Certografia: um corretor ortográfico automático para português e resultados de um estudo de caso aplicado na área jurídica

Eduardo Pittol[1], Sandro José Rigo[1]

Resumo: As ferramentas desenvolvidas com recursos de Processamento de Linguagem Natural apresentam crescentes possibilidades de apoiar trabalhos em diversas áreas. Uma das atividades em que este tipo de ferramenta pode ser bastante útil é a correção ortográfica. Entretanto ainda são escassos trabalhos desse tipo voltados à língua portuguesa. Neste trabalho, é apresentado o Certografia, um sistema que permite fazer correções ortográficas automáticas em textos de língua portuguesa com o foco em textos da área jurídica. São apresentadas, ainda, as técnicas utilizadas para o seu desenvolvimento, os trabalhos relacionados que apoiaram a sua definição e também a arquitetura utilizado pelo sistema. O protótipo desenvolvido foi avaliado em testes que indicaram preliminarmente resultados bastante promissores.

Palavras-chave: Corretor ortográfico, Processamento de Linguagem Natural.

Abstract: The tools developed in Natural Language Processing field present increasing opportunities to support work in several areas. One of the activities in which this kind of tool can be very useful is spell checking. This work presents Certografia, a system that allows automatic spelling corrections in Portuguese texts with focus on legal area. In this article are commented techniques used for it's development, related works in several languages and system architecture. The developed prototype was evaluated in preliminary tests that showed promising results.

Keywords: Spell Checker, Natural Language Processing.

1 Introdução

Com os avanços na área de Processamento de Linguagem Natural (PLN), surgiu também a necessidade de ferramentas para auxiliar nas diversas atividades envolvidas no processo de entendimento da linguagem [1]. Para que uma informação textual seja processada de forma adequada é preciso que ela esteja descrita de forma correta. No caso de sistemas para PLN, tais como os sistemas de Extração de Informações ou geração de resumos, entre outros, é importante garantir que os dados recebidos estejam corretos e de acordo com a língua que está sendo considerada no processamento. Para isso, antes de fazer o processamento de um texto é de grande valia verificar em que medida ele está escrito de forma apropriada. Os sistemas de computação tratam as palavras a partir de seu armazenamento como dados em formato numérico, onde uma palavra é uma sequência de caracteres codificados em números. Em vários contextos de sistemas de computação, não é possível identificar se a palavra que está sendo processada está escrita corretamente ou não. Para resolver esse problema, é possível aplicar técnicas baseadas em métodos probabilísticos que empregam recursos adicionais, tais como léxicos e dicionários. Utilizando esses métodos é possível obter eficiência em procedimentos dedicados a identificar se uma palavra está escrita corretamente e, caso necessário, indicar qual palavra pode substituí-la.

Esse trabalho possui como motivação tratar o problema de verificação da correção ortográfica em textos da língua portuguesa. Este objetivo está relacionado a um trabalho realizado pelo grupo SEMANTEC (http://projeto.unisinos.br/semantec) da Unisinos (http://www.unisinos.br), que possui como objetivo o estudo e desenvolvimento de ferramentas para o tratamento de informações textuais na área jurídica. Com a ferramenta desenvolvida neste trabalho, os textos tratados neste projeto podem ser analisados de forma mais eficiente e com maior correção. Este trabalho pode ser útil também para analisadores sintáticos, tais como o *parser* Palavras [2],

[1] Programa de Pós-Graduação em Computação Aplicada – PIPCA – Unisinos
São Leopoldo – RS – Brasil
{edpittol@gmail.com, rigo@unisinos.br}

http://dx.doi.org/10.5335/rbca.2015.3776

que enfrentam dificuldades em construir uma árvore de análise sintática de uma frase que apresenta erros ortográficos.

De forma geral, existem dois tipos de erro ortográfico: erros observados por palavras escritas incorretamente ou por palavras posicionadas incorretamente. O primeiro caso, conhecido também como erro por palavra incorreta, ocorre quando a palavra analisada não está presente no dicionário, considerando-se, por exemplo, situações ocasionadas por erros de grafia das palavras. O segundo caso, denominado por vezes de erro por palavra real, ocorre quando a palavra está escrita corretamente, porém não é a palavra que deveria estar naquela posição no texto. Já um sistema de correção ortográfica pode ser separado em quatro categorias, comentadas a seguir. A primeira é a categoria denominada "Não confiável". Este é o nível mais simples existente entre os corretores ortográficos, pois seu funcionamento pode ser resumido como um processo de análise de um texto de entrada, que então identifica ocorrências de palavras incorretas, sendo que para tal é utilizado um dicionário que permite verificar a presença ou não de cada palavra neste dicionário. O Segundo tipo de corretor é o denominado de "Menos confiável", que além de verificar a existência de cada palavra do texto em um dicionário, irá retornar uma lista com as palavras que podem substituir a palavra identificada como incorreta. Para gerar essa lista, em geral esse tipo de sistema utiliza algoritmos como o de Distância Mínima de Edição, apresentado na seção 3. Já o tipo de corretor denominado "Mais confiável", vai também identificar e sugerir a melhor correção a ser adotada para a palavra errada. Para chegar ao resultado, além de utilizar as técnicas acima citadas, esse tipo de sistema utiliza o modelo de linguagem N-gram, descrito na seção 3. Por fim, os sistemas denominados "Muito confiáveis em correção" são os corretores que têm autonomia em corrigir textos automaticamente. Eles utilizam todas as mesmas técnicas citadas acima, porém devem garantir que está sendo realizada a substituição da palavra incorreta pela palavra correta.

Para este trabalho foi escolhida a abordagem denominada como "Muito confiáveis em correção", com o objetivo de proporcionar o desenvolvimento de uma ferramenta adequada para utilização em situações em que um grau alto de correção é necessário. As técnicas utilizadas foram também validadas com estudos de trabalhos relacionados e foi definido como tema de estudo e validação o tratamento dos textos em área jurídica. Os resultados obtidos foram validados e considerados bastante promissores, atendendo aos objetivos iniciais definidos para este trabalho. O software gerado, juntamente com o seu código fonte, foi disponibilizado para utilização de forma cooperativa, sendo que os dados de acesso estão descritos na seção 5.

