BeShort: Uma nova abordagem para encurtamento de URLs

Pedro P. S. Freitas[1], **Wellington J. Dores**[1], **Fabrício Benevenuto**[1,2]

[1]Departamento de Ciência da Computação
Universidade Federal de Ouro Preto (UFOP)
Ouro Preto, MG, Brasil

[2]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, MG, Brasil

{pedropufop, wellingtonjdores, benevenuto}@gmail.com

Abstract. *Microblogs like Twitter are social systems designed to allow users to post messages containing no more than 140 characters. With the wide use of short messages on the Web, the use of URL shorteners are increasingly becoming popular. These systems translate a shortened URL into a new URL, typically with few characters, and redirect requests that target the shortened version of the URL to the original long URL. Although extremely efficient, the centralized architecture of such services can introduce delays to users and have been widely used as a way to obfuscate spam, phishing and malware. This paper presents BeShort, a distributed approach for shortening URLs able to avoid such problems. Our approach consists of replacing frequently terms (e.g. "www" and ".com.br") for UTF-8 characters that are usually not used in URLs. To test BeShort we built a dataset containing 50 million URLs of two popular URL shortening services. Our results show that the BeShort obtains compression rates as efficient as the rates obtained by existing approaches.*

Resumo. *Microblogs como o Twitter são sistemas sociais voltados unicamente para a postagem de mensagens com no máximo 140 caracteres. Com o grande uso de mensagens curtas na Web, o uso de encurtadores de URLs está se tornando cada vez mais comum. Sistemas encurtadores traduzem uma URL em uma nova URL, tipicamente com poucos caracteres, e redirecionam requisições à URL encurtada para a URL longa original. Apesar de extremamente eficiente, esses serviços podem introduzir atrasos para seus usuários e têm sido amplamente utilizados para esconder spam, phishing e malware. Esse trabalho apresenta o BeShort, um algoritmo para encurtamento de URLs capaz de evitar tais problemas. Nossa abordagem consiste em substituir partes frequentes de URLs (ex. "www" e ".com.br") por caracteres UTF-8, normalmente não utilizados em URLs. Para testar nossa abordagem, utilizamos uma base contendo 50 milhões de URLs de dois serviços encurtadores de URL bastante populares. Nossos resultados mostram que o BeShort consegue taxas de encurtamento tão eficientes quanto as taxas praticadas pelos sistemas mais populares atuais.*

1. Introdução

Recentemente, a Web tem recebido uma nova onda de aplicações associadas ao crescimento e proliferação das redes sociais online. De acordo com a Nielsen Online [Re-

port 2009], as mídias sociais já superaram a troca de emails como a atividade online mais popular. Mais de dois terços da população online global visita ou participa de redes sociais e blogs. Como comparação, se o Facebook fosse um país, seus 850 milhões de usuários registrados colocariam esta aplicação como o terceiro país mais populoso do mundo [Facebook 2012]. O Twitter, um sistema que permite unicamente a postagem de mensagens curtas com no máximo 140 caracteres (tweets), recebe cerca de 500 milhões de mensagens por dia, que são enviadas a milhões de usuários.

O uso de mensagens curtas tem sido amplamente explorado nas redes sociais, permitindo que os usuários postem mensagens sobre tudo, incluindo notícias, casualidades, opiniões sobre a repercussão de eventos ou produtos [Cha et al. 2010] e até mesmo expressões de sentimentos e aspectos emocionais [Gonçalves et al. 2012]. O Twitter ainda é utilizado em campanhas políticas [Obama 2012], promoção de negócios, onde as lojas criam contas para fazer divulgação de ofertas e informações de seus produtos, e na comunicação de situações de emergência [Hughes and Palen 2009, Sutton et al. 2008, Sakaki et al. 2010]. Nesse mesmo contexto, milhões de URLs são compartilhadas todos os dias [Rao 2010], mudando a forma como as pessoas descobrem conteúdo na Web [Rodrigues et al. 2011]. Várias redes sociais impõem um limite superior no tamanho da mensagem, levando os usuários a utilizar um serviço encurtador de URLs para economizar espaço de suas mensagens.

