

Com o objetivo de prover uma ferramenta de normalização completa, todos os métodos aqui apresentados serão disponibilizados publicamente para uso. A ferramenta pode ser acessada em [<github.com/tfcbertaglia/enelvo>](https://github.com/tfcbertaglia/enelvo). Os recursos utilizados no desenvolvimento deste projeto também serão disponibilizados, se permitido pela licença que os rege. Léxicos e corpus serão disponibilizados juntamente à ferramenta, enquanto os modelos de *embeddings* utilizados estarão disponíveis por meio do repositório de *word embeddings* do NILC⁷. Esta seção apresenta brevemente o funcionamento de cada um dos módulos que compõem a ferramenta desenvolvida neste projeto – que foi denominada Enelvo. A Figura 28 ilustra o fluxo geral seguido pela ferramenta.

A ferramenta recebe como entrada um texto de CGU. Como a segmentação de sentenças é realizada por um sistema externo, considera-se a sentença como a unidade de entrada da ferramenta para melhor explicar o seu funcionamento. Em seguida, a sentença é pré-processada com o objetivo de deixá-la no formato requerido pelos demais métodos.

⁷ [<nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc>](https://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc)

Figura 28 – Fluxo de funcionamento da ferramenta Enelvo.

Fonte: Elaborada pelo autor.

Após esse processo, o módulo de análise é responsável por identificar possíveis palavras fora do padrão, que são consideradas ruídos. Estes ruídos são então recebidos pelo módulo de geração de candidatos, que encontra possíveis correções para cada um deles. Por fim, o módulo de ranqueamento atribui uma pontuação para cada um dos candidatos, possibilitando desse modo selecionar qual o mais adequado para substituir a palavra ruidosa. É importante ressaltar que nessa etapa pode haver intervenção da aplicação – ou seja, podem ser inseridas informações que flexibilizem o processo de ranqueamento de acordo com as necessidades da aplicação específica. Para ilustrar o funcionamento do sistema, o seguinte exemplo de uma *review* hipotética sobre um celular será utilizado:

comprei esse celular da sansung q eh mt ruuuuim :(
fikei decepcionada, um da apple era melhor #xatiada

Módulo de Pré-processamento

É o passo inicial do sistema. O pré-processamento é responsável por receber as amostras de CGU, que são entrada do sistema, e prepará-las de modo a serem processadas pelos módulos subsequentes. A preparação do texto envolve segmentação de sentenças, inserção de pontuação, capitalização, tokenização e remoção de sequências de caracteres com mais de três repetições (*legaaaaal*, por exemplo, é substituído por *legaaal*). Esse último passo auxilia na geração de candidatos.

A segmentação de sentenças é por si só uma tarefa complexa de PLN. Como foge do escopo desse trabalho, serão utilizados sistemas já prontos para efetuar essa etapa. No UGCNormal, a ferramenta proposta por [Condori e Pardo \(2015\)](#) é utilizada. A capitaliza-

ção de início de sentença será efetuada com base na saída do segmentador. A tokenização é responsável por separar as sentenças em *tokens*. Enelvo disponibiliza um tokenizador projetado especialmente para CGU. A principal característica que o diferencia de um tokenizador comum é a capacidade de identificar emojis, emoticons, *hashtags* e nomes de usuários Twitter. Entidades relacionadas a números (datas, telefones, valores monetários etc.) também podem ser agrupados como *tokens* únicos. A ferramenta ainda disponibiliza a opção de substituir determinados tipos de *tokens* por etiquetas determinadas de acordo com a necessidade da aplicação. Nomes de usuários, por exemplo, podem ser substituídos por USUÁRIO.

A saída do pré-processamento é o texto segmentado em sentenças, tokenizado e corretamente capitalizado de acordo com a pontuação inserida pelo segmentador. Considerando a *review* supracitada como exemplo, a saída desse módulo seria:

```
Comprei esse celular da sansung q eh mt ruuuim :( .  
Fikei decepcionada , um da apple era melhor #xatiada .
```

Módulo de Análise

Considerando que o sistema deverá se adequar às necessidades da aplicação, é fundamental prover meios para entender quais são essas necessidades. Nesse contexto, o módulo de análise é responsável por gerar estatísticas e informações que permitam melhor compreender as amostras de texto a serem tratadas. Alguns exemplos de estatísticas são: número total de palavras fora do padrão; média de palavras fora do padrão por sentença; tamanho médio de sentença; tamanho médio de *tokens* e outros dados pertinentes. Essas informações podem ser apresentadas de forma descritiva (apenas escrevendo-as) e também de forma gráfica, tornando o processo de tomada de decisão quanto aos ruídos mais fácil e intuitivo.

O módulo de análise também é responsável pela identificação das palavras fora do padrão. Essa tarefa pode ser feita por meio do uso de léxicos ou de abordagens de aprendizado de máquina. Como discutido na [Seção 6.2](#), a identificação por meio de léxicos é mais adequada para ser utilizada na prática. Apesar disso, ambas são disponibilizadas para uso no sistema. Cabe à aplicação escolher qual será utilizada. Contudo, recomenda-se o uso dos léxicos por ser mais eficiente computacionalmente. Destacando palavras fora do padrão com a cor vermelha, a saída desse módulo para a *review* de exemplo seria:

```
Comprei esse celular da sansung q eh mt ruuuim :( .  
Fikei decepcionada , um da apple era melhor #xatiada .
```

É importante ressaltar que *tokens* constituídos apenas de sinais de pontuação (como *emoticons*) não serão considerados palavras, portanto não são identificados como

fora do padrão. *Hashtags* são uma particularidade e podem ser tratadas de duas maneiras distintas. A primeira opção é não considerá-las como fora do padrão, visto que em geral são utilizadas para representar palavras-chave. Outra alternativa é separar o símbolo # e considerar o restante como uma palavra só – que será consequentemente tratada como as demais. Nos experimentos conduzidos neste trabalho, esse tipo de *token* foi simplesmente removido do texto, portanto não passou pelas etapas de normalização.

Módulo de Geração de Candidatos

É o módulo responsável por analisar as palavras fora do padrão e gerar candidatos para normalizá-las. Nele, constam os métodos de geração por similaridade lexical e por *word embeddings*. No caso da geração por similaridade lexical, o sistema permite que qualquer medida seja utilizada – não somente as avaliadas neste trabalho. Também é possível utilizar várias medidas e gerar candidatos com base na combinação entre elas. No método que utiliza *word embeddings*, é necessário obter os modelos treinados no repositório do NILC. O funcionamento segue o método descrito na [Seção 6.3](#).

Por fim, os candidatos obtidos para todas as palavras fora do padrão são enviados ao próximo módulo. Para ilustrar, os candidatos hipoteticamente gerados para cada palavra fora do padrão são representados em azul:

```
Comprei esse celular da sansung (Samsung) q (que, quê)
eh (é, eu) mt (muito, muita, mito) ruuuim (ruim) :( .
Fikei (Fiquei) decepcionada , um da apple (Apple)
era melhor #xatiada (#xatiada, # chateada) .
```

Módulo de Ranqueamento

Uma vez gerados diversos candidatos de normalização para cada palavra, é necessário avaliar cada um e determinar qual o mais adequado para efetuar a substituição. Esse módulo é responsável por esse procedimento, utilizando técnicas baseadas em similaridade lexical e em *word embeddings*, de maneira análoga à geração de candidatos. É importante ressaltar que as técnicas são aplicadas em conjunto, sendo que a similaridade lexical é empregada na etapa de expansão, conforme descrito na [Seção 6.3](#). A saída desse módulo pode ser o candidato de maior pontuação ou uma lista de n candidatos ordenados por pontuação, sendo n um valor definido pela aplicação.

