# HASCH: Um Corretor Ortográfico Automático de Alto Desempenho para Textos Oriundos da Web*

***Abstract.*** *Recently, we observe a real democratization data generation caused by the rise of the* Web 2.0*. These data are mostly provided in the form of texts, ranging from the reports provided by news portals, using a formal language, to comments in blog and micro-blogging applications, that abuse the use of an informal language ("Internetês"). Address this heterogeneity is an essential preprocessing so that these data can be used by tools that aim to infer accurate information based on such data. Thus, this work presents the HASCH (***H****igh Performance* **A****utomatic* **S****pell* **CH***Ecker), HASCH is completely parallelized in shared memory. In our evaluation the HASCH was extremely effective in the correction of texts from different sources, with a linear speedup in the processing of large texts.*

***Resumo.*** *A* Web 2.0 *provocou uma democratização no âmbito da geração de dados, providos na grande maioria na forma de textos, tanto formais, como reportagens providas por portais de noticias, quanto informais ("Internetês"), como comentários em micro-blogging. Tratar essa heterogeneidade é uma preprocessamento indispensável para que esses dados possam ser utilizados por ferramentas que visam inferir informações precisas. Apresentamos nesse trabalho o HASCH (***H****igh Performance* **A****utomatic* **S****pell* **CH****ecker), um corretor ortográfico automático, completamente paralelizado em memória compartilhada, cujo objetivo é preprocessar grandes volumes de textos em português coletados da Web, de forma eficiente.*

## 1. Introdução

Com o surgimento da Web 2.0 observamos uma verdadeira democratização no âmbito da geração de dados. Essa democratização, aliada à falta de controle do que é disponibilizado na Web possibilita uma geração massiva de dados. Esses dados são, em sua grande maioria, providos na forma de textos, que vão desde de reportagens veiculadas por diversos portais de notícias, que utilizam na maioria das vezes uma linguagem mais formal, até comentários adicionados em aplicações de blog e micro-blogging (e.g., Twitter), que utilizam uma linguagem bastante informal e típica da internet, o "Internetês". Essas diferenças fazem com que a qualidade (aspectos sintáticos e semânticos) dos dados seja bastante heterogênea. A qualidade dos dados é um aspecto indispensável a ser considerado pelas ferramentas que visam inferir informações precisas a partir desses dados, como técnicas de mineração de dados, sistemas de recomendação, etc. Um cenário que demonstra a importância desse aspecto, está relacionado ao uso de opiniões postadas em aplicações de micro-blogging por grandes corporações na tomada de decisões estratégicas. Por não haver nenhum mecanismo de controle de qualidade do conteúdo publicado, e por ser um espaço extremamente informal, torna-se imprescindível considerar a qualidade dos dados.

*Esse trabalho foi parcialmente financiado por CNPq, CAPES, FINEP, Fapemig, e INWEB.

Nessa linha, uma importante ferramenta que precede qualquer outra tarefa de tratamento desses textos são os corretores ortográficos [Norvig ]. A correção ortográfica pode ser dividida em dois subproblemas: a verificação do erro; e a correção do erro encontrado. O primeiro consiste apenas em definir se a palavra pertence ou não a língua que está sendo focada. O segundo é a correção em si, que realiza a escolha de uma palavra entre uma lista de palavras "próximas"da palavra errada. Os corretores ortográficos, podem realizar correções seguindo duas abordagens: iterativas e não iterativas. Corretores iterativos se beneficiam da ajuda do usuário para selecionar a palavra mais apropriada para a correção e os não interativos são totalmente automáticos, verificando e corrigindo as palavras erradas mediante a aplicação de heurísticas [Kenneth W. Church 1991].