O presente texto está organizado da seguinte forma. Na seção 2, são apresentados elementos de Linguística Computacional. Na seção 3, são descritas abordagens matemáticas para a resolução de problema de correção ortográfica. Na seção 4, são apresentados trabalhos relacionados juntamente com um comparativo entre características de corretores criados para outras línguas e o sistema desenvolvido. Na seção 5, são descritas as abordagens e as técnicas utilizadas para desenvolver o corretor Certografia, sendo apresentada sua arquitetura, o plano de desenvolvimento, o protótipo e os testes de validação realizados. Por fim, a seção 6 apresenta as conclusões e indicações de trabalhos futuros.

2 Linguística Computacional

Segundo Othero e Menuzzi [3], a linguística computacional é a área da ciência linguística voltada para o tratamento computacional da linguagem e das línguas naturais. Seus primeiros esforços iniciaram nos anos 1950. É uma área relativamente nova em relação à ciência Linguística, visto que os primeiros estudos linguísticos no Ocidente começaram há 2400 anos com gramáticos e filósofos Gregos e Romanos. Já, no oriente, os estudos começaram ainda antes, como no caso da Índia, onde os estudos gramaticais iniciaram há 2500 anos.

Alguns trabalhos são considerados como fundamentais para a estruturação desta área, como é o caso dos trabalhos dos autores Frege [20], do linguista norte-americano Noam Chomsky [19] e do linguista Richard Montague [18]. Entretanto, este pode ser considerado um progresso lento. O grande impulso da Linguística Computacional ocorreu com o surgimento da Inteligência Artificial [5]. Isso ocorreu durante as décadas de 1950 e 1960 com o desenvolvimento de programas de tradução automática. Desde o início do desenvolvimento dos computadores já se viu que eles possuíam potencial para processar a linguagem natural. No final dos anos 1970, a área teve um grande crescimento. Com a evolução da Inteligência Artificial e da Linguística Computacional, essa área tem estimulado a motivação para novas pesquisas, principalmente na melhoria de aspectos da Interação Homem Computador (IHC) e no desenvolvimento de softwares voltados para os diversos usos da linguagem natural.

Na área da fonética e da fonologia, encontramos muitos aplicativos de Processamento de Linguagem Natural. A fonética se ocupa com os sons e a fonologia com os fonemas e o sistema fonológico. Os desenvolvimentos observados nessas áreas estão direcionados para o reconhecimento de fala, síntese de fala e sistemas de diálogo em língua falada. Já os estudos em sintaxe e semântica são de grande importância para o desenvolvimento de programas na área de Processamento de Linguagem Natural. A sintaxe é o estudo das regras e dos constituintes das frases. A semântica é o estudo do significado das palavras e das proposições de sua utilização. Esses estudos são importantes para apoiar o desenvolvimento de sistemas que precisam compreender e/ou gerar frases. Os sistemas conhecidos como *Chatterbots* são um exemplo de programas que utilizam essas técnicas. Eles são programas que interagem com humanos por meio de diálogo em linguagem natural.

Outro tipo de aplicação que se utiliza de recursos de Processamento de Linguagem Natural é a tradução eletrônica. Essas aplicações têm como objetivo fazer a tradução automática de textos e de sentenças de uma língua para outra. Essa classe pode ser dividida em três tipos de programas. Os gerenciadores de terminologia auxiliam na tradução de textos técnicos de determinada área. As ferramentas de tradução automática traduzem qualquer tipo de documento, sem restrição. Alguns programas auxiliam um tradutor humano na tradução de um texto. Essa última categoria não se preocupa com a perfeição, mas auxilia na tradução de grande quantidade de textos. Acredita-se [3] que o uso de tais programas pode ajudar um tradutor humano a reduzir em 50% o tempo total gasto no trabalho de tradução de 15% até 30% no custo total com a tradução.

O conhecimento em sintaxe e semântica e sua aplicação na computação também são fundamentais para criação de aplicações tais como *parsers*, geradores automáticos de resumos, corretores ortográficos e gramaticais, classificadores automáticos de documentos digitais, entre outros. O foco desse trabalho está no desenvolvimento de corretores ortográficos e gramaticais.

3 Aprendizado de máquina e modelos de linguagem

Nesta seção são destacados alguns aspectos de elementos envolvidos no desenvolvimento do corretor ortográfico apresentado. Inicialmente são destacados aspectos da área de aprendizagem de máquina e de modelos utilizados para representar e processar textos, bem como recursos de manipulação probabilística e cálculos de distância mínima entre palavras.

Segundo Xue e Zhu [4] o aprendizado de máquina é o estudo de como o computador simula ou realiza o estudo do comportamento do ser humano. O objetivo é obter o novo conhecimento ou habilidade e organizar a estrutura de conhecimento, o que pode ser feito com uma melhoria progressiva de sua própria performance. Dentre várias outras aplicações conhecidas, sistemas de Aprendizagem de Máquina podem ser utilizados para inferir um modelo de classificação em um *corpus* com classes conhecidas. Aplicações de Mineração de Dados utilizam essa técnica para filtrar informações automaticamente e apresentar ao usuário apenas dados de seu interesse. Neste trabalho, é empregado um processo de aprendizagem de máquina para que a ferramenta possa consultar em uma base de dados informações coletadas de outros textos. Então, a partir desses textos é possível calcular e deduzir a palavra que irá substituir a palavra errada.