Para ilustrar o funcionamento desse módulo, a [Tabela 38](#) apresenta um ranqueamento hipotético da palavra fora do padrão *mt*, com base na saída exemplo do módulo anterior.

Por fim, como saída final do processo de normalização, o seguinte texto pode ser obtido:

Tabela 38 – Exemplo de ranqueamento de candidatos

Candidato	Pontuação
Muito	1,00
Muita	0,75
Mito	0,50

Fonte: Elaborada pelo autor.

Comprei esse celular da Samsung que é muito ruim :(.
Fiquei decepcionada, um da Apple era melhor #xatiada .

Avaliação

Por fim, esta seção tem como objetivo avaliar o processo de normalização como um todo e comparar o normalizador Enelvo ao UGCNormal. Para tanto, é necessário deixar claro o que está sendo avaliado em cada categoria de ruído. Como o UGCNormal não fornece lista de candidatos (sempre retorna apenas uma sugestão de normalização), todas as comparações são efetuadas utilizando o *top* – 1 candidato gerado pelo Enelvo. Para erros ortográficos, abreviações e internetês, o candidato com maior pontuação atribuída pelo melhor resultado do método baseado em *embeddings* é comparado à correção proposta pelo UGCNormal. Para erros ortográficos contextuais, o resultado da arquitetura neural de *encoder* é avaliado. Para nomes próprios e abreviações, o resultado corresponde à precisão obtida pelo método de identificação baseado em aprendizado de máquina para as *tags NP* e *SI*, respectivamente. Para estrangeirismos, é avaliado o resultado do mesmo método para a *tag OK* – que indica que não há ação a ser realizada nesse tipo de ruído. A Tabela 39 apresenta os resultados obtidos (em porcentagens) pelas ferramentas Enelvo e UGCNormal na identificação e correção de diferentes tipos de ruído contidos no corpus TUB-UGC. Considerando que este corpus contém textos de diversas origens e portanto engloba os demais corpus de CGU disponíveis, optou-se por avaliar as ferramentas somente nele.

Os resultados obtidos pela ferramenta Enelvo foram superiores em todas as categorias, exceto para nomes próprios. A fim de verificar se a diferença de desempenho entre as ferramentas foi obtida ao acaso, o teste de Wilcoxon (*Signed-Rank*) foi realizado. Ao nível de significância de 5%, a hipótese da melhora de desempenho não ser significativa foi rejeitada (*p-value* obtido igual a 0.02034) (GEHAN, 1965). Portanto, é seguro afirmar que o desempenho do Enelvo é de fato superior ao UGCNormal. A diferença na categoria de nomes próprios se deve ao fato de que a *tag NP* foi a mais difícil para a RCNN classificar corretamente. No entanto, a abordagem baseada em léxicos é capaz de identificar 100% dos nomes próprios, siglas e estrangeirismos, pois os léxicos que os identificam foram atualizados com as anotações do corpus. Obviamente, esse tipo de comparação é injusta

Tabela 39 – Comparação de resultados entre as ferramentas Enelvo e UGCNormal no corpus TUB-UGC.

Tipo de Ruído	Enelvo	UGCNormal
Erros ortográficos	<u>86,57</u>	79,32
Ortográficos Contextuais	<u>79,93</u>	66,78
Internetês	<u>80,09</u>	30,38
Abreviações	<u>77,92</u>	71,71
Siglas	<u>78,78</u>	38,05
Nomes Próprios	67,65	<u>72,30</u>
Estrangeirismos	<u>97,54</u>	69,67

Fonte: Elaborada pelo autor.

devido ao léxico já “conter as respostas”. Portanto, somente a abordagem automática é comparada ao do UGCNormal. No geral, os resultados foram satisfatórios e apresentaram um percentual elevado de correção. A ferramenta UGCNormal, apesar de obter resultados inferiores, ainda foi capaz de corrigir parte considerável dos ruídos. As categorias de internetês e de siglas foram exceções. Para internetês, o UGCNormal teve dificuldade em lidar com repetições de caracteres, não sendo capaz de normalizar, por exemplo, *partiuiiiu* para *partiu*. Já na categoria de siglas, seu funcionamento foi inconsistente. Algumas siglas foram somente capitalizadas, algumas foram expandidas corretamente para sua versão completa e outras forma expandidas erroneamente. A ferramenta Enelvo foi consistente em todas as categorias, comprovando que o processo de normalização pode ser realizado de maneira efetiva para diferentes modalidades de CGU com baixa dependência de recursos.

CONCLUSÕES E TRABALHOS FUTUROS

Este capítulo apresenta uma visão geral sobre o trabalho desenvolvido, bem como uma discussão sobre os pontos principais observados em relação aos experimentos realizados.

O conteúdo gerado por usuário é uma fonte rica de informação. Processar textos deste tipo – que incluem *reviews* de produtos, *blogs* e publicações em redes sociais – é uma tarefa relevante para extração de conhecimento, principalmente para a área de PLN. Entretanto, há desafios consideráveis em processar esse tipo de conteúdo. A linguagem utilizada em CGU diverge em diversas maneiras da língua padrão. É comum encontrar erros ortográficos, abreviações, expressões de internetês e outros tipos de ruídos em textos oriundos da *web*. Ferramentas tradicionais de PLN têm dificuldade em lidar com esse tipo de linguagem, visto que foram projetadas originalmente para tratar de textos estruturados na forma padrão. Nesse contexto, a normalização textual é uma tarefa que tem como objetivo adaptar textos ruidosos, visando a torná-los mais próximas da norma culta da língua.

Antes de propor métodos de normalização capazes de lidar com textos de CGU, foi necessário efetuar uma ampla análise sobre ruídos. Para tanto, um corpus de textos de CGU foi anotado de modo a conter anotações sobre ruídos, suas correções e suas categorias. Três modalidades de CGU foram incluídas: *reviews* de produtos do Buscapé, publicações no fórum da UOL e *tweets*. Até então, os corpus disponíveis continham apenas *reviews*. O processo de anotação foi fundamental para gerar recursos que subsidiassem a construção de métodos de normalização. Além disso, ele também possibilitou a realização de uma investigação metódica sobre as diferentes categorias de ruídos e de como separá-las. Em decorrência disso, uma categorização bem definida foi proposta, visando a evitar ambiguidades. O corpus anotado, denominado TUB-UGC, conta com milhares de ruídos distribuídos entre todas as categorias.