Nesse trabalho propomos uma ferramenta de correção ortográfica completamente automática para ser utilizada na ponta de coletores automáticos de texto da Web (tweets, notícias, etc.), o *HASCH* (**H**igh Performance **A**utomatic **S**pell **CH**ecker). O objetivo deste corretor é preprocessar os textos coletados, primando pela qualidade, para que os mesmos possam ser efetivamente utilizados pelas ferramentas que visam inferir informações. O *HASCH* é baseado em heurísticas de correção bem conhecidas em contextos mais gerais [Norvig , Kenneth W. Church 1991, Peterson 1980], além de algumas abordagens próprias com o intuito de tratar o problema específico de correção do "Internetês", bem como abordar peculiaridades da língua portuguesa, como a acentuação. Como visa tratar um grande volume de dados, o *HASCH* é completamente paralelizado em memória compartilhada afim de minimizar o tempo de processamento. Avaliamos o *HASCH* tanto da perspectiva de eficácia (qualidade das correções realizadas), quanto da eficiência (tempo de execução) utilizando diversos textos coletados de diferentes fontes para teste. Os resultados mostraram que o *HASCH* foi extremamente eficaz na correção de textos oriundos de diferentes fontes, apresentando um *speedup* linear no processamento de textos grandes. Este corretor poderia ser utilizado em qualquer tipo de aplicação para correção de textos, mas em geral os corretores iterativos são mais atrativos quando se trata de correção de textos de usuários de editores comuns.

## 2. Trabalhos Relacionados

As pesquisas na área de correção automática de textos são normalmente divididas entre métodos para a detecção de erros e métodos para a sugestão de correções. O método mais utilizado de detecção de erros ortográficos em um determinado texto é a verificação se a palavra pertence ou não a um determinado dicionário, que deve representar bem a língua em questão. Armazenar um dicionário em memória pode ser extremamente caro em termos computacionais, sendo assim, alguns trabalhos propõem a manutenção desse dicionário na forma de mapa de bits [Nix 1981], mantêm apenas o radical da palavra para economizar espaço [McIlroy 1982], ou mesmo não utilizam nenhuma estrutura de dicionário [Edward M. Riseman 1974]. As propostas que não utilizam uma estrutura de dicionários, apesar da economia de memória, acabam pecando em eficiência. Assim, como nossa proposta é um corretor ortográfico automático eficiente, optamos por utilizar um dicionário completo uma lista com 261.945 palavras da língua portuguesa, incluindo palavras com prefixos e sufixos, conseguindo englobar a maior parte do vocabulário da língua portuguesa.

Um corretor ortográfico falha quando sinaliza uma palavra como um erro sendo que na verdade a mesma está correta, ou quando deixa de apontar um erro, sendo esse último mais preocupante. Esse tipo de falha pode acontecer quando erros de ortografia coincidem com palavras no dicionários e são conhecidos como "real-word errors" [McIlroy 1982]. Existem poucos trabalhos que abordam esse problema, sendo os mais comuns os baseados em probabilidade de ocorrência de palavras [McIlroy 1982], a qual utilizamos em nosso trabalho por meio de duas etapas de calibragem (uma anterior ao correção do texto e outra no decorrer da correção), determinamos quais palavras ocorrem mais para que tenham os maiores pesos no processo de correção, de acordo com o estilo de texto coletado, visando assim minimizar erros de "real-word".

Como uma análise semântica é computacionalmente inviável, os estudos para o aumento da qualidade das verificações e correções de palavras continuam concentradas no nível ortográfico, como vimos nos trabalhos acima descritos. Diferentemente desses, o corretor ortográfico proposto nesse trabalho não visa propor novas técnicas e sim adaptar os diversos estudos apresentados acima com o objetivo de aplicá-lo em um cenário diferente: coletores de textos da WEB em português. Nesse cenário, é necessário que o corretor seja completamente automático, capaz de corrigir não apenas erros ortográficos, mas inferir palavras a partir de abreviações, tratar das peculiaridades associadas a língua portuguesa, bem como ser extremamente rápido e capaz de processar um volume grande de dados em um espaço curto de tempo.