Os modelos de linguagem podem ser definidos, segundo Jurafsky e Martin [5], como modelos estatísticos de sequências de palavras. Por exemplo, um modelo denominado N-gram usa a palavra anterior, na posição N-1, para predizer a ocorrência da próxima palavra. A tarefa de predizer qual será a próxima palavra de uma frase é essencial para a detecção de erros ortográficos. A partir da observação das palavras anteriores, esse tipo de abordagem permite a obtenção de uma previsão das palavras que possam vir a seguir. Utilizando-se, por exemplo, o trecho da frase "Eu retirei um ...", a próxima palavra pode ser “livro” ou “carro”, dentre outras possibilidades. Porém a próxima palavra normalmente não pode ser “casa” ou “professor”, por exemplo. A partir dessa lógica, ao ser conhecida a composição habitual de sequência de palavras e suas probabilidades de ocorrência, existe a possibilidade de prever qual palavra é provavelmente a mais correta, de acordo com o trecho da frase que a precede ou sucede.

Os sistemas atuais baseados em N-gram utilizam como elemento fundamental a forma da palavra, que é a forma flexionada com a qual ela aparece no corpus. Por exemplo, as palavras “casa” e “casas” são tratadas como palavras diferentes nesses sistemas. Porém, isso pode não ser adequado para domínios simplificados que precisam tratar essas duas situações como sendo uma mesma, sendo esta obtida a partir do uso do lema da palavra “casa“. De acordo com o tamanho do corpus, quando as palavras são tratadas apenas com a forma flexionada, existe a chance de ser gerada uma probabilidade muito pequena para cada ocorrência de palavra

seguinte. Para encaminhar uma solução para este problema, são adotadas abordagens como as descritas a seguir. A equação (1) representa a probabilidade P de uma sequência completa de palavras (w_1, w_2, ...w_n).

$$P(w_1, w_2, ..., w_{n-1}, w_n) \tag{1}$$

Para decompor a equação anterior (equação 1) pode ser utilizada a regra da Cadeia de Markov [6], sendo o seu resultado descrito na equação (2).

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)...P(w_n|w_1^{n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \end{aligned} \tag{2}$$

A equação (2) pode não ser facilmente computada se o *corpus* a ser tratado tiver uma imensa quantidade de palavras, pois seria necessário calcular todas as probabilidades das palavras anteriores. Para resolver esse problema, a abordagem N-gram utiliza o cálculo sobre uma aproximação da situação real do *corpus*. Essa aproximação é dada sobre um número N de palavras anteriores à palavra para a qual se deseja calcular a probabilidade. Assim, no modelo bigram a aproximação é feita utilizando apenas a palavra anterior. Então, ao invés da probabilidade condicional de todas as palavras anteriores, é considerada somente a probabilidade da palavra imediatamente anterior. Tomando por exemplo a frase "João foi passear no parque", a probabilidade de ocorrência da palavra "parque" nesta frase poderia ser representada como P(parque | João foi passear no), que indica a necessidade de cálculo para a probabilidade da palavra "parque" junto com cada uma das palavras anteriores. Já no modelo bigram, a situação definida para esta mesma frase e palavra pode ser representada como P(parque | no), onde pode ser observada a indicação de cálculo entre a palavra "parque" e apenas mais uma palavra (a palavra "no").

Essa abordagem é baseada na suposição de Marvov [6]. Baseando-se no formalismo de uma cadeia de Markov pode-se considerar cada palavra de um determinado texto como sendo um estado. Segundo esse formalismo, um modelo bigram é chamado de um modelo de primeira ordem, enquanto um modelo trigram é um modelo de segunda ordem. Por fim, um modelo N-gram calcula a probabilidade de uma palavra ser a próxima a ocorrer utilizando N−1 palavras anteriores. Quanto maior for a ordem do modelo, maior o custo computacional para calcular e recuperar a probabilidade de ocorrência da palavra. A equação geral para a probabilidade condicional é a dada pela equação (3), descrita a seguir.

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1}) \tag{3}$$

Segundo Manning e Shutze [7], *tagging* é a tarefa de rotular cada palavra em uma sentença com sua apropriada "[...] parte do discurso. ... cada palavra é um pronome, verbo, adjetivo, ou outra coisa". O *tagging* é um processamento feito sobre uma sentença, sendo que esse processamento adiciona um conjunto de rótulos às palavras. Nesses rótulos estarão descritas informações gramaticais sobre cada uma das palavras analisadas. Com base em uma análise já realizada em um corpus com um processo de *tagging*, é possível treinar este corpus utilizando o modelo de Markov. O resultado desta modelagem pode ser expresso pela equação (4), descrita a seguir.

$$P(w^l|t^j) = \frac{C(w^l, t^j)}{C(t^j)} \tag{4}$$

Na equação (4), o termo $P(w_l \mid t_j)$ representa a probabilidade da palavra w possuir marcação (*tag*) t. O termo $C(w_l, t_j)$ representa o número de ocorrências em que w é marcado por t. Por sua vez, o termo $C(t_j)$ descreve o número de ocorrências da marcação j. Para encontrar o melhor formato de marcação $t_{1,n}$ para a sentença $w_{1,n}$ é aplicado o teorema de Bayes [4]. Como resultado é obtida a seguinte equação:

$$P(w_{1,n}|t_{1,n}) = \prod_{i=1}^{n}[P(w_i|t_i) \times P(t_i|t_{i-1})] \quad (5)$$

Depois dessa fase, o algoritmo pode ser considerado como treinado para o *corpus* em questão. Após esta operação é necessário marcar as sentença de entrada. Existem dois métodos para esse fim. Um método é o de Jelinek (descrito pela equação 6) e o outro é o de Kupiec (descrito pela equação 7).

$$b_{j,l} = \frac{b^*_{j.l} C(w^l)}{\sum w^m b_{j.m} C(w^m)} \quad (6)$$

$$b_{j.L} = \frac{b_{j,L} C(u_L)}{\sum u_L b_{j.L'} C(u'_L)} \quad (7)$$

No caso do método de Jelinek, assume-se que a probabilidade de uma palavra ocorre igualmente com cada marcador (*tag*) possível, sendo que deste modo a soma que define a probabilidade é calculada sobre todas as palavras w_m com os seguintes critérios (8) :