Para o desenvolvimento de técnicas, o processo de normalização foi basicamente dividido em três etapas: identificação de ruídos, geração de candidatos e ranqueamento de candidatos. Cada uma delas foi investigada individualmente, sendo desempenhada por meio de métodos diferentes. Para identificação de ruídos, exploraram-se abordagens baseadas em léxicos e em aprendizado de máquina. A primeira consiste no uso de listas de palavras contendo ruídos e suas correções em diversas categorias. Constatou-se que os léxicos existentes para o português, que foram extraídos de *córpus de reviews*, não são capazes de alcançar uma boa cobertura em *córpus* que contêm diferentes modalidades de CGU. Estes léxicos foram então expandidos, acrescentando as anotações do TUB-UGC a eles. A abordagem baseada em aprendizado de máquina tem como principal objetivo eliminar a necessidade de utilizar léxicos. Um arquitetura neural profunda, formada por uma rede neural recorrente convolucional, foi treinada em uma amostra de 1 milhão de *tweets* anotados por meio de supervisão distante. A arquitetura foi então avaliada na identificação de ruídos do *córpus* TUB-UGC. A precisão média obtida foi de 93,64% sendo em geral consistente nas diferentes categorias de ruídos. Desse modo, comprovou-se que é factível identificar ruídos com baixa dependência de recursos.

Métodos de geração de candidatos foram abordados de duas maneiras: por meio de medidas de similaridade lexical e com base em *word embeddings*. A primeira abordagem não apresentou resultados satisfatórios. Utilizando as medidas de distância de edição, LCS e *simlex*, o número de candidatos gerados foi grande e com baixa cobertura das correções. Com exceção de erros ortográficos, nos quais este método foi capaz de gerar em média 88% dos candidatos corretos, os resultados foram inferiores a 50% de cobertura. Todavia, concluiu-se que a medida *simlex* pode ser usada de maneira adequada na etapa de ranqueamento. Por fim, um método baseado em *word embeddings* foi proposto. Três modelos de representação vetorial de palavras foram avaliados na geração de candidatos. Todos apresentaram resultados superiores aos obtidos pelas medidas de similaridade lexical. A cobertura foi boa para todas as categorias de ruídos, mostrando que as *embeddings* são capazes de capturar a semelhança entre um ruído e sua versão correta. O modelo Wang2Vec apresentou os melhores resultados, com cobertura média de 86%.

O ranqueamento de candidatos utilizou as mesmas técnicas de geração. Métodos baseados em similaridade lexical não são por si só bons para pontuar os candidatos. No entanto, a medida *simlex* combinada à similaridade do espaço vetorial das *word embeddings* apresentou os melhores resultados – obtendo taxa de correção média de 86% considerando os *top – 10* candidatos ranqueados pelo modelo Wang2Vec. O ranqueamento de candidatos proposto oferece também a possibilidade de interação com a aplicação, sendo capaz de incluir novas medidas conforme necessidade. Ainda nesse contexto, um método para tratar especificamente de erros ortográficos contextuais foi proposto. Utilizando uma arquitetura neural *encoder*, foi possível corrigir esse tipo de ruído com 91,98% de precisão e 79,93% de F1 no *córpus* TUB-UGC. Todas as técnicas propostas foram combinadas em uma

ferramenta de normalização disponível publicamente.

Embora os métodos tenham sido projetados para o português brasileiro, há efetivamente pouca dependência de língua. Com exceção da medida *simlex*, que foi adaptada de modo a conter informações específicas do português (mas que pode ser substituída pela versão original proposta por Hassan e Menezes (2013)), todas as técnicas podem ser replicadas exatamente como descritas e aplicadas a outros idiomas. É necessário, ainda assim, que os recursos correspondentes estejam disponíveis para a língua – principalmente léxicos e modelos de *word embeddings*.

Pode-se concluir que os objetivos gerais do trabalho foram cumpridos. A anotação do corpus TUB-UGC permitiu uma ampla análise de ruídos, mostrando que diferentes modalidades de CGU têm de fato distribuições de ruídos diferentes. Desse modo, é válido desenvolver técnicas genéricas e independente de domínio. Os métodos propostos conseguem cumprir esses requisitos, pois utilizam poucos recursos restritos e podem ser facilmente expandidos e adaptados. Além disso, foram avaliados para todas as categorias de ruído em textos diferentes. Portanto, é possível afirmar que são capazes de corrigir efetivamente palavras ruidosas em textos de CGU.

Apesar de todos os pontos positivos, ainda há limitações nos métodos desenvolvidos. Um ponto a ser destacado é a correção de erros ortográficos contextuais. Embora a arquitetura proposta tenha atingindo resultados bem altos no conjunto de treino (88% de F1), no conjunto de teste (que contém textos de CGU), os resultados foram apenas ligeiramente superiores ao *baseline*. De acordo com o treinamento, a arquitetura neural em teoria aprendeu a definir se uma palavra deve ou não ser acentuada de acordo com seu contexto. Na prática, entretanto, seu desempenho foi aquém do esperado ao lidar com CGU. Desse modo, é válido investigar com mais profundidade as causas deste comportamento, possivelmente buscando outras maneiras de corrigir ruídos desse tipo. Outro ponto importante, que não foi explorado no trabalho, é avaliar extrinsecamente o normalizador – ou seja, analisar o desempenho de tarefas de PLN ao processar textos de CGU ruidosos antes e após serem normalizados.

Partindo das limitações mencionadas, há várias possibilidades a explorar para trabalhos futuros. Primeiramente, a avaliação extrínseca do normalizador em tarefas de PLN permitiria compreender com mais clareza o impacto da normalização nos textos. Para tanto, é necessário avaliar as diferenças de desempenho obtidas pela correção de cada tipo de ruído. Outro ponto a ser investigado é a correção de ruídos por meio de aprendizado de máquina. Uma possibilidade é utilizar arquiteturas neurais para desempenhar essa tarefa. Essa abordagem é desafiadora, pois é difícil modelar o problema como uma tarefa de aprendizado supervisionado e há poucos dados disponíveis para treinamento. No entanto, em caso de sucesso, o processo de normalização seria ainda mais flexível e dependeria de menos recursos. Avaliar a arquitetura proposta para identificação de ruídos em outro

cópus também é interessante. Desse modo, seria possível verificar se ela foi capaz de aprender ruídos além dos contidos nos léxicos usados para anotar o cópus de supervisão distante. Outra possibilidade é explorar maneiras de integrar a normalização a tarefas de PLN. Em vez de utilizar um normalizador como etapa de pré-processamento – aplicado nos textos antes de serem utilizados pela tarefa –, seria ideal que ele fosse integrado à tarefa em si, aprendendo junto a ela qual a melhor maneira de efetuar a normalização dos textos que estão sendo processados.