## 3. HASCH

O grande desafio de um corretor ortográfico totalmente automático é, dada uma palavra errada, conseguir encontrar uma correção mais provável para a mesma. Sabemos que isto não é possível apenas com uma análise em nível ortográfico, visto que, mesmo ortograficamente correta, uma palavra pode inserir erros gramaticais ou estilísticos. Assim, nessa seção descrevemos nosso corretor ortográfico para textos da WEB em português, o *HASCH* (**H**igh Performance **A**utomatic **S**pell **CH**ecker) que visa abordar os desafios acima propostos. Toda implementação foi feita na linguagem de programação C, utilizando bibliotecas padrão da linguagem e completamente paralelizado utilizando a biblioteca OpenMP. É importante mencionar que não encontramos disponível um corretor totalmente automático como o proposto neste trabalho.

### 3.1. Funcionamento Geral

Como nosso foco é trabalhar na ponta de coletores de textos da WEB, carregados de hábitos de linguagens ("Internetês") que fogem de toda regra ortográfica da língua portuguesa, nosso corretor é baseado em dois dicionários, armazenados por meio de uma estrutura *hash*: (1) o principal, contendo as 261.945 palavras mais utilizadas na língua portuguesa, incluindo palavras com seus prefixos e sufixos (palavras flexionadas), dicionário utilizado em corretores conhecidos como os do *OpenOffice*; e (2) dicionário de "Internetês", o qual nos dá uma relação dos vícios de linguagem e sua forma correta. Ambos os dicionários são implementados utilizado uma tabela hash para que o custo de pesquisa no mesmo seja o menor possível ($O(1)$).

Após o processo de construção do dicionário principal, propomos duas estratégias para ponderar as palavras de acordo com o estilo de texto específico a ser coletado. Palavras de maior peso terão probabilidade maior de serem utilizadas no processo de correção. Essa ponderação visa minimizar erros de "real-word" descritos na seção 2. A primeira estratégia, denominada **Calibragem Externa**, pode ser definida como um pré-processamento do corretor ortográfico automático. Como padrões ortográficos distintos são observados em diferentes tipos de texto, nessa estratégia, o coletor realiza a leitura de diversos textos de treino de mesmo gênero dos textos a serem corrigidos, e conta a ocorrência de todas as palavras nesses textos, preenchendo a tabela *hash* que armazena o dicionário principal antes de iniciar a correção com os pesos de cada palavra. Na segunda estratégia, denominada **Calibragem Interna**, a contagem das palavras é feita durante a execução do corretor, conforme sugerido por [Norvig ].

Estabelecidas as estratégias de construção dos dicionários, o próximo passo é descrever efetivamente o funcionamento do nosso corretor. O *HASCH* é baseados em 4 passos principais descritos a seguir:

**1 - Consulta Dicionário Principal:** Para cada palavra $w$ retirada do texto a ser corrigido, verifica-se se a mesma está presente no dicionário principal a um custo $O(1)$. Caso a palavra esteja no dicionário significa que ela está correta, e nada precisa ser feito. Caso a palavra não se encontre no dicionário principal, aciona-se o passo 2;

**2 - Consulta Dicionário "Internetês":** Para cada palavra $w$ retirada do texto coletado a ser corrigido, verifica-se se a mesma se encontra no dicionário "Internetês"a um custo $O(1)$. Caso a palavra seja encontrada, a substituição é feita pela sua forma correta armazenada. Essa estratégia garante que erros provenientes de hábitos de linguagem sejam corrigidos sem a necessidade de se passar pelas demais estratégias de correção, tornando assim nosso corretor mais rápido. Caso a pesquisa por $w$ falhe, aciona-se o passo 3;

**3 - Verifica Acentuação:** Uma questão fundamental na língua portuguesa é o uso de acentos. A falta de acentuação, ou a acentuação inadequada constituem um grande número de erros ortográficos, visto que é um pequeno detalhe que envolve inúmeras regras. Dessa forma, esse passo do corretor consiste em acentuar cada vogal de $w$ com os possíveis acentos para a mesma e verificar se a palavra acentuada está no dicionário principal. O custo dessa etapa é $O(v)$, onde $v$ é o número de vogais de $w$. Considerando o tamanho de $w$ igual a $n$, temos que, no pior caso, o custo desse passo é $O(n)$. Assim como o passo anterior, essa estratégia visa evitar que os próximos passos sejam executados, tornando assim nosso corretor mais rápido. Caso haja uma falha nesse passo, finalmente o passo 4 é acionado;