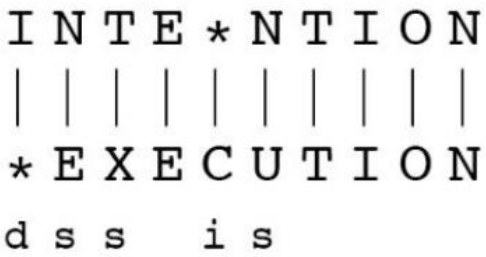
$$b_{j.l} = \begin{cases} 0 & \text{se } t^j \text{ não é POS permitido por } w^l \\ \dfrac{1}{T(w^l)} & \text{caso contrário} \end{cases} \quad (8)$$

Já no caso do método de Kupiec, é realizado um agrupamento de palavras com a mesma classificação gramatical em uma meta-palavra uL, sendo que, neste caso, L é definido como um subconjunto de inteiros, variando de 1 até o número de marcadores permitidos para w. Desse modo a soma no denominador dessa equação é realizada sobre todas as meta-palavras uL, com os seguintes critérios (9) :

$$b_{j.L} = \begin{cases} 0 & \text{se } j \notin L \\ \dfrac{1}{|L|} & \text{caso contrário} \end{cases} \quad (9)$$

Segundo Wagner e Fischer [9], o cálculo de distância mínima de edição é baseado nas três operações de Morgan. As três operações são: substituição de um caractere por outro; deleção de um caractere; inserção de um caractere dentro da *string*. A distância será o número necessário de operações para transformar uma *string* em outra. As operações podem ser representadas de três maneiras: traços, alinhamento e lista de operação. Na Figura 1 há um exemplo aplicado no cálculo da distância mínima entre a palavra intenção e a palavra execução. A Figura 1 mostra, portanto, a operação utilizando o alinhamento entre os caracteres da palavra.

Figura 1: Distância Mínima de Edição exibida por alinhamento. Fonte: Jurafsky e Martin [5]



A informação descrita na Figura 1 deve ser interpretada da seguinte forma. Nos traços, a falta de ligação entre um caractere representa uma exclusão ou uma inserção. No caso de uma associação entre duas letras diferentes, o traço representa uma substituição. No alinhamento, a ocorrência de um caractere "*" na linha de cima representa uma inserção e na linha de baixo representa uma exclusão. Já um caractere diferente em cima e embaixo representa uma substituição. Na lista de operações que pode ser vista na última linha da Figura 1 são identificadas as operações que são realizadas e seu resultado.

Em [10] o autor propõe que cada operação de inserção ou deleção tenha custo de valor igual a um e que operações de substituição não sejam permitidas. O que acontece é que uma substituição é o resultado de uma inserção e uma deleção, portanto ela possui custo dois. Para computar a distância mínima é usado um algoritmo de programação dinâmica no qual é utilizada uma matriz com uma coluna para cada símbolo da *string* fonte e uma linha para cada símbolo da *string* de alvo. Essa matriz é chamada de *edit-distance* e cada célula é representada por *edit-distance* [i, j] e contém a distância entre o primeiro caractere *i* do alvo e o primeiro caractere *j* da fonte. O valor de cada célula é o caminho mais curto existente. A organização abaixo (equação 10) mostra a estrutura para identificação dos melhores caminhos possíveis.

$$P(t|c) = min \begin{cases} edit\text{-}distance[i-1,j] + ins\text{-}cost(target_i) \\ edit\text{-}distance[i-1,j-1] + subst\text{-}cost(source_j, target_i) \\ edit\text{-}distance[i,j-1] + del\text{-}cost(source_j) \end{cases} \quad (10)$$

A matriz sempre inicia com valores da célula *edit-distance* [0, 0] = 0. O primeiro caminho é o valor da coluna anterior mais o custo de inserção. O segundo é valor da linha e coluna anteriores mais o custo da substituição. E o terceiro é o valor da linha anterior mais o custo da deleção. O resultado é o valor do último elemento da matriz.

4 Trabalhos Relacionados

Foram realizados estudos sobre sistemas com propósito similar, de modo a identificar suas principais características e, dessa forma, auxiliar nas decisões deste trabalho. A maioria dos trabalhos estudados apresenta como proposta a resolução de uma das duas categorias de problemas de ortografia. A primeira categoria busca o tratamento das chamadas "pseudopalavras", que são palavras que não são encontradas no léxico da linguagem. A segunda categoria consiste no tratamento das denominadas "palavras verdadeiras", que são palavras que existem no léxico, porém não estão sendo utilizadas apropriadamente no texto. Esses trabalhos também utilizam em geral alguma forma de *corpus* treinado, que é um conjunto de textos analisado em sua composição e correção ortográfica e a partir do qual são calculadas as probabilidades sobre as palavras desse. Geralmente, para calcular essas probabilidades, os trabalhos estudados utilizam o modelo bigram ou trigram. A seguir, são descritos maiores detalhes dos trabalhos destacados como de maior interesse e também são agrupados de forma resumida suas características, para fins de comparação.

Em [11], é apresentada uma proposta de um corretor totalmente automático para a língua inglesa. Para desenvolvê-lo, foi utilizado o modelo trigram e o analisador morfossintático. Para reduzir o custo computacional, é utilizado o método POS trigram, que faz a união entre o modelo trigram com o analisador morfossintático. Porém em situações em que as palavras candidatas possuem a mesma classificação gramatical, utiliza-se somente o modelo de linguagem trigram. Este trabalho também utiliza a distância mínima de edição para fornecer a palavra mais provável para a substituição da palavra errada. No trabalho de Delden, Bracewell e Gomez [12] é apresentado um algoritmo que avalia por etapas qual é a lista de palavras candidatas para substituir a palavra errada. No final, o algoritmo indica a palavra mais provável a substituir a palavra errada. Durante as etapas deste algoritmo, são utilizadas a distância mínima de edição de forma reversa, o modelo bigram e é também avaliado o número de ocorrências da palavra no *corpus*.