De fato, encurtar URLs vem se tornando uma das principais maneiras para a fácil disseminação e compartilhamento de URLs. Serviços encurtadores de URLs, como Bit.ly[1] e TinyURL,[2] estão se tornando cada vez mais comuns. Os encurtadores de URLs traduzem uma URL (que pode consistir de centenas de caracteres) em uma nova URL, tipicamente com poucos caracteres que retorna os códigos HTTP 301 ou 302 de redirecionamento para a URL longa original [Antoniades et al. 2011]. Por exemplo, o link http://nyti.ms/1VKbrC, ao ser acionado, irá redirecionar para o sítio Web original, que comparado com a versão encurtada, consiste em uma URL com mais do que o dobro de caracteres. Apesar do primeiro sistema, com popularidade notável, ter surgido em 2002, hoje usuários podem escolher uma imensa variedade de serviços [PRLOG 2010].

Embora úteis, encurtadores de URLs introduzem alguns novos problemas que serão discutidos a seguir.

Facilidade para *phishing* e *spam*: Serviços encurtadores de URL são comumente utilizados como forma de esconder *spam* e *phishing*, o que vem sendo reportado em um grande número de trabalhos científicos recentes [Benevenuto et al. 2010, Chhabra et al. 2011, Grier et al. 2010, Thomas et al. 2011, Lee et al. 2011, Klien and Strohmaier 2012]. Em particular, *phishers* utilizam URLs encurtadas para esconder suas URLs, conforme mostrado recentemente em [Chhabra et al. 2011]. Como exemplo, a figura 1 mostra um *tweet* real contendo uma URL que aparece em listas negras de *phishing*, ofuscada por uma URL do Bit.ly. Clicando na URL do *tweet*, usuários são levados a uma página que parece a página de *login* do Twitter. Tentados pela oferta de acessar o perfil de outros usuários, um usuário pode entrar com suas credenciais do Twitter e perder sua conta para *phishers*. De posse de uma conta real, *phishers* podem realizar ataques mais elaborados (ex. ataques para obter cartões de crédito ou contas de bancos) aos seguidores dos usuários com a conta

[1] http://bit.ly
[2] http://tinyurl.com

comprometida [Chhabra et al. 2011].

Figura 1. Um tweet com uma URL encurtada (utilizando bit.ly). Usuários que clicam no link, são direcionados para uma página de *phishing*

Atrasos de redirecionamento: Os serviços encurtadores de URL mais utilizados atualmente, como o Bit.ly e TinyURL, encontram-se hospedados no exterior, exigindo, muitas vezes, redirecionamento para servidores localizados em regiões muito distantes. Tal redirecionamento pode impor severos atrasos no tempo de resposta, conforme medido e reportado em [Antoniades et al. 2011]. Além disso, por se tratarem de serviços gratuitos, a grande maioria desses encurtadores de URL não oferecem garantias sobre a qualidade de seus serviços e podem até mesmo não atender requisições, caso estejam sobrecarregados.

Esse cenário sugere a necessidade do desenvolvimento de uma arquitetura que seja capaz de encurtar URLs de forma a evitar os problemas apontados acima. Este trabalho visa investigar uma abordagem para encurtar URLs que dispensa o uso de redirecionamento para um servidor central. A ideia é que um algoritmo de encurtamento de URL seja executado no momento do envio da mensagem à rede social (o algoritmo encurta a URL) que, ao ser recebida, é expandida e exibida aos usuários em sua versão longa original.

O restante do trabalho está organizado da seguinte forma. A seção 2 aborda trabalhos relacionados, mostrando estudos que apontam a utilização das URLs curtas e confirmam seu uso para ofuscar URLs maliciosas. Em seguida, a seção 4 mostra como foi obtida a base de dados utilizada. Depois, na seção 3, apresentamos a arquitetura do BeShort comparando-a com a arquitetura praticada pelos serviços utilizados atualmente. Finalmente, nas seções 5 e 6, apresentamos os experimentos realizados e concluímos o trabalho.