**4 - Verifica Distância de Edição:** Conforme já descrito, esse passo é acionado somente quando os demais passos anteriores falharam na correção de $w$. Para determinar a palavra correta $c$ que deverá substituir $w$, primeiramente levantamos as diversas opções possíveis. Para isso, propomos uma estratégia baseada na proposta em [Norvig ]. Dada a palavra $w$ a ser corrigida, consideramos uma lista de palavras que são ortograficamente próximas a ela. Essa proximidade é dada pela distância de edição, ou seja, o número de modificações necessárias para transformar $w$ em

outra. Essas edições podem ser de quatro formas: (1) deleção: quando removemos uma ou mais letra de $w$; (2) inserção: quando inserimos uma ou mais letras em $w$; (3) substituição: quando alteramos uma ou mais letra de $w$ por alguma letra do alfabeto; e por fim (4) transposição: corresponde a troca de letras adjacentes em $w$. Pela literatura 80% a 95% dos erros estão na distância 1 e portanto para o corretor ortográfico automático apresentado nesse trabalho consideramos apenas a listagem das palavras que estão a uma distância de edição 1 de $w$ e que estejam no dicionário. É importante mencionar que este processo é computacionalmente caro, uma vez que a complexidade da edição de deleção é $n$, da transposição $n-1$, da substituição $26^n$ e da inserção é $26^{(n+1)}$, a complexidade total dessa heurística é de $O(26^n)$, sendo $n$ o tamanho de $w$. Dentre as opções selecionadas, seleciona para correção a palavra $c$ de maior maior pontuação dentro do dicionário, a um custo $O(n)$.

A correção de textos grandes, com milhões de palavras, claramente é um processo demorado, já que as palavra são analisadas e corrigidas uma a uma por ao longo dos passos apresentados. Para cada uma dessas palavras o tempo de correção é diferente. Se considerarmos uma palavra correta, ela terá sua análise realizada de forma bastante rápida, visto que é apenas uma busca no dicionário. Porém quando temos palavras que não estão corretas e que necessitam da aplicação de um dos passos para a sua correção, a aplicação desses passos pode também apresentar diferentes tempos, tendo em vista que a palavra pode ser corrigida no passo 2 (Consulta Dicionário "Internetês"), no passo 3 (Verifica Acentuação) ou falhar em ambas e apenas ser corrigida no passo 4 (Verifica Distância de Edição) a qual é, claramente, a parte mais custosa. Conforme podemos observar pelas descrições feitas dos passos de execução do nosso corretor ortográfico o mesmo apresenta uma complexidade dada por $T * max(O(1), O(n_{avg}), O(26^{n_{avg}}))$ sendo $n_{avg}$ o número médio de caracteres das palavra a serem corrigidas e $T$ o total de palavras a processadas. Sendo assim, no pior caso, a complexidade final do nosso corretor é dado por uma função exponencial em relação ao número médio de caracteres das palavra a serem corrigidas, o que é computacionalmente caro. Essa análise reforça o argumento de que uma implementação eficiente é essencial. Dessa forma, apresentamos na seção seguinte, a estratégia de implementação paralela adotada.

### 3.2. Paralelização

Uma questão interessante a ser considerada na correção de textos utilizando o corretor ortográfico automático proposto é o fato de ser considerado apenas o nível da palavra, ou seja, a correção de uma palavra independe totalmente da correção das outras palavras, tornando esse processamento uma tarefa embaraçosamente paralela. Tendo em vista essas considerações, é visível que uma paralelização de dados é facilmente aplicável ao problema, no qual o texto a ser corrigido pode ser dividido em diferentes processos. Assim, na implementação de nosso corretor adotamos a paralelização de dados utilizando memória compartilhada e a biblioteca utilizada foi a OpenMP. A seguir, apresentamos alguns detalhes de nossa paralelização.