Em [13], é apresentado um experimento para correção de textos na língua vietnamita. Além de usar os tradicionais bigram e analisador morfossintático, ele utiliza o algoritmo SoundEX. O algoritmo é utilizado para detectar erros em relação a fonemas. Traçando um paralelo com a frases faladas em português, podemos tomar como exemplo os sons de "x" e "ch". Em [14] é apresentado um corretor automático para a língua coreana no qual os autores classificam os textos em duas categorias: língua falada e língua escrita. Estas duas categorias são diferenciadas pelo seu formalismo, sendo considerada a primeira categoria menos informal, como no caso de

mensagens curtas de texto e postagens em blogs. A solução foi desenvolvida para textos da língua falada. Na língua coreana cada sílaba é representada por um símbolo, portanto as regras de correção são aplicadas aos símbolos. Essas regras são armazenadas em um banco de dados que é consultado pelo corretor, sendo que nesse banco de dados são armazenadas tuplas compostas pelo contexto anterior, o símbolo que deve ser trocado, contexto posterior e o símbolo correto. Para avaliar a acurácia de uma regra foi utilizada a fórmula de Bayes.

Em [15] é apresentado um corretor automático para a língua indonésia. Na arquitetura apresentada pelos autores destaca-se um analisador morfológico que separa a palavra em morfemas. Assim, o sistema pode detectar o erro na palavra com base em um dicionário. Após a detecção do erro, são escolhidos candidatos por meio da distância mínima de edição. Sobre essa lista é aplicado o modelo de Markov e o modelo de linguagem bigram.

Em [16], é apresentado um corretor automático para língua tailandesa, para melhorar os resultados de reconhecimento ótico de caracteres. Uma peculiaridade dessa língua é que as palavras não são separadas por espaço, o que dificulta a separação delas no seu reconhecimento. Para detectar os limites entre as palavras é utilizado um algoritmo que busca por tokens dentro de um dicionário. Após a consulta, esse processo gera como resultado um grafo com as possíveis palavras que formam uma sentença. Nos testes realizados com o modelo trigram, dependendo do assunto do documento, o sistema pode ajudar na correção, mas em alguns casos o sistema não é adequado e pode introduzir mais erros do que os existentes.

A Tabela 1 apresenta um comparativo aos trabalhos relacionados acima. Pode-se perceber que os trabalhos realizados sobre línguas estilo ocidental utiliza um tipo de modelo de linguagem, análise morfossintática e fazem uso do cálculo de distância mínima. Enquanto que as línguas orientais, por apresentarem uma estrutura diferente, não utilizam essas técnicas.

Tabela 1: Comparativo entre corretores ortográficos de diversas línguas

Trabalho	Língua	Modelo	Marcação	Distância Mínima
RUCH; BAUD; GEISSBUHLER, 2001 [11]	Inglês	POS Trigram	Sim	Sim
DELDEN; BRACEWELL; GOMEZ, 2004[12]	Inglês	Trigram	Sim	Sim
NGUYEN et al., 2008 [13]	Vietnamita	Bigram	Sim	Sim
BYUN; RIM; PARK, 2007 [14]	Coreano	Não	Não	Não
ALKANHAL et al., 2012 [15]	Indonésio	Não	Não	Não
RODPHON et al, 2001 [16]	Tailandês	HMM Bigram	Sim	Sim

5 Sistema desenvolvido

Neste capítulo, são descritos o modelo e a implementação do corretor ortográfico, bem como os testes de avaliação. Foi proposta uma arquitetura completa, porém simples, para desenvolver o corretor ortográfico. Alguns componentes da arquitetura empregam o reuso de bibliotecas disponíveis em regime de código aberto. Dessa forma, o foco do desenvolvimento deste trabalho ficou concentrado no desenvolvimento das regras de correção dos textos.

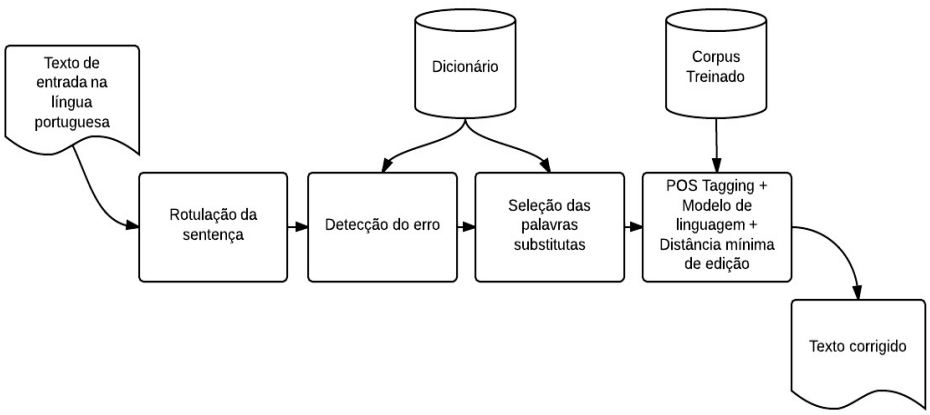
5.1 Arquitetura e componentes

A Figura 2 mostra os principais componentes da arquitetura da ferramenta desenvolvida, que é composta por uma base de dados com dicionário de palavras, um módulo de rotulação das sentenças, o módulo seletor de possíveis palavras substitutas, um componente que implementa um filtro para marcação morfossintática e tratamento do modelo de linguagem, além da base de dados para armazenar resultados do treino.

O dicionário tem como objetivo armazenar o maior número possível de palavras do léxico, para permitir que sejam identificadas as palavras dos textos analisados como pertences ou não ao léxico. Os dados desse dicionário ficam armazenados em um banco de dados. Para a coleta das palavras do dicionário, foi utilizado o conjunto dos dados do projeto Vero (http://www.broffice.org). Esse projeto é mantido pela comunidade vinculado ao Software Livre e é base para a correção dos textos das versões portuguesas do LibreOffice (http://www.libreoffice.org/). Nesse banco de dados, são armazenadas as palavras e sua classe gramatical. A

informação da classe gramatical é importante, pois após o processo de marcação gramatical podemos selecionar palavras próximas e com a mesma classe, reduzindo o número de palavras candidatas.