REFERÊNCIAS

- ALMEIDA, G. A. de M.; AVANÇO, L.; DURAN, M. S.; FONSECA, E. R.; NUNES, M. d. G. V.; ALUÍSIO, S. M. Evaluating phonetic spellers for user-generated content in brazilian portuguese. **Proceedings of International Conference on Computational Processing of the Portuguese Language**, n. 1, p. 361–373, 2016. Citado na página [3](#).
- ANCHIÊTA, R. T.; NETO, F. A. R.; SOUSA, R. F. de; MOURA, R. S. Using stylometric features for sentiment classification. **Proceedings of International Conference on Intelligent Text Processing and Computational Linguistics**, p. 189–200, 2015. Citado na página [3](#).
- AVANÇO, L. V. **Sobre normalização e classificação de polaridade de textos opinativos na web**. 85 p. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo, 8 2015. Citado nas páginas [4](#), [12](#), [25](#), [51](#), [52](#), [53](#), [54](#), [71](#) e [72](#).
- AVANÇO, L. V.; BRUM, H. B.; NUNES, M. d. G. V. Improving opinion classifiers by combining different methods and resources. **Anais do XIII Encontro Nacional de Inteligência Artificial e Computacional**, Recife, PE, Brasil, p. 25–36, 2016. Citado na página [3](#).
- BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. **CoRR**, abs/1409.0473, p. 15, 2014. Citado nas páginas [64](#) e [65](#).
- BENGIO, Y.; DUCHARME, R.; VINCENT, P.; JANVIN, C. A neural probabilistic language model. **Journal of Machine Learning Research**, JMLR.org, v. 3, p. 1137–1155, mar. 2003. ISSN 1532-4435. Citado na página [33](#).
- BERTAGLIA, T. F. C.; NUNES, M. d. G. V. Exploring word embeddings for unsupervised textual user-generated content normalization. **Proceedings of the 2nd Workshop on Noisy User-generated Text**, p. 112–120, 2016. Citado nas páginas [98](#) e [101](#).
- BOJANOWSKI, P.; GRAVE, E.; JOULIN, A.; MIKOLOV, T. Enriching word vectors with subword information. **arXiv preprint arXiv:1607.04606**, p. 12, 2016. Citado nas páginas [34](#), [35](#) e [101](#).
- BRILL, E.; MOORE, R. C. An improved error model for noisy channel spelling correction. In: **Proceedings of the 38th Annual Meeting on Association for Computational Linguistics**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2000. (ACL '00), p. 286–293. Citado nas páginas [19](#), [5](#), [31](#), [57](#), [60](#) e [67](#).
- BRUM, H.; ARAUJO, F.; KEPLER, F. Sentiment analysis for brazilian portuguese over a skewed class corpora. **Computational Processing of the Portuguese Language: 12th International Conference, PROPOR 2016**, Springer International Publishing, p. 134–138, 2016. Citado na página [2](#).

- BRUM, H.; KEPLER, F. Análise de sentimentos para português brasileiro usando redes neurais recursivas. **Anais do IV Workshop de Iniciação Científica em Tecnologia da Informação e da Linguagem Humana (TiLic)**, p. 5, 2015. Citado na página 2.
- CARLETTA, J. Assessing agreement on classification tasks: The kappa statistic. **Journal of Computational Linguistics**, MIT Press, Cambridge, MA, USA, v. 22, n. 2, p. 249–254, jun. 1996. ISSN 0891-2017. Citado na página 82.
- CAZALA, J. **Synaptic: architecture-free neural network library for node.js and the browser**. 2016. Disponível online em github.com/cazala/synaptic. Citado na página 41.
- CHE, X.; WANG, C.; YANG, H.; MEINEL, C. Punctuation prediction for unsegmented transcript based on word vector. **Proceedings of LREC**, p. 654–658, 2016. Citado na página 89.
- CHO, K.; MERRIENBOER, B. van; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning phrase representations using rnn encoder–decoder for statistical machine translation. **Proceedings of Empirical Methods in NLP**, Association for Computational Linguistics, p. 1724–1734, 2014. Citado nas páginas 46 e 108.
- CHOLLETON, A. **Visual Word2Vec**. 2016. Disponível online em github.com/aubry74/visual-word2vec. Citado na página 35.
- CHOUDHURY, M.; SARAF, R.; JAIN, V.; MUKHERJEE, A.; SARKAR, S.; BASU, A. Investigation and modeling of the structure of texting language. **Int. J. Doc. Anal. Recognit.**, Springer-Verlag, v. 10, n. 3, p. 157–174, dez. 2007. ISSN 1433-2833. Citado nas páginas 19, 58, 60 e 67.
- CLARK, E.; ARAKI, K. Text normalization in social media: progress, problems and applications for a pre-processing system of casual english. **Procedia-Social and Behavioral Sciences**, Elsevier, v. 27, p. 2–11, 2011. Citado nas páginas 19, 5, 49 e 67.
- COHEN, J. A coefficient of agreement for nominal scales. **Educational and psychological measurement**, Sage Publications, v. 20, n. 1, p. 37–46, 1960. Citado na página 82.
- COLLOBERT, R.; WESTON, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. p. 160–167, 2008. Citado na página 33.
- COLLOBERT, R.; WESTON, J.; BOTTOU, L.; KARLEN, M.; KAVUKCUOGLU, K.; KUKSA, P. Natural language processing (almost) from scratch. **Journal of Machine Learning Research**, p. 2493–2537, 2011. Citado na página 33.
- CONDORI, R. L.; PARDO, T. A. Experiments on sentence boundary detection in user-generated web content. **Proceedings of the International Conference on Intelligent Text Processing and Computational Linguistics**, p. 227–237, 2015. Citado nas páginas 52, 72, 98 e 112.
- COOK, P.; STEVENSON, S. An unsupervised model for text message normalization. In: **Proceedings of the Workshop on Computational Approaches to Linguistic Creativity**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009. (CALC '09), p. 71–78. ISBN 978-1-932432-36-7. Citado nas páginas 19, 60 e 67.

- CORREA, E. A.; MARINHO, V. Q.; SANTOS, L. B. dos; BERTAGLIA, T. F. C.; TREVISIO, M. V.; BRUM, H. B. PELESent: Cross-domain polarity classification using distant supervision. **ArXiv e-prints – Accepted for Publication in BRACIS 2017**, p. 7, jul. 2017. Citado na página 89.
- DAMERAU, F. J. A technique for computer detection and correction of spelling errors. **Communications of the. ACM**, ACM, New York, NY, USA, v. 7, n. 3, p. 171–176, mar. 1964. ISSN 0001-0782. Citado nas páginas 25, 26, 31, 36 e 57.
- DAUGHERTY, T.; EASTIN, M. S.; BRIGHT, L. Exploring Consumer Motivations for Creating User-Generated Content. **Journal of Interactive Advertising**, v. 8, n. 2, p. 16–25, 2008. ISSN 1525-2019. Citado nas páginas 10, 13 e 16.
- DROUIN, M.; DAVIS, C. R u txtng? is the use of text speak hurting your literacy? **Journal of Literacy Research**, v. 41, n. 1, p. 46–67, 2009. ISSN 1086-296X. Citado na página 27.
- DURAN, M. S.; AVANÇO, L.; NUNES, M. G. V. A Normalizer for UGC in Brazilian Portuguese. **Proceedings of the ACL 2015 Workshop on Noisy User-generated Text**, p. 38–47, 2015. Citado nas páginas 19, 5, 6, 7, 29, 37, 51, 54, 67, 71 e 83.
- EISENSTEIN, J. What to do about bad language on the internet. **Proceedings of the Anual Conference of the North American Chapter of ACL-HLT**, n. June, p. 359–369, 2013. Citado nas páginas 26, 27 e 28.
- FILHO, P. P. B.; PARDO, T. A.; ALUISIO, S. M. An evaluation of the brazilian portuguese liwc dictionary for sentiment analysis. **Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology (STIL)**, p. 215–219, 2013. Citado na página 2.
- FLEISS, J. L.; COHEN, J. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. **Educational and psychological measurement**, Sage Publications Sage CA: Thousand Oaks, CA, v. 33, n. 3, p. 613–619, 1973. Citado na página 82.
- FONSECA, E. R.; ALUISIO, S. M. A deep architecture for non-projective dependency parsing. **NAACL; Workshop on Vector Space Modeling**, p. 56–61, 2015. Citado na página 40.
- FOSTER, J.; ÇETINOGLU, Ö.; WAGNER, J.; ROUX, J. L.; NIVRE, J.; HOGAN, D.; GENABITH, J. van. From news to comment: Resources and benchmarks for parsing the language of web 2.0. **Proceedings of the Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011**, p. 893–901, 2011. Citado na página 29.
- FREITAS, C.; MOTTA, E.; MILIDIÚ, R.; CESAR, J. Vampiro que brilha... rá! desafios na anotação de opinião em um corpus de resenha de livros. **Anais do XI Encontro de Linguística de Corpus – ELC 2012**, v. 11, p. 13, 2012. Citado na página 2.
- GEHAN, E. A. A generalized wilcoxon test for comparing arbitrarily singly-censored samples. **Biometrika**, Oxford University Press, v. 52, n. 1-2, p. 203–224, 1965. Citado nas páginas 111 e 115.