**1 - Divisão dos dados:** A divisão do texto é feita considerando-se o número de palavras presentes. Dessa forma, o primeiro passo antes da correção é a contagem de quantas palavras estão contidas no texto e posteriormente para cada processo (*thread* e seu respectivo *id*), são definidas quantas palavras cada um receberá.

**2 - Região Crítica:** A única região crítica existe quando ocorre a atualização dos pesos no dicionário principal, no qual implementamos uma exclusão mútua simples para resolver o problema.

**3 - Sincronização da correção:** A independência dos dados entre as *threads* não é completa, visto que devemos manter a ordem do texto, ou seja, o trecho de texto corrigido da *thread* 0 deve vir antes do texto corrigido pela *thread* 1, e assim sucessivamente. Para tanto utilizamos um *buffer* que irá receber as partes de textos corrigidas, concatenando os textos de acordo o *id* das *threads*. Caso uma *thread*, por exemplo a de *id*2, termine sua correção antes da *thread* de id 1, ele ficará esperando até a *thread* 1 terminar e escrever no *buffer*, mantendo a ordem e semântica do texto original.

## 4. Avaliação do HASCH

Nessa seção apresentamos e discutimos os resultados relacionados a avaliação do nosso corretor ortográfico automático, o *HASCH*. Nossa avaliação foi dividida em duas perspectivas: (1) eficácia, relacionada com qualidade das correções realizadas; e (2) eficiência, relacionada aos tempos de execução do corretor.

Para a avaliar a eficácia do *HASCH* utilizamos, basicamente, dois contextos distintos: textos formais coletados em portais de notícias, e textos informais coletados de microblogs (Twitter). A partir desses contextos, criamos diversos cenários de avaliação, variando o contexto de aplicação e os textos utilizados na calibragem. Por exemplo, correção e calibragem utilizando textos do mesmo contexto e correção e calibragem utilizando textos de contextos distintos. Após a realização dos testes, validamos de forma manual, permitindo uma análise detalhada do comportamento do nosso corretor, observando o tratamento ou não de cada um dos erros presentes no texto, e avaliando individualmente cada passo implementado.

Para avaliar a eficiência do nosso corretor, foi realizada uma bateria de testes, em que o foco era o tempo de resposta do corretor. Sendo assim, executamos nosso corretor em diferentes cenários onde variamos o tamanho dos arquivos de entrada e o número de processos utilizados (de 1 a 8), observando o tempo real de execução. Os testes foram executados em uma máquina com CPU Intel(R) Core(TM) i7 2.67 Ghz e memória RAM de 8GB. Para tomar os tempos, foram feitas 20 execuções para cada cenário (tamanho de arquivo + número de processos) e o tempo final apresentado é resultado da média dos tempos das 20 execuções. Nas seções subsequentes são detalhados os testes e analisados os resultados obtidos, bem como descrevemos as bases usadas para a realização das avaliações.

### 4.1. Bases

Para testar a eficácia utilizamos duas bases de dados de teste distintas, coletadas da Web: a primeira contendo um total de 200 *posts* de diferentes usuários do Twitter; e a segunda contendo artigos completos, todos relacionados ao tema "Economia"de um portal de notícias. Cada uma das bases contabiliza aproximadamente 3.300 palavras a serem avaliadas e possivelmente corrigidas pelo *HASCH*. Além desses dois conjuntos, coletamos também outros dois conjuntos, relacionados aos mesmos dois temas apresentados acima, para realizar o processo de calibragem, cada um

contendo aproximadamente 6.000 palavras. Ambas as bases de teste foram processados utilizando três versões distintas do dicionário de principal: sem calibragem, calibrado com *posts* do Twitter e calibrado com textos de portais de notícias.