Figura 2: Arquitetura do corretor ortográfico



Para ajudar no processo de identificação da palavra correta para a substituição, as sentenças devem passar pelo analisador morfossintático PALAVRAS [2]. Ele irá fazer a rotulação da sentença, gravando suas informações gramaticais. Depois da descoberta de uma palavra que não se encontra no léxico, é então gerada uma lista de palavras que podem substituir a palavra incorreta. Essa listagem é feita utilizando a técnica de Distância Mínima de Edição, descrita anteriormente. Para montar essa lista é novamente consultado o dicionário armazenado no banco de dados léxico. O grande desafio nessa tarefa é otimizar a criação dessa lista. Visto que o dicionário possui um número grande de palavras, é necessário um algoritmo específico que crie atalhos para encontrar essas palavras, tornando assim a geração da lista eficiente.

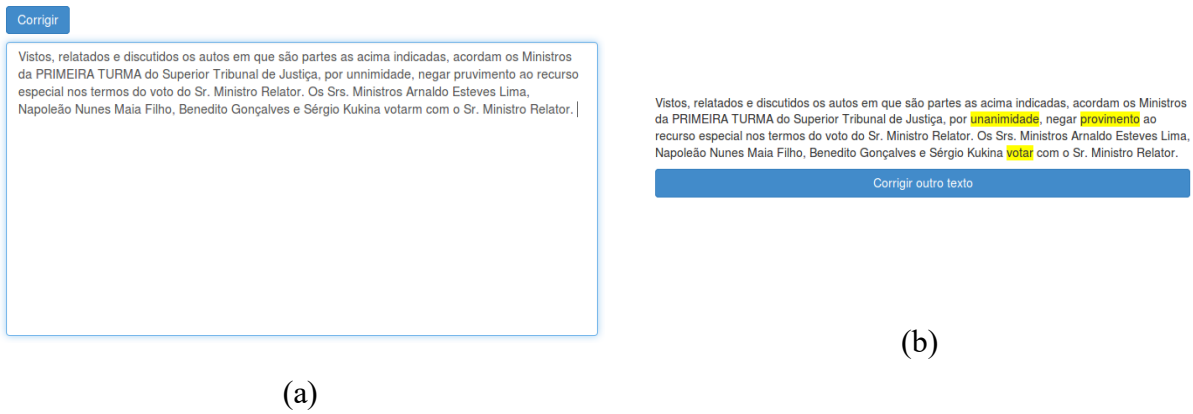
Para seleção da palavra substituta é utilizada uma combinação do filtro para marcação morfossintática e um modelo de linguagem. O filtro para marcação morfossintática definido serve para realizar uma comparação entre a classe gramatical da palavra incorreta com as classe gramatical das possíveis palavras substitutas. O modelo de linguagem escolhido para trabalhar com o corretor foi o bigram. Esta escolha se deve ao curto período disponível para o desenvolvimento da ferramenta e aos bons resultados possíveis de obter com este modelo. Após passar por esse processo, é escolhida a palavra que será utilizada na substituição da palavra incorreta. São realizados testes para verificar a importância de cada técnica para gerar a palavra substituta.

5.2 Interface

A interface do ambiente pode ser acessada por qualquer um que possua conexão com a Internet. O acesso se dá através de um browser. A interface recebe o texto a corrigir e apresenta a correção de palavras digitadas pelo usuário. O usuário basicamente informa um texto ou uma palavra em um campo de texto e depois seleciona a operação de correção. A interface destacará, através do uso de cores, as palavras que foram corrigidas.

No primeiro cenário, o usuário interage com uma caixa de texto e um botão "Corrigir". O sistema substitui as palavras incorretas pelas escolhidas para permuta e as destaca visualmente. O usuário pode clicar com o botão direito do mouse sobre uma palavra destacada e ver, em uma lista, como ele havia escrito a palavra originalmente. A Figura 3 exibe um texto com erros no item “a”, seguido pelo resultado da correção, no item “b“.

Figura 3: Exemplo da interface do corretor ortográfico e seus resultados



(a) (b)

5.3 Desenvolvimento

O desenvolvimento da ferramenta foi dividido em três etapas: treino, implementação e teste. A primeira etapa do desenvolvimento do corretor é bastante importante, pois por meio dessa é construída toda a base de dados para a verificação das palavras. Primeiramente foram importadas para o banco de dados as palavras do dicionário do projeto Vero, juntamente com a realização da transformação do formato que ele utiliza para tratar as classes gramaticais de modo que essas fiquem adequadas ao formato utilizado pelo *parser* morfossintático PALAVRAS. O processo inicial de treino foi realizado sobre um corpus anotado, originado na área de foco do estudo de caso definido, que foi a área do Direito. Para a anotação do *corpus* foi utilizado o *parser* morfossintático PALAVRAS, tendo sido utilizado o modelo bigram como modelo de linguagem, conforme comentado anteriormente. Posteriormente a essa etapa foi realizada a implementação da principal parte do trabalho.

Depois da base de dados estar pronta, a próxima etapa envolveu preparar a ferramenta para corrigir os textos de forma automática. Para realizar essa tarefa, o primeiro passo foi implementar a detecção das palavras que estão incorretas, através de consultas ao dicionário. Quando uma palavra incorreta é encontrada, uma consulta é realizada na base de dados com palavras que possam substituir as palavras incorretas. Para isso, foi utilizado o algoritmo de distância mínima de edição, que gera uma lista de possíveis palavras para substituir a palavra que está incorreta. Juntamente com essa lista, são utilizadas como apoio na verificação as probabilidades armazenadas na fase de treino. Para isso, é consultado o banco de dados de modelo de linguagem gerado na fase de testes.

Após toda a segunda etapa estar implementada, foram realizados os testes de eficiência da ferramenta. Os textos para testes são processados com o software PALAVRAS. Os relatórios dos testes são armazenados em um banco de dados e podem ser visualizados por meio páginas em formato HTML.