2. Trabalhos relacionados

Pelo fato de nenhuma proposta relacionada ao BeShort ter sido apresentada na literatura até o presente momento, a seguir apresentamos trabalhos que mostram como são utilizadas as URLs encurtadas, em quais aplicação estão sendo usadas e trabalhos que quantificam o uso de URLs curtas para uma melhor propagação de URLs que contém algum tipo de ataque malicioso (*spam*, *phishing*).

Rodrigues e colaboradores [Rodrigues et al. 2011] provêm uma série de análises sobre os padrões de propagação de URLs entre os usuários do Twitter. Eles quantificam o aumento de audiência que o uso de *retweets* pode causar a uma URL e mostram que sites pouco populares na web podem se tornar populares através do Twitter. O trabalho ainda identifica características típicas da estrutura das árvores de propagação de informação nesse sistema e mostra que, no Twitter, as árvores são mais largas do que profundas.

O trabalho mencionado ressalta o grande interesse por espalhamento de informações em sistemas como o Twitter e quantificam o volume de URLs comparti-

lhadas nesses sistemas, sendo que a maioria dessas é postada de forma encurtada. De fato, Antoniades e colaboradores [Antoniades et al. 2011] mostram que cerca de 87% das URLs postadas no Twitter são encurtadas e que URLs encurtadas não são utilizadas apenas no Twitter, mas também em outros serviços como emails e outras redes sociais online. Além disso, os autores mostram que URLs encurtadas possuem vida curta em termos de popularidade, ou seja, URLs possuem popularidade por intervalos pequenos de tempo e depois deixam de ser usadas. Tal observação sugere que sistemas encurtadores centralizados mantenham listas enormes de URLs que só possuem acessos em um curto intervalo de tempo. Outro ponto investigado é o atraso imposto por encurtadores de URL, que se mostrou significativo, chegando a ser 2 vezes superior do que o acesso direto à URL em algumas situações.

Um aspecto relacionado à grande popularidade do uso de encurtadores de URLs é a proliferação de diferentes formas de *spam* na Web. Chhabra e colaboradores [Chhabra et al. 2011] realizam uma ampla análise do espalhamento de *phishing* através de URLs encurtadas. Por meio da análise de URLs extraídas de listas negras e estatísticas de acessos a essas URLs obtidas pela API do Bit.ly, os autores mostram que *websites* de redes sociais como Twitter e Orkut competem com serviços de comércio eletrônico como PayPal e eBay em termos do foco de *phishers*. Os autores mostram que várias URLs encurtadas por *phishers* tiveram pouca redução em seu tamanho, sugerindo que o uso do sistema por alguns usuários é simplesmente para ofuscar URLs contendo *phishing*. Outras evidências de *spam* no Twitter também foram apresentadas em outros estudos [Grier et al. 2010, Chhabra et al. 2011, Lee et al. 2011].

Finalmente, além do Twitter, *spam* tem sido observado em outras mídias sociais, como MySpace [Lee et al. 2010], YouTube [Benevenuto et al. 2009], Facebook [Gao et al. 2010], Delicious [Markines et al. 2009], FourSquare [Vasconcelos et al. 2012] e Apontador [Costa et al. 2013]. Potencialmente, encurtadores de URL também poderiam ser utilizados para ofuscar URLs nesses sistemas. Nossa contribuição nesse trabalho é investigar as bases para a construção de um sistema de encurtamento de URLs com uma abordagem que seja capaz de minimizar os problemas trazidos pelos sistemas atuais de encurtamento de URLs.

3. Arquitetura do BeShort

Os sistemas encurtadores de URL atuais utilizam uma arquitetura centralizada, como é ilustrado na figura 2. Nesses sistemas, usuários precisam encurtar suas URLs antes de serem postadas. Ao acessar uma URL encurtada, usuários fazem requisições para o serviço de encurtamento que retorna os códigos HTTP 301 ou 302 de redirecionamento para a URL longa original.

Nosso objetivo nesse trabalho é propor e avaliar a viabilidade de uma arquitetura segura para encurtar URLs que dispense o uso de um servidor central, de forma a evitar diversos aspectos indesejáveis da arquitetura centralizada. A figura 3 ilustra o funcionamento da nossa abordagem, que chamamos de BeShort. Em nossa abordagem, usuários postam URLs em formato original. Essas, por sua vez, são encurtadas pelo BeShort e em seguida enviadas de forma encurtada para as redes sociais. Antes da URL ser visualizada pelos destinatários da mensagem, o BeShort é novamente acionado para realizar a descompressão da URL para a sua forma original. Os procedimentos de compressão e

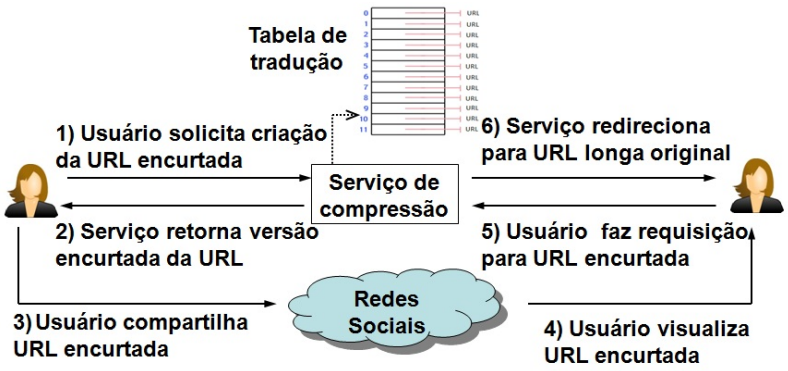


Figura 2. Arquitetura para encurtamento de URLs de forma centralizada

descompressão do BeShort acontecem de forma transparente para os usuários que postam e recebem as mensagens.

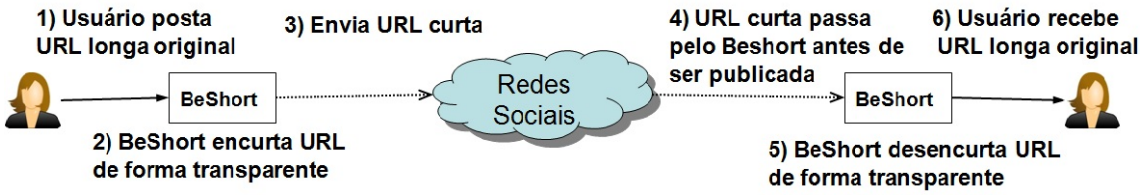


Figura 3. Arquitetura para encurtamento de URLs de forma descentralizada

Existem diferentes lugares onde o BeShort poderia ser implantado, como nos próprios clientes através de *plugins* em navegadores ou mesmo incorporado diretamente em aplicações de redes sociais específicas, como Twitter. Outra alternativa seria a implantação do BeShort em redes de fornecimento de conteúdo (CDNs – *content delivery networks*).

A estratégia de compressão do BeShort é baseada em substituir termos frequentes ocorridos em URLs (ex. http://www. ou google) por caracteres UTF normalmente não utilizados em URLs, porém aceitos em APIs de redes sociais online, como Twitter e Facebook. Para definir esses caracteres utilizamos o Padrão Unicode [The Unicode Consortium 2012], que é o padrão de codificação de caractere universal utilizado na escrita de caracteres e textos. Existem três formas de codificação do Padrão Unicode que são: UTF-8, UTF-16 e UTF-32, onde qualquer uma delas pode representar toda a gama de caracteres do padrão, a diferença entre elas é a maneira de utilização. Para este trabalho iremos utilizar a forma UTF-8 que é a codificação mais difundida na web [Davis 2008].

Para permitir que uma URL do BeShort seja identificada e transformada novamente em sua versão longa original, é necessário uma forma de distinguir uma URL encurtada pelo BeShort, dos demais caracteres postados nas mensagens nas quais o BeShort foi compartilhado. Sendo assim, definimos como identificador de uma URL encurtada pelo BeShort o radical "b://". Como exemplo, suponha que os termos "*http://www.*", "*decom*", "*ufop*" e "*.br*" sejam traduzidos respectivamente pelos caracteres: $\heartsuit$, ∇, $\bowtie$, e Ψ. Sendo assim, a URL *http://www.decom.br* seria traduzida pelo BeShort para b://$\heartsuit\nabla \bowtie \Psi$.

Finalmente, definir quais os termos devem ser substituídos por caracteres do

padrão UTF-8 não é um tarefa trivial, visto que, termos de tamanhos diferentes podem possuir popularidades diferentes. Como exemplo, qual é a melhor estratégia, utilizar um caractere UTF-8 para o termo ".com" e outro para o termo ".br", ou devemos utilizar um único caractere UTF-8 para ".com.br"? A seguir, apresentamos nossa abordagem para a definição dos termos que compõem o dicionário para tradução de URLs.

3.1. Construção do dicionário

Para a construção do dicionário foram estudados algoritmos de compressão e de criação de dicionários dinâmicos. Dentre eles, o que mais se aproxima do nosso interesse é o Algoritmo de Huffman baseado em palavras, que é eficaz em texto com linguagem natural [Ziviani 2004]. Entretanto, o algoritmo de Huffman não é apropriado para a criação do dicionário, pois às URLs não possuem as mesmas características das linguagens naturais, não havendo, em virtude disso, como determinar um padrão claro de separadores que marcam o início e termino das palavras. Assim, não é possível determinar as palavras que estão contidas em uma URL. Por isso, decidiu-se desenvolver uma nova abordagem para a criação de um dicionário fixo.

Nossa abordagem para a construção de um dicionário fixo é composta de três etapas importantes. Primeiramente, precisamos definir um tamanho para o dicionário, visto que existe um número limitado de caracteres no padrão Unicode que podem ser utilizados para a tradução de termos encontrados nas URLs (Seção 3.2). Em seguida precisamos gerar um grande número de termos (ex.: subpalavras existentes nas URLs) candidatos a fazer parte do dicionário (Seção 3.3). Por fim, selecionamos os termos para compor o dicionário (Seção 3.4). As próximas seções detalham nossa abordagem para cada uma dessas etapas.

3.2. Definição do tamanho do Dicionário

O padrão Unicode contém **1.114.112** caracteres, dos quais os **65.536** correspondem aos caracteres utilizados nos principais idiomas do mundo e podem ser facilmente encontrados em bibliotecas de diferentes linguagens de programação [The Unicode Consortium 2012]. As URLs comuns utilizam como caracteres apenas 128 dos 256 existentes no padrão ASCII, que não podem ser empregados na substituição dos termos do dicionário.

Desse modo, vamos considerar dois tamanhos de dicionários. O primeiro, que chamaremos de UTF-Parcial, considera como entrada do dicionário apenas os caracteres utilizados nos principais idiomas do mundo com exceção dos caracteres presentes no parão ASCII, contendo 65.408 caracteres. Note que o UTF-Parcial pode representar uma opção mais viável para implementação por consumir menos recursos (i.e. memória). O segundo, que chamamos de UTF-Total, corresponde a todas as possíveis entradas do padrão Unicode: $1.113.984$. Esses valores não correspondem aos mostrados anteriormente, pois, foram subtraídos os 128 caracteres utilizados em URLs, evitando, assim, que um determinado padrão não seja confundido com uma porção da URL que não foi traduzida.

3.3. Definição dos termos candidatos do dicionário

Para a criação dos termos candidatos, foi definido um algoritmo que recebe como entrada um conjunto de URLs, denominado conjunto de treino T. Em seguida, fazemos a

extração de todos os potenciais termos (subpalavras) das URLs de T, com tamanho $i\ |2 \leq i \leq M$, onde M é o valor máximo para o tamanho de um termo, definido empiricamente e discutido na Seção 5.4. O tamanho começa em 2, pois é o menor tamanho em que ainda se pode ter um ganho na compressão. Assim, para cada valor de i, o algoritmo extrai todos termos com esse tamanho da URL u, retirada da base de treino T.

Ao percorrer o conjunto T, a frequência de ocorrência dos termos identificados é contabilizada. Os termos identificados e suas frequências são armazenados no conjunto S. Após ter gerado todos os termos e contabilizado todas as frequências, é realizado uma ação com os termos do conjunto S. Esta ação é realizada para gerar o conjunto final F, que contém os termos e suas respectivas características, que são, frequência, tamanho (número de caracteres) e a multiplicação da frequência pelo tamanho do termo. Os termos das URLs gerados em F são considerados termos candidatos a formarem o dicionário. Com isso o conjunto F de saída do nosso algoritmo será utilizado pelas diferentes estratégias de seleção discutidas na Seção 3.4. Os passos para a construção dos termos candidatos estão descritos no Algoritmo 1.

Algoritmo 1: Algoritmo para criação dos termos candidatos ao dicionário

```
Entrada: Entrada: Conjunto T de URLs;
Saída: Conjunto F de termos com suas características
tamanho do termo i = 2;
enquanto i < M faça
    enquanto existir URL u em T faça
        enquanto u não atingir o fim faça
            Para cada termo s de tamanho i em u;
            se s ∉ S então
                Insere s em S e inicia o valor da frequência de s igual a 1;
            senão
                Soma 1 à frequência de s;
            fim se
        fim enqto
    fim enqto
fim enqto
enquanto existir termo s em S faça
    Insere s em F, junto a sua frequência, tamanho e frequência × tamanho;
fim enqto
```

3.4. Estratégias para seleção de termos

Após definida a quantidade de termos que irão compor o dicionário e ter gerado o conjunto F de termos candidatos na segunda etapa, precisamos selecionar os termos que efetivamente farão parte do dicionário. A seguir são apresentadas cinco estratégias para seleção desses termos.

1. **Aleatório** - Os termos são selecionados de F de forma aleatória.
2. **Tamanho** - São selecionados o termos de F que tiverem maior tamanho.
3. **Frequência X Tamanho** ($Freq \times Tam$) - A seleção é feita com os termos de F que tiverem o maior resultado da operação $frequencia \times tamanho$.

4. **Frequência** - Os termos de F que tiverem os valores mais altos de frequência são selecionados.
5. **Frequência menos Subpalavra** ($Freq-Sub$) - Primeiro o conjunto F é ordenado em ordem decrescente da frequência dos termos. Em seguida, excluímos os termos de menor frequência que são subpalavras de termos com maiores frequências. Como exemplo, se o termo "http://www"é mais frequente que o termo "http://", esse último não é incluído no dicionário visto que ele é uma subpalavra do primeiro e é menos frequente.

A seleção de termos é feita até que alcance os tamanhos de dicionário definidos na Seção 3.2.

4. Coleção de URLs encurtadas

Para testarmos nossa abordagem, precisamos construir uma ampla coleção de URLs encurtadas postadas em sistemas sociais, como o Twitter. Mais importante, precisamos obter a versão longa dessas URLs encontradas, de forma a permitir a avaliação da eficácia do mecanismo de compressão de URLs proposto.

Nossa abordagem consiste em extrair URLs existentes em uma grande coleção de *tweets* obtida em um trabalho anterior [Cha et al. 2010] e depois traduzir essas URLs para suas versões originais. Essa coleção possui **54.981.152** usuários do Twitter, todos os elos de seguidores e seguidos (grafo com **1.963.263.821** arestas) e todos os *tweets* postados por esses usuários (**1.755.925.520** *tweets*). Esses *tweets* correspondem a todos os *tweets* já postados por todos os usuários do Twitter até o período da coleta, ou seja, desde a criação do Twitter em 2006 até julho de 2009. Essa base de dados é apropriada para o nosso propósito por se tratar de uma coleta quase completa do Twitter em um período e não de uma amostra potencialmente tendenciosa.

Visando traduzir as URLs encurtadas encontradas nos *tweets* para suas versões longas, foi desenvolvida uma ferramenta capaz de resolver à URL de um *tweet*. O sistema envia uma requisição de GET para cada URL e identifica o redirecionamento, que é o mecanismo utilizado por sistemas encurtadores. Ao detectar um redirecionamento, a versão longa original da URL é armazenada junto à versão curta. Esse procedimento foi realizado para todas as URLs encontradas na base e considerou-se que um serviço encurtador de URLs foi utilizado quando o domínio obtido era diferente e a URL era maior do que a versão encurtada. Sendo assim, ao final desse procedimento foram obtidos diversos pares de URLs curtas e suas versões longas. No total identificamos **67.324.019** pares de URLs. Partindo desses pares de URLs, os domínios das URLs curtas foram ranqueados e os mais populares passaram por uma investigação manual, ou seja, forma abertos em um navegador, de forma que 77 diferentes encurtadores foram identificados. Esses encurtadores foram responsáveis pelo encurtamento de **57.763.943** (86% das URLs). Dentre os 77 serviços encontrados, os mais populares foram o Bit.ly e o TinyURL, que juntos representam mais de 87% das URLs de todos os serviços identificados.

5. Avaliação Experimental

Nesta seção inicialmente fazemos uma análise das estratégias de seleção de termos para o dicionário. Posteriormente, apresentamos os experimentos realizados com

o BeShort que medem sua capacidade de compressão e comparamos os resultados aos obtidos pelos sistemas Bit.ly e TinyURL.

Para a realização dos experimentos utilizamos duas bases, cada uma contendo 1 milhão de URLs dos serviços de encurtamento Bit.ly e TinyURL que foram extraídas aleatoriamente de uma base de dados que contém todas as URLs extraídas dos tweets. Para construir o dicionário utilizamos 500 mil URLs encontradas no sistema do Bit.ly e outras 500 mil encontradas no sistema do TinyURL. A partir dessa base, as URLs foram partidas em tamanhos de no máximo 15 caracteres, variável M do algoritmo. Para a construção desse dicionário foi utilizada uma máquina Linux com um processador AMD Phenom II e 4 GB de memória.

Alguns resultados utilizam o termo "porcentagem de encurtamento", que representa a porcentagem de caracteres que diminui da URL longa quando esta é encurtada. Cabe ressaltar que nessas porcentagens de encurtamento foi considerada a parte fixa da URL encurtada (ex. b://, www.bit.ly/, www.tinyurl.com/). A seguir, a seção 5.1 compara as estratégias de seleção dos termos para o dicionário. A seção 5.2 mostra os principais resultados relacionados à eficiência da porcentagem de compressão. A seção 5.3 estuda o impacto do tamanho da URL na compressão. Por fim, a seção 5.4 explora parâmetros da construção do dicionário do BeShort.

5.1. Análise das estratégias de seleção dos termos

As comparações entre as estratégias foram feitas em função da porcentagem de encurtamento alcançada para cada URL. Para gerar as porcentagens, selecionamos 1 milhão de URLs longas de forma aleatória, retiradas das bases do Bit.ly e do TinyURL. Os resultados do experimento são mostrados nos gráficos da figura 4, onde podemos visualizar a distribuição de probabilidade cumulativa (CDF) das porcentagens de encurtamento de cada uma dessas estratégias.

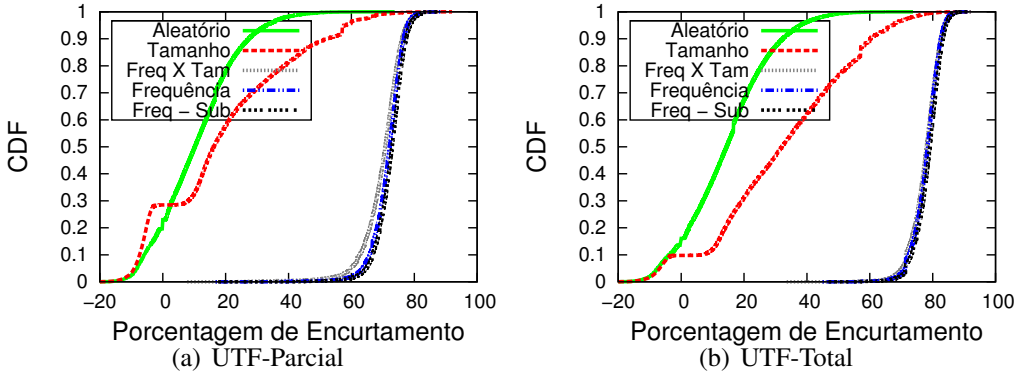


Figura 4. Compressão das estratégias de seleção dos termos

Analisando o gráfico da figura 4(a), no qual foi utilizado o tamanho de dicionário UTF-Parcial, percebemos que as estratégias