Para a avaliação da eficiência do corretor ortográfico proposto, utilizamos 10 conjuntos de teste, contendo diferentes quantidades de *posts* coletados do microblog Twitter. Os conjuntos foram construídos variando o número de *posts* de 100 a 100.000. A quantidade total de palavras a serem processadas variou de 1.852 a 1.211.881, de acordo com cada conjunto.

### 4.2. Eficácia da correção

Na primeira avaliação feita utilizamos um conjunto de testes contendo apenas *posts* do Twitter e um dicionário principal construído sem calibragem externa (preprocessamento). A tabela 1 apresenta uma visão geral dos resultados obtidos. Os erros identificados foram classificados em cinco classes: erros identificados e corrigidos corretamente (CC); erros identificados, porém não corrigidos (NC) ou corrigidos erroneamente (CE); erros gramaticais (EG), isto é, erros que deveriam ser corrigidos, mas não foram porque a palavra existe no dicionário, mas está empregada no texto de forma incorreta; correções desnecessárias (CD); e erros cuja palavra esperada não foi encontrada no dicionário principal (NE). Para esta análise, as correções de *hashtags*, *usernames*, *URLs*, *emoticons*, nomes próprios e expressões em línguas estrangeiras foram desconsiderada.

| CC | NC ou CE | EG | CD | NE |
|---|---|---|---|---|
| 157 | 64 | 45 | 7 | 5 |

**Tabela 1. Resultados Obtidos para o Conjunto de Teste Contendo *posts* do Twitter**

Podemos observar pela tabela 1 que, mesmo num cenário onde a frequência de erros é consideravelmente alta, como é o caso de microblogs, o corretor apresentou um bom funcionamento, corrigindo aproximadamente 56% dos erros encontrados. É importante ressaltar que boa parte dos erros não corrigidos estão em nível gramatical. Este tipo de erro está fora dos alcances de correção da ferramenta, uma vez que a palavra em questão, apesar de inadequada ao contexto, está ortograficamente correta, isto é, existe no dicionário de palavras.

O número de correções desnecessárias e de erros nos quais a palavra esperada não foi encontrada no dicionário principal, apesar de relativamente baixo, mostra que o aumento do tamanho dos dicionários principal e de "Internetês"poderia influenciar positivamente funcionamento do corretor. No primeiro caso, a correção não deveria acontecer, pois a palavra está correta. No segundo caso, a palavra de fato está incorreta e foi identificada como tal, mas de maneira alguma teria como ser corrigida corretamente, uma vez que a palavra esperada não existe nos dicionários utilizados. O número de erros não corrigidos ou corrigidos erroneamente corresponde a aproximadamente 23% do total de erros. Podemos dizer que este valor é relativamente baixo, uma vez que estamos tratando de um cenário onde a preocupação com a escrita é mínima, e o número de abreviações e erros propositais de ortografia é significativamente alto. No gráfico da figura 1 apresentamos mais detalhes com relação ao funcionamento do nosso corretor em cada um dos passos de execução.

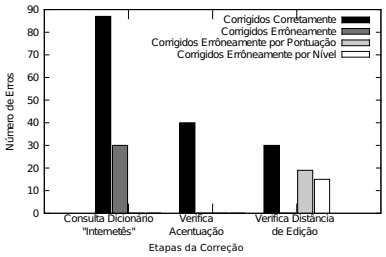


**Figura 1. Avaliação das Correções**

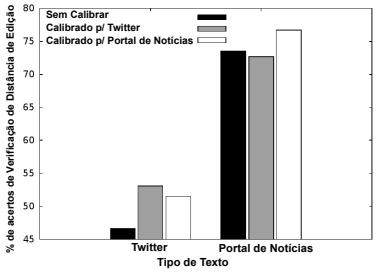


**Figura 2. Calibragem Externa.**

Observando a figura 1, podemos notar que a maior parte dos erros identificados consiste de abreviações detectadas pelo dicionário de "Internetês". Esse passo foi o que proporcionou a maior contribuição para o bom funcionamento do corretor. Aproximadamente 68% das abreviações foram identificadas e corrigidas corretamente. Os 32% restantes consistem de abreviações que não foram identificadas, o que mostra que investir em um dicionário de "Internetês"mais completo pode proporcionar melhoras significativas na eficácia do corretor. Observamos também um bom funcionamento no passo da verificação de acentuação. Todos os erros identificados pelo mesmo foram perfeitamente corrigidos, salvo aqueles considerados erros gramaticais.