Para desenvolvimento do protótipo foi utilizada a linguagem de programação Python (http://www.python.org) com as bibliotecas pyHunspell (http://code.google.com/p/pyhunspell) e *Natural Language Toolkit* (NLTK) (http://www.nltk.org). A escolha da linguagem de programação Python ocorreu devido a NLTK ser escrita nessa linguagem. Como banco de dados, para armazenamento do modelo de linguagem foi utilizado o sqlite. A NLTK [17] é um projeto *open-source* que foi desenvolvido na Universidade da Pennsylvania em 2001. Ela é um conjunto de módulos e corpora utilizados para aprendizado e pesquisa em Processamento de Linguagem Natural. Neste trabalho, ela foi utilizada para realizar a tokenização das palavras e também para efetuar o cálculo do modelo de linguagem bigram.

A Hunspell é uma biblioteca utilizada para a verificação ortográfica escrita em C++, também *open-source*. Entre suas principais características, além da verificação ortográfica, estão a análise morfológica e a separação do radical (stemming) das palavras. A pyHunspell é uma biblioteca alternativa em Python para uso da Hunspell. Essa biblioteca foi utilizada por suas facilidades de acesso ao dicionário do VERO. A escolha do sqlite (http://www.sqlite.org) se deu pelo motivo de ser um Sistema de Gerenciamento de Base de Dados leve e fácil de implementar e além disso, a linguagem Python oferece uma biblioteca nativa para seu gerenciamento. Para a

interface web, foram criadas duas views, uma para inserção do texto a ser corrigido e outra para exibição do resultado. A implementação foi feita utilizando o framework Django (http://www.djangoproject.com).

5.4 Testes Realizados

Os testes de avaliação inicial foram realizados sobre quatro jurisprudências. Os dois primeiros documentos foram retirados do *corpus* de treino e os seguintes do portal JusBrasil (http://www.jusbrasil.com.br). Para cada arquivo foram realizados três testes. O primeiro teste foi realizado com o conteúdo do arquivo que está correto. No segundo teste, foram modificadas dez palavras. Já no terceiro teste foram modificadas vinte palavras.

A mensuração da eficiência do sistema, detalhada na Tabela 2, foi realizada utilizando as medidas *precision* (equação 11), *recall* (equação 12) e *F-measure* (equação 13), sendo que seus resultados são indicados em uma faixa numérica na qual o valor zero representa o pior resultado e o valor 1 representa o melhor resultado. O resultado dessas medidas é obtido a partir das seguintes definições de seus termos: Verdadeiro-positivo (tp), quando o sistema classificou a palavra como correta e realmente está correta; Verdadeiro-negativo (tn), se o sistema classificou a palavra como correta e ela está incorreta; Falso-positivo (fp), quando o sistema classificou a palavra como incorreta e ela está incorreta; Falso-negativo (fn) no caso do sistema classificar como incorreta e a palavra que está correta.

$$Precision = \frac{tp}{tp + fp} \tag{11}$$

$$Recall = \frac{tp}{tp + fn} \tag{12}$$

$$F\text{-}measure = 2 \times \frac{precision \times recall}{precision + recall} \tag{13}$$

Os resultados foram obtidos com o uso do sistema desenvolvido e posteriormente analisados por um especialista em linguística, que realizou a confirmação dos resultados verificados. A Tabela 2 exibe um resumo dos resultados obtidos, sendo que podem ser destacadas a informação da quantidade de palavras (na coluna "Palavras"), a quantidade de palavras que foram modificadas (na coluna "Mod."). As colunas que expressam os tempos de processamento e as métricas de precisão (*precision*, na coluna "P"), cobertura (*recall*, na coluna "R") e a media-f (*F-measure*, na coluna "F") indicam a configuração dos resultados positivos obtidos. Já as colunas correção utilizando Distância Mínima de Edição ("Dist.Mín.") e o Modelo de Linguagem Bigram ("Big.") foram acrescentadas para detalhar mais a quantidade de palavras que foram identificadas com cada um destes recursos.

Tabela 2 – Resultado dos testes realizados

Doc.	Teste	Palavras	Mod.	Tempo (ms)	Dist. Mín.	Big.	P	R	F
1	1	5659	154	57205	121	26	0.979	0.990	0.985
1	2	5659	163	58265	122	34	0.977	0.990	0.984
1	3	5659	172	70488	127	38	0.975	0.990	0.983
2	1	3157	171	59561	154	14	0.973	0.964	0.969
2	2	3157	180	63382	155	22	0.970	0.964	0.967
2	3	3157	187	65371	161	23	0.967	0.964	0.966
3	1	1880	48	13810	38	8	0.992	0.978	0.985
3	2	1880	59	19909	42	15	0.987	0.978	0.982
3	3	1880	67	24027	49	16	0.982	0.977	0.980
4	1	1299	30	8058	21	2	0.991	0.982	0.987
4	2	1299	39	13897	25	7	0.983	0.982	0.982
4	3	1299	54	18459	33	14	0.969	0.982	0.975

Para execução dos testes, foi utilizado um computador com as seguintes configurações: processador AMD Turion(tm) X2 Ultra Dual-Core Mobile ZM-82 com 500MHz e quantidade de memória RAM igual a 3699 MB.

6 Conclusão e trabalhos futuros

O desenvolvimento de um corretor ortográfico é uma tarefa grande e complexa. No contexto da língua natural, torna-se difícil definir uma fórmula precisa que indique com exatidão se uma palavra está errada e se a palavra substituta é a que melhor se encaixa no contexto de determinada frase. Por isso, são utilizados métodos probabilísticos para que seja possível aproximar a escolha da palavra mais provável a substituir a palavra errada.

Devido a esse assunto ser relativamente novo, não foram encontrados muitos materiais para a realização dessa pesquisa, mas foi possível conhecer e estudar alguns corretores para outras línguas. Foi detectado que todos os corretores ortográficos de línguas ocidentais desenvolvidos apresentam características semelhantes, o que ajudou na escolha da arquitetura deste trabalho. Todos os trabalhos têm um banco de dados com informações que foram obtidas na fase de treino. Essas informações são a base para a escolha da definição da palavra substituta. Portanto, a fase de treino é muito importante para o projeto. Como o trabalho é voltado para área jurídica, foi utilizado um *corpus* nessa área.