- GIMENES, P. A.; ROMAN, N. T.; CARVALHO, A. M. Spelling error patterns in brazilian portuguese. **Computational Linguistics.**, MIT Press, Cambridge, MA, USA, v. 41, n. 1, p. 175–183, mar. 2015. ISSN 0891-2017. Citado nas páginas 25, 26 e 27.
- GIMPEL, K.; SCHNEIDER, N.; O’CONNOR, B.; DAS, D.; MILLS, D.; EISENSTEIN, J.; HEILMAN, M.; YOGATAMA, D.; FLANIGAN, J.; SMITH, N. A. Part-of-speech tagging for twitter: Annotation, features, and experiments. In: **Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011. (HLT ’11), p. 42–47. ISBN 978-1-932432-88-6. Citado na página 29.
- GO, A.; BHAYANI, R.; HUANG, L. Twitter sentiment classification using distant supervision. **CS224N Project Report, Stanford**, v. 1, n. 2009, p. 12, 2009. Citado na página 89.
- GOODFELLOW, Y. B. I.; COURVILLE, A. Deep learning. Book in preparation for MIT Press. 2016. Disponível em: <<http://www.deeplearningbook.org>>. Citado nas páginas 33 e 44.
- GOUWS, S.; METZLER, D.; CAI, C.; HOVY, E. Contextual bearing on linguistic variation in social media. In: **Proceedings of the Workshop on Languages in Social Media**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011. (LSM ’11), p. 20–29. ISBN 978-1-932432-96-1. Citado na página 28.
- GREFF, K.; SRIVASTAVA, R. K.; KOUTNÍK, J.; STEUNEBRINK, B. R.; SCHMIDHUBER, J. Lstm: A search space odyssey. **IEEE transactions on neural networks and learning systems**, p. 10, 2016. Citado na página 45.
- HARTMANN, N.; AVANÇO, L.; BALAGE, P.; DURAN, M.; NUNES, M. D. G. V.; PARDO, T.; ALUÍSIO, S. A large corpus of product reviews in portuguese: Tackling out-of-vocabulary words. In: **Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)**. Reykjavik, Iceland: European Language Resources Association (ELRA), 2014. p. 26–31. ISBN 978-2-9517408-8-4. Citado nas páginas 2, 6, 7, 23, 24, 51, 69, 72 e 98.
- HASSAN, H.; MENEZES, A. Social text normalization using contextual graph random walks. **Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)**, p. 1577–1586, 2013. Citado nas páginas 94, 96 e 119.
- HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Journal of Neural Computation**, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. Citado na página 45.
- IKEDA, T.; SHINDO, H.; MATSUMOTO, Y. Japanese text normalization with encoder-decoder model. **Proceedings of the 2nd Workshop on Noisy User-generated Text**, p. 129–137, 2016. Citado nas páginas 15, 19, 41, 64, 65, 66, 67 e 108.
- JOULIN, A.; GRAVE, E.; BOJANOWSKI, P.; MIKOLOV, T. Bag of tricks for efficient text classification. **arXiv preprint arXiv:1607.01759**, p. 5, 2016. Citado na página 34.

- KAPLAN, A. M.; HAENLEIN, M. Users of the world, unite! The challenges and opportunities of Social Media. **Business Horizons**, v. 53, n. 1, p. 59–68, 2010. ISSN 00076813. Citado na página 17.
- KARYPIS, G. Evaluation of item-based top-n recommendation algorithms. In: **Proceedings of the Tenth International Conference on Information and Knowledge Management**. New York, NY, USA: ACM, 2001. (CIKM '01), p. 247–254. ISBN 1-58113-436-3. Citado na página 47.
- KATZ, D. The functional approach to the study of attitudes. **Public opinion quarterly**, AAPOR, v. 24, n. 2, p. 163–204, 1960. Citado nas páginas 13 e 19.
- KATZ, R. Empowerment and synergy: Expanding the community's healing resources. **Prevention in human services**, Taylor & Francis, v. 3, n. 2-3, p. 201–226, 1984. Citado na página 14.
- KIETZMANN, J. H.; HERMKENS, K.; MCCARTHY, I. P.; SILVESTRE, B. S. Social media? Get serious! Understanding the functional building blocks of social media. **Business Horizons**, v. 54, n. 3, p. 241–251, 2011. ISSN 00076813. Citado na página 19.
- KIM, Y. Convolutional neural networks for sentence classification. **Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL**, p. 1746–1751, 2014. Citado na página 90.
- KOHAVI, R. *et al.* A study of cross-validation and bootstrap for accuracy estimation and model selection. **Proceedings of Ijcai**, v. 14, n. 2, p. 1137–1145, 1995. Citado na página 91.
- KRUMM, J.; DAVIES, N.; NARAYANASWAMI, C. User-generated content. **IEEE Pervasive Computing**, v. 7, n. 4, p. 10–11, 2008. ISSN 15361268. Citado nas páginas 1, 10 e 11.
- LAI, S.; XU, L.; LIU, K.; ZHAO, J. Recurrent convolutional neural networks for text classification. **Proceedings of AAI**, p. 2267–2273, 2015. Citado na página 89.
- LANG, K. J.; WAIBEL, A. H.; HINTON, G. E. A time-delay neural network architecture for isolated word recognition. **Neural networks**, p. 23–43, 1990. Citado na página 43.
- LEEMAN-MUNK, S.; LESTER, J.; COX, J. Ncsu_sas_sam: Deep encoding and reconstruction for normalization of noisy text. **Proceedings of the ACL 2015 Workshop on Noisy User-generated Text**, p. 154–161, 2015. Citado nas páginas 19, 41, 63 e 67.
- LEUNG, L. User-generated content on the internet: An examination of gratifications, civic engagement and psychological empowerment. **New Media & Society**, v. 11, n. 8, p. 1327–1347, 2009. ISSN 1461-4448. Citado nas páginas 14 e 15.
- LING, W.; DYER, C.; BLACK, A.; TRANCOSO, I. Two/too simple adaptations of word2vec for syntax problems. **Proceedings of NAACL-HLT**, p. 6, 2015. Citado nas páginas 34 e 101.
- LIU, B. Sentiment analysis and opinion mining. **Synthesis lectures on human language technologies**, Morgan & Claypool Publishers, v. 5, n. 1, p. 1–167, 2012. Citado na página 2.

- LUONG, M.-T.; PHAM, H.; MANNING, C. D. Effective approaches to attention-based neural machine translation. **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**, p. 1412–1421, 2015. Citado na página 64.
- MANNING, C. D.; SCHÜTZE, H. *et al.* **Foundations of statistical natural language processing**. [S.l.]: MIT Press, 1999. v. 1. 620 p. Citado na página 5.
- MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Distributed Representations of Words and Phrases and their Compositionality. **Nips**, p. 1–9, 2013. ISSN 10495258. Citado na página 98.
- _____. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, p. 12, 2013b. Citado nas páginas 33 e 34.
- MIKOLOV, T.; YIH, W.-t.; ZWEIG, G. Linguistic regularities in continuous space word representations. **Proceedings of HLT-NAACL 2013**, p. 746–751, 2013. Citado na página 98.
- MIN, W.; MOTT, B. Ncsu_sas_wookhee: A deep contextual long-short term memory model for text normalization. **Proceedings of the ACL 2015 Workshop on Noisy User-generated Text**, p. 111–119, 2015. Citado nas páginas 15, 19, 41, 61, 63 e 67.
- MITCHELL, T. M. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. 414 p. ISBN 0070428077, 9780070428072. Citado na página 38.
- MORAES, S. M.; MANSSOUR, I. H.; SILVEIRA, M. S. 7x1pt: um corpus extraído do twitter para análise de sentimentos em língua portuguesa. **Proceedings of Symposium in Information and Human Language Technology**, Natal, RN, Brasil, p. 21–25, 2015. Citado na página 3.
- MORAES, S. M.; SANTOS, A. L.; REDECKER, M. S.; MACHADO, R. M.; MENE-GUZZI, F. R. Classificação de sentimentos em nível de sentença: uma abordagem de múltiplas camadas para em língua portuguesa. **Anais do XIII Encontro Nacional de Inteligência Artificial e Computacional**, Recife, PE, Brasil, p. 109–120, 2016. Citado na página 3.
- MUNIZ, M. C. M.; GRAÇAS, M.; NUNES, V.; LAPORTE, E. Unitex-pb, a set of flexible language resources for brazilian portuguese. **Proceedings of the III Information and Language Technology Workshop**, p. 2059–2068, 2005. Citado na página 23.
- OLAH, C. **Understanding LSTM Networks**. 2015. Disponível online em colah.github.io/posts/2015-08-Understanding-LSTMs/. Citado nas páginas 43 e 44.
- OWOPUTI, O.; O'CONNOR, B.; DYER, C.; GIMPEL, K.; SCHNEIDER, N.; SMITH, N. A. Improved part-of-speech tagging for online conversational text with word clusters. **Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**, p. 380–381, 2013. Citado na página 62.
- PANG, B.; LEE, L. Opinion mining and sentiment analysis. **Found. Trends Inf. Retr.**, Now Publishers Inc., Hanover, MA, USA, v. 2, n. 1-2, p. 1–135, jan. 2008. ISSN 1554-0669. Citado na página 2.

- PAUL, D. B.; BAKER, J. M. The design for the wall street journal-based csr corpus. In: **Proceedings of the Workshop on Speech and Natural Language**. Stroudsburg, PA, USA: Association for Computational Linguistics, 1992. (HLT '91), p. 357–362. ISBN 1-55860-272-0. Citado na página [29](#).
- PLANK, B. What to do about non-standard (or non-canonical) language in NLP. **CoRR**, abs/1608.07836, p. 8, 2016. Citado na página [29](#).
- POWERS, D. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. **Journal of Machine Learning Technologies**, v. 2, p. 37–63, 2011. Citado na página [47](#).
- RATNAPARKHI, A. A maximum entropy model for part-of-speech tagging. In: **Proceedings of the Conference on Empirical Methods in Natural Language Processing**. University of Pennsylvania, Philadelphia, PA: Association for Computational Linguistics, 1996. p. 133–142. Citado nas páginas [29](#) e [54](#).
- READ, J.; DRIDAN, R.; OEPEN, S.; SOLBERG, L. J. Sentence boundary detection: A long solved problem? **COLING (Posters)**, v. 12, p. 985–994, 2012. Citado na página [4](#).
- RITTER, A.; CLARK, S.; MAUSAM; ETZIONI, O. Named entity recognition in tweets: An experimental study. In: **Proceedings of the Conference on Empirical Methods in Natural Language Processing**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011. (EMNLP '11), p. 1524–1534. ISBN 978-1-937284-11-4. Citado nas páginas [4](#) e [29](#).
- ROMAN, N. T.; PIWEK, P.; CARVALHO, A. M. B. R.; ALVARES, A. R. Introducing a corpus of human-authored dialogue summaries in portuguese. **International Conference Recent Advances in Natural Language Processing (RANLP 2013)**, p. 692–701, September 2013. Citado na página [25](#).
- RUSELL, R. **US Patent 1261167 issued 1918-04-02**. 1918. Citado nas páginas [37](#) e [52](#).
- SCHUSTER, M.; PALIWAL, K. K. Bidirectional recurrent neural networks. **IEEE Transactions on Signal Processing**, IEEE, v. 45, n. 11, p. 2673–2681, 1997. Citado na página [44](#).
- SHANNON, C. The zero error capacity of a noisy channel. **IRE Transactions on Information Theory**, IEEE, v. 2, n. 3, p. 8–19, 1956. Citado na página [39](#).
- SØRENSEN, T. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. **Kongelige Danske Videnskabernes Selskab**, v. 5, p. 1–34, 1948. Citado na página [37](#).
- SPROAT, R.; BLACK, A. W.; CHEN, S.; KUMAR, S.; OSTENDORF, M.; RICHARDS, C. Normalization of non-standard words. **Comput. Speech Lang.**, Academic Press Ltd., London, UK, UK, v. 15, n. 3, p. 287–333, jul. 2001. ISSN 0885-2308. Citado nas páginas [5](#) e [31](#).

- SRIDHARD, K. R. V. Unsupervised Text Normalization Using Distributed Representations of Words and Phrases. **Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**, p. 8–16, 2015. Citado nas páginas 98 e 99.
- SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, JMLR.org, v. 15, n. 1, p. 1929–1958, jan. 2014. ISSN 1532-4435. Citado nas páginas 43 e 90.
- STIILPEN, M.; MERSCHMANN, L. H. C. A methodology to handle social media posts in brazilian portuguese for text mining applications. **Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web**, ACM, p. 239–246, 2016. Citado nas páginas 13, 15, 19, 55, 57 e 67.
- SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. **Advances in neural information processing systems**, p. 3104–3112, 2014. Citado na página 64.
- TAGLIAMONTE, S. A.; DENIS, D. Linguistic ruin? lol! instant messaging and teen language. **American Speech**, v. 83, n. 1, p. 3–34, 2008. ISSN 0003-1283. Citado na página 27.
- TAI, K.; SOCHER, R.; MANNING, C. Improved semantic representations from tree-structured long short-term memory networks. **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics**, The Association for Computer Linguistics, p. 1556–1566, 2016. Citado na página 40.
- TIAN, T.; DINARELLI, M.; TELLIER, I. Lattice: Data adaptation for named entity recognition on tweets with features-rich crf. **Proceedings of the ACL 2015 Workshop on Noisy User-generated Text**, p. 68–71, 2015. Citado na página 5.
- TILK, O.; ALUMÄE, T. Bidirectional recurrent neural network with attention mechanism for punctuation restoration. p. 3047–3051, 2016. Citado na página 89.
- TOUTANOVA, K.; KLEIN, D.; MANNING, C. D.; SINGER, Y. Feature-rich part-of-speech tagging with a cyclic dependency network. In: **Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003. (NAACL '03), p. 173–180. Citado na página 29.
- TRAMMELL, K.; HOFOMOKL, A.; SAPP, A. Republic of blog: Examining polish bloggers through content analysis. **Journal of Computer-Mediated Communication**, International Communication Association, v. 11, p. 702–722, 2006. Citado na página 12.
- TREVISIO, M. V.; SHULBY, C.; ALUÍSIO, S. M. Sentence segmentation in narrative transcripts from neuropsychological tests using recurrent convolutional neural networks. **Proceedings of EACL**, p. 1–10, 2017. Citado nas páginas 41, 89, 90 e 91.
- TURIAN, J.; RATINOV, L.; BENGIO, Y. Word representations: A simple and general method for semi-supervised learning. **Proceedings of the 49th Annual Meeting of**

the Association for Computational Linguistics, p. 384–394, 2010. Citado na página [33](#).

ULLMAN, J.; AHO, A.; HIRSCHBERG, D. Bounds on the complexity of the longest common subsequence problem. **Journal of the ACM (JACM)**, ACM, v. 23, n. 1, p. 1–12, 1976. Citado nas páginas [37](#) e [94](#).

WANG, T.; CHEN, P.; ROCHFORD, J.; QIANG, J. Text simplification using neural machine translation. **Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence**, p. 4270–7271, 2016. Citado na página [40](#).

WERBOS, P. **Beyond regression: New tools for prediction and analysis in the behavioral sciences**. 906 p. Tese (Doutorado) — Harvard University, 1974. Citado na página [42](#).

WERBOS, P. J. Backpropagation through time: what it does and how to do it. **Proceedings of the IEEE**, p. 1550–1560, 1990. Citado na página [44](#).

ZHANG, C.; BALDWIN, T.; HO, H.; KIMELFELD, B.; LI, Y. Adaptive parser-centric text normalization. p. 1159–1168, 2013. Citado na página [5](#).

MANUAL DO ANOTADOR DO CÓRPUS TUB-UGC

A anotação deste corpus consiste em identificar palavras ruidosas (não reconhecidas pela norma padrão da língua), categorizá-las de acordo com o tipo do ruído e eventualmente corrigi-las. Os textos a serem anotados são provenientes de três fontes distintas da *web*: *tweets*, *reviews* de produtos no Buscapé e *posts* no fórum do UOL. Cada texto foi deixado em caixa baixa dividido em blocos de dez sentenças. Em cada sentença, as palavras ruidosas estão destacadas. A ferramenta UGCNormal foi utilizada para pré-categorizar os tipos de ruído. A anotação manual poderá confirmar ou modificar a classificação prévia, ou ainda adicionar outras anotações.

A [Figura 29](#) apresenta a *interface* principal do sistema de anotação. Cada palavra que precisa ser anotada já vem associada a uma etiqueta que pode ser de duas cores: amarela ou vermelha. Etiquetas amarelas são as pré-definidas pelo UGCNormal. O texto da etiqueta é composto pela abreviação da categoria (NP para Nome Próprio, por exemplo) antecedido por **P-**. Devido a possíveis erros do UGCNormal, essa etiqueta indica que o ruído **possivelmente** pertencente àquela categoria. Nesse caso, seu trabalho é confirmar se a categorização está correta. Etiquetas vermelhas, **Ruído**, não possuem categoria pré-determinada e indicam que essas palavras não pertencem a nenhum lexico usado no pré-processamento. O anotador deve selecionar a categoria mais pertinente. A etapa de pré-processamento identifica a **possível** categoria de cada ruído, associando a ele a etiqueta da categoria. Caso não haja categoria pré-estabelecida, a palavra é marcada simplesmente como **Ruído**.

Escolha qualquer um deles e clique duas vezes para abri-lo. Ao abrir o arquivo, chegaremos à tela ilustrada na Figura 29. O próximo passo é fazer *login* para poder iniciar a anotação. Ao posicionar o ponteiro do *mouse* na parte superior da tela, o menu irá aparecer. Clique no botão **Login** e preencha as informações. O usuário é **nilc** e a senha é **nilc@clin**.

Depois de estar logado no sistema, é possível começar a anotar. Clique na etiqueta associada a determinada palavra para anotá-la. Note que você deve clicar na **etiqueta** e não na palavra em si. Ao clicar, o menu ilustrado na Figura 31 irá aparecer.

Figura 31 – Menu de anotação

Fonte: Elaborada pelo autor.

Todo o processo de anotação acontece nesse menu. O primeiro passo é escolher a categoria. As opções estão dispostas em cor verde, abaixo de **Anotar**. Mesmo que exista uma categoria pré-determinada, você deverá confirmá-la, escolhendo sua versão correspondente em verde. A correção, quando necessária, deve ser inserida na caixa de texto **Notes**. Note que nomes próprios não necessitam de correção e que siglas só precisam ser expandidas quando não foram pré-categorizadas. Após confirmar que tudo está correto, clique em **Ok**. As informações serão salvas e a etiqueta correspondente ficará na cor verde.

A anotação do bloco termina quando **todas** as etiquetas estiverem verdes. Em caso de dúvida entre categorias, opte pela pré-determinada (exceto no caso da categoria **Ruído**, que deve ser alterada). Caso não haja nenhuma pré-determinada, ou seja, é **Ruído**,

e você não sabe como classificá-lo, siga um dos 3 caminhos: (a) pesquise pela palavra nas ferramentas da caixa **Search**, no menu de anotação (ali você pode seguir pelo Google ou Wikipedia); (b) discuta o caso com outros anotadores; ou (c) entre em contato com o responsável pela anotação. Toda etiqueta **Ruído** deve ser substituída por uma das demais. Na caixa **Search** presente no menu de anotação, é possível pesquisar a palavra a ser anotada no Google apenas clicando em **Google**. As ferramentas auxiliares de busca são especialmente úteis para descobrir siglas e nomes próprios desconhecidos pelo anotador.

Ao terminar de anotar um bloco de sentenças, basta clicar na seta no canto superior esquerdo da tela ou pressionar a seta direita no teclado, para passar para o próximo bloco.

A.2 Categorias de Ruído

São consideradas, ao todo, dez categorias diferentes de ruído. Para algumas, uma alternativa ao ruído deve ser fornecida. Pode ser sua correção, a expansão de uma sigla ou de uma abreviatura, etc. Para simplificar, falamos simplesmente em “corrigir” a palavra. Assim, corrigir significa escrever a versão correta da palavra na caixa de texto **Notes** no menu de anotação. Cada categoria será descrita com exemplos nesta seção.

- **Erro Ortográfico (O)**: grafia incorreta de palavras existentes no dicionário que resulta em uma palavra fora do padrão. Em geral, um erro ortográfico corresponde a um erro de digitação – com troca de letras, ausência de acentos etc. Alguns exemplos são: **nao** em vez de **não**, **ecesso** em vez de **excesso**, **desing** em vez de **design** e **tdos** em vez de **todos**. Erros ortográficos **devem ser corrigidos**.
- **Erro Ortográfico Contextual (OC)**: ocorre quando uma palavra válida (presente no dicionário) é utilizada incorretamente no lugar de outra também válida. Em geral é causado por ausência de um único acento. Alguns exemplos são **esta** em vez de **está**, **porem** em vez de **porém** e **noticia** em vez de **notícia**. Para corrigi-lo, é necessário prestar atenção no contexto, pois ambas as formas da palavra (com e sem acento) existem. Além das etiquetas amarelas com categorias pré-determinadas, também existe a etiqueta **Contexto**. Essa marcação indica que aquela palavra é potencialmente um erro ortográfico contextual. Você deve verificar se a palavra está correta no contexto ou se sua versão acentuada deveria ter sido utilizada. Caso esteja errada, selecione a categoria **Ortografico-Contextual** e a corrija. Caso esteja correta, basta selecionar a categoria **Ja-corrigida**.
- **Sigla (SI)**: é uma sequência de letras com o objetivo de abreviar palavras. Lembre-se que todos os caracteres estarão minúsculos, portanto preste atenção no contexto

para identificar siglas. É importante notar que a expansão da sigla deve **sempre** resultar em mais de uma palavra. Alguns exemplos são **ONU** para **Organização das Nações Unidas**, **BBB** para **Big Brother Brasil** e **RJ** para **Rio de Janeiro**. Embora siglas geralmente correspondem a nomes próprios, como os exemplos anteriores, a prioridade é marcá-las como SI. Siglas só precisam ser corrigidas no caso de não terem sido previamente categorizadas como tal. Ou seja, se a etiqueta da palavra que é uma sigla **não** for P-SI, é preciso anotar sua versão expandida como correção, além de optar pela categoria SI. Em caso de dúvida sobre o significado, pesquise ou consulte outros anotadores. Se a versão expandida for realmente desconhecida, apenas marque a categoria SI e não corrija/expandia.

- **Abreviação (AB)**: versão reduzida de uma palavra. Diferente das siglas, as abreviações representam uma **única** palavra. Alguns exemplos são **pq** para **porque**, **tbm** para **também**, **ctz** para **certeza**, **s** para **sim** e **n** para **não**. Note que podem haver diferentes abreviações para a mesma palavra, como **qd**, **qnd** e **qndo** para **quando**. Abreviações **devem** ser corrigidas para sua forma expandida. Caso perceba que é abreviação, mas não sabe de qual palavra, pesquise ou consulte outros anotadores.
- **Internetês (IN)**: gírias e expressões usualmente empregadas na linguagem da Internet. Ruídos são categorizados como IN somente em situações específicas, tais como:
 1. Aumentativos/diminutivos: **moção** para **amor** e **coisica** para **coisa**.
 2. Repetição de caracteres: **lindooo** para **lindo** e **muuuuito** para **muito**.
 3. Gírias e expressões que podem ser transformadas em uma palavra válida: **topzera** para **top**, **ain** para **ai**, **eh** para **é** e **nois** para **nós**.

Ruídos nessa categoria **devem ser corrigidos**. Se ficar em dúvida sobre o significado de alguma palavra, pesquise ou consulte outros anotadores.

- **Estrangeirismo (EN)**: palavras de outros idiomas. Não é necessário corrigi-las, apenas identificar a categoria. Note que, no caso de palavras estrangeiras muito frequentes, erros ortográficos podem acontecer, como em **mouze** em vez de **mouse**. Nesses casos, prefira a categoria Erro Ortográfico e corrija. Ou seja: se a palavra for frequente na língua portuguesa e estiver escrita incorretamente, categorize como **Erro Ortográfico** e corrija; caso contrário, classifique como EN e deixe o campo Notes em branco.

- **Nome Próprio (NP):** nome de pessoa, lugar, entidade etc. Atenção, pois não haverá caixa alta para sinalizá-los. Ruídos desse tipo não precisam ser corrigidos, apenas categorizados. No caso de nomes próprios compostos, cada uma das palavras deve ser anotada como Nome Próprio. Estrangeirismos também se encaixam nessa situação. Um exemplo é **Game of Thrones**, que se refere à série de televisão e portanto cada uma das palavras deve ser marcada como NP.
- **Sem Categoria (SC):** palavras que não ocorrem nos léxicos e que não se enquadram em nenhuma das categorias acima. Palavras aglutinadas (ex. **meunome**) ou separadas (ex. **ba ra to**) encaixam-se nessa categoria. Devem ser anotadas como SC e corrigidas (**meu nome** e **barato**). No caso de sílabas separadas, como em **ba ra to**, cada uma delas deve vir como Ruído, então devem ser selecionadas juntas para receberem uma única categoria (SC).
- **Já corrigido:** palavras que receberam a classe Ruído, provavelmente por incompletude de algum léxico usado na pré-anotação, mas que estão corretas e devem permanecer como estão. Logo, não precisam de qualquer correção, basta mudar a categoria para **Já corrigido**. Essa categoria também é utilizada quando a pré-anotação aponta um possível erro ortográfico contextual, mas, naquele caso, o emprego da palavra está correto.
- **Lixo:** são *tokens* que não podem ser transformadas em palavras reconhecidas pela língua. Risadas (como **kkk** e **hahaha**) e sequências aleatórias de caracteres são exemplos.

A.3 Resumo Passo-a-Passo

Um resumo das etapas que devem ser feitas na anotação.

1. Acesse o sistema em <http://nilc.icmc.usp.br/brat/index.xhtml#/Ruídos/>
2. Acesse sua pasta de anotador
3. Selecione um dos arquivos dentro de sua pasta
4. Faça login no menu que fica no canto superior direito na tela
5. Você deve anotar cada uma das etiquetas amarelas e vermelhas
6. Clique em uma etiqueta para abrir o menu de anotação

7. Selecione a categoria do ruído na **cor verde**
8. Quando necessário, corrija a palavra, digitando sua forma correta na caixa de texto **Notes**
9. Aperte **Ok** para salvar. Ignore os demais botões
10. Quando todas as etiquetas estiverem verdes, o bloco está pronto
11. Passe para o próximo, clicando na seta para direita, e repita
12. Em caso de dúvida, entre em contato com <thalesbertaglia@gmail.com>

A.4 Algumas Definições

Alguns padrões estabelecidos para a anotação.

- Quando **pq** estiver se referindo a **porque**, corrija sempre para **porque**, independente de qual o uso correto da palavra naquele contexto. Ou seja, nada de por que, porquê ou por quê.
- A forma padrão de e-mail é **e-mail**, então email deve ser corrigido como **Erro Ortográfico**.
- Ocorrências de medidas em expressões sem espaço, como **1080p**, **16h20**, **320x480** e **10gb**, aparecerão como Ruídos mas devem ser consideradas como já corrigidas.
- No entanto, unidades de medida isoladas devem ser consideradas **Abreviações** e devem ser corrigidas. Por exemplo, **gb** para **gigabyte**, **h** para **hora**.
- A correção deve ser sempre para uma palavra dicionarizada (verifique nos dicionários online via Google) ou que seja um estrangeirismo (no caso de **topzera** para **top**, por exemplo).