Com relação ao passo de verificação de distância de edição, os erros corrigidos erroneamente foram divididos em dois tipos: os causados por *score* e os causados por nível. No primeiro caso, a palavra correta poderia ter sido selecionada, mas não foi, pois uma outra palavra com um *score* maior, também foi encontrada. Em parte, isto se deve a ausência de uma calibragem externa do dicionário principal. Ao utilizarmos apenas a calibragem interna temos que palavras menos utilizadas possuem a mesma chance de serem escolhidas do que aquelas comumente utilizadas. No segundo caso, os chamados erros por nível (comuns no cenário de microblogs), ocorrem quando a distância de edição da palavra a ser corrigida em relação a esperada é muito grande, e portanto a heurística não a alcança. Sendo assim, temos que o passo que verifica a distância de edição das palavras apresenta os resultados menos favoráveis quando tratamos de textos oriundos de microblogs, corrigindo aproximadamente 46% dos erros identificados pela mesma.

Afim de avaliar a importância do processo de calibragem externa, comparamos o funcionamento de nosso corretor utilizando as duas bases de teste mencionadas (Twitter e notícias de economia) em três cenários diferentes: (1) sem a calibragem, (2) calibrado com textos do Twitter; e (3) calibrado com textos de economia. Os resultados dessa avaliação são apresentados no gráfico da figura 2.

Podemos observar que, sempre que o dicionário é calibrado de maneira coerente ao tipo de texto a ser tratado, há um acréscimo na eficácia. Quando a calibragem é realizada utilizando textos de estilos diferentes do que será tratado, há o risco de perda de eficácia, mesmo em relação ao dicionário sem calibragem. Este caso é mostrado no gráfico, quando utilizamos um dicionário calibrado com textos do Twitter para corrigir textos de portais de notícias. A quantidade de texto de treino utilizada no processo de calibragem externa foi pequena, e ainda sim podemos notar uma sensível melhoria na qualidade da correção realizada quando

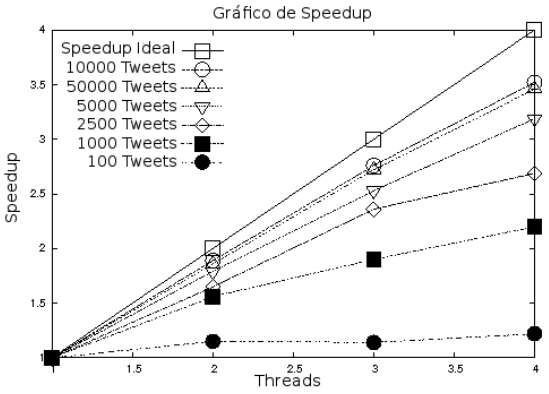


**Figura 3. Speedup**

a calibragem é feita adequadamente. A partir desses resultados podemos concluir que a utilização de uma quantidade maior de textos para a calibragem externa do dicionário pode proporcionar resultados ainda melhores.

Assim, sob o aspecto da eficácia do *HASCH*, temos que o uso de um dicionário extra voltado para o "Internetês´´ se mostrou uma excelente estratégia e que, uma expansão desse dicionário tende a levar a um corretor ainda mais preciso. Outro ponto de destaque foi a estratégia de verificação de acentuação, a qual conseguiu resgatar uma quantidade razoável de erros muito comuns na língua portuguesa que é a falta de acentuação.