A partir do estudo dos trabalhos relacionados foi possível a definição e a criação de corretor ortográfico para a língua portuguesa, tratando um conjunto reduzido, porém significativo, de situações. Além disso, o corretor foi desenvolvido de modo a disponibilizar todas as possibilidades para uma expansão futura. As técnicas utilizadas foram escolhidas com base nos resultados apresentados nos trabalhos relacionados, sendo que o principal diferencial do trabalho consiste na abordagem de treino utilizada, integrando aspectos probabilísticos ao processo. O sistema implementado foi avaliado com um *corpus* jurídico, de 424.662 palavras e 147 textos, sendo que se verificou a obtenção de 97,9% de precisão na correção. Dessa forma, considera-se que os resultados obtidos foram positivos e atenderam ao objetivo inicial deste trabalho.

O código fonte do sistema e a interface web podem ser visualizados e acessados no site GitHub. Os endereços dos projetos são, respectivamente, http://github.com/edpittol/certografia e http://github.com/edpittol/certografia-web. Também foi criado um pacote e enviado para o site Pypi (http://pypi.python.org). Assim, ele pode ser facilmente instalado utilizando o comando pip.

Como trabalho futuro, está previsto o estudo para implantar o corretor ortográfico como sistema de acesso automatizado, a partir de um serviço web para que ele possa ser acessado por outros serviços na internet, assim sistemas diversos poderão realizar consultas a esse sistema. Outro aspecto a ser desenvolvido em trabalhos futuros está associado à ampliação de uso de recursos linguísticos. O trabalho desenvolvido não verifica a ordem das palavras, de acordo com sua categoria, em uma sentença. Para isso poderia ser utilizada a marcação morfossintática em conjunto com um modelo de linguagem e a geração de um outro banco de dados. Esse banco armazenaria as informações com as regras gramaticais. Assim seria possível avaliar se a ordem das classes gramaticais das palavras está correta na frase e realizar testes que identificariam melhorias possíveis nos resultados.

Referências

[1] RAM, A,; MOORMAN, K. *Understanding Language Understanding: computational models of reading*. London: The MIT Press, 1999.

[2] BICK, E. Parsing System Palavras. *Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*. Arhus: University of Arhus, 2000.

[3] OTHERO, G. A.; MENUZZI, S. M. *Linguística Computacional* - Teoria e Prática. [S.l.]: Parábola, 2005.

[4] XUE, M.; ZHU, C. A Study and Application on Machine Learning of Artificial Intellligence. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 21, 2009. *Anais*... Pasadena: IEEE Computer Society, 2009. p. 272-274.

[5] JURAFSKY, D.; MARTIN, J. *Speech and language processing:* an introduction to natural language processing, computational linguistics, and speech recognition. [S.l.]: Prentice Hall, 2000. (Prentice Hall series in artificial intelligence).

[6] MARRINAN, T. *Markov Chains:* roots, theory, and application. [S.l.: s.n.], 2008.

[7] MANNING, C. D.; SCHOTZE, H. *Foundations of Statistical Natural Language Processing*. [S.l.]: MIT Press, 1999.

[8] BAYES, T. An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, v. 53, p. 370-418, 1763.

[9] WAGNER, R.; FISCHER, M. The String-to-String Correction Problem. *Journal of the ACM*, v. 21, n. 1, p. 168-173, 1974.

[10] LEVENSHTEIN, V. Binary Codes Capable of Correcting Deletions and Insertions and Reversals. *Soviet Physics Doklady*, [S.l.], v. 10, n. 8, p. 707-710, 1966.

[11] RUCH, P.; BAUD, R.; GEISSBUHLER, A. Toward filling the gap between interactive and fully-automatic spelling correction using the linguistic context. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, 2001. *Anais*... [S.l.: s.n.], 2001. p. 199-204. v.1.

[12] DELDEN, S. van; BRACEWELL, D. B.; GOMEZ, F. Supervised and Unsupervised Automatic Spelling Correction Algorithms. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, 2004. *Anais*... , 2004. p. 530-535.

[13] NGUYEN, P. H.; NGO, T. D.; PHAN, D. A.; DINH, T. P. T.; HUYNH, T. H. Vietnamese spelling detection and correction using Bi-gram, Minimum Edit Distance, SoundEx algorithms with some additional heuristics. In: RIVF, 2008. *Anais*... IEEE, 2008. p. 96-102.

[14] BYUN, J.; RIM, H.-C.; PARK, S.-Y. Automatic Spelling Correction Rule Extraction and Application for Spoken-Style Korean Text. In: ALPIT, 2007. *Anais*... IEEE Computer Society, 2007. p. 195-199.

[15] ALKANHAL, M. et al. Automatic Stochastic Arabic Spelling Correction With Emphasis on Space Insertions and Deletions. *IEEE Transactions on Audio, Speech, and Language Processing*, [S.l.], v. 20, n. 7, p. 2111-2122, 2012.

[16] RODPHON, M.; SIRIBOON, K.; KRUATRACHUE, B. Thai OCR error correction using token passing algorithm. In: IEEE PACIFIC RIM CONFERENCE ON COMMUNICATIONS, COMPUTERS AND SIGNAL PROCESSING, 2001. *Anais*... [S.l.: s.n.], 2001. v. 2. p. 599-602.

[17] BIRD, S.; KLEIN, E.; LOPER, E. *Natural Language Processing with Python*: analyzing text with the natural language toolkit. Beijing: O'Reilly, 2009.

[18] DOWTY, D. R.; WALL, R. E.; PETERS, S. *Introduction to Montague semantics*. Dordrecht: D. Reidel Pub. Co., 1981.

[19] CHOMSKY, N. Syntactic structures. Berlin: The Hague Mounton & Co. 1957.

[20] GEACH, P.; BLACK, M. *Translations from the philosophical writings of Gottlob Frege*. Totowa: Barnes & Noble Books, 1952.