### 4.3. Eficiência da paralelização

Nessa seção apresentamos os resultados relacionados a avaliação de eficiência do *HASCH* para diferentes conjuntos de entrada, com tamanhos diferentes, variando o número de processadores utilizados. Sumarizamos esses resultados por meio do gráfico da figura 3 no qual apresentamos os valores de *speedup* obtidos para os diversos conjuntos de teste utilizados, além da curva de *speedup* ideal.

Analisando a figura 3 podemos constatar que a paralelização é pouco eficiente quando utilizada para tratar conjuntos pequenos de palavras. O valor de *speedup* para as bases com menos de 400.000 palavras (2.500 *tweets*) mostra-se pouco significativo (pequena redução no tempo de execução a medida que aumentamos o número de processos), não justificando o uso de mais de uma *thread*. Isto se deve ao fato de que o processo de divisão do texto a ser corrigido entre as *threads* gera um *overhead* que acaba por anular o ganho que seria proporcionado pela paralelização dos dados, uma vez que, dado o pequeno número de palavras, a execução do algoritmo, mesmo em serial, é bastante rápida. Entretanto, quando cresce o tamanho do conjunto de palavras, o aumento no número de *threads* executando simultaneamente proporciona uma redução no tempo médio da correção. Para conjuntos de teste com quantidade de palavras acima de 400.000 (5.000 *tweets*), o *speedup* alcançado é próximo do *speedup* ideal. Isso ocorre porque o tempo gasto no processo de divisão do texto mantem-se o mesmo, mas a tarefa de correção em si é dividida igualmente entre as *threads*, e executadas em paralelo. Por fim, fica claro que, apesar do valor de *speedup* ter se estabilizado em relação ao aumento do conjunto de teste, o ganho continua

sendo significativo quando aumentamos o número de processadores e tratamos de conjuntos de teste relativamente grandes. Isto indica que, nestes casos, o aumento da quantidade de *threads* em execução pode continuar proporcionando uma redução linear no tempo de correção, equivalente à quantidade de processadores utilizados.

## 5. Conclusões e Trabalhos Futuros

Nesse trabalho propomos uma ferramenta de correção ortográfica completamente automática, paralelizado em memória compartilhada para ser utilizada na ponta de coletores automáticos de textos da WEB em português, o *HASCH*. Avaliamos tanto a eficácia quanto a eficiência de nosso corretor utilizando textos coletados de diferentes fontes. Sob o aspecto da eficácia, constatamos que o uso de um dicionário extra voltado para o "Internetês"se mostrou uma excelente estratégia e que, sua expansão tende a levar a um corretor ainda mais preciso. Outro ponto de destaque foi a estratégia de verificação de acentuação que conseguiu resgatar uma razoável quantidade de erros comuns na língua portuguesa. Por fim, podemos destacar a estratégia de calibragem externa em que concluímos que o uso de uma base de treino maior tende a melhorar a eficácia de nosso corretor. Além disso, sob o aspecto eficiência, constatamos que nossa ferramenta apresentou um excelente desempenho, alcançando um *speedup* próximo ao linear em textos com mais de 400000 palavras.

Como trabalhos futuros, propomos avaliar o corretor utilizando dicionários de palavras e de "Internetês" mais completos, em contextos mais variados, além de avaliar outras estratégias de paralelização e comparar com a adotada nesse trabalho.

## Referências


Edward M. Riseman, A. R. H. (1974). A contextual post-processing system for error correction using binary n-grams. *IEEE Trans Computers*, C-23(5):480–493.

Kenneth W. Church, W. A. G. (1991). Probability scoring for spelling correction. *Statistics and Computing*, 1:93–103.

McIlroy, M. D. (1982). Development of a spelling list. *IEEE Transactions on Communications*, COM-30(1):91–99.

Nix, R. (1981). Experience with a space efficient way to store a dictionary. *Communications of the A.C.M.*, 24(5):297–298.

Norvig, P. How to Write a Spelling Corrector. *http://norvig.com/spell-correct.html*.

Peterson, J. L. (1980). Computer programs for detecting and correcting spelling errors. *Communications of the A.C.M.*, 23(12):676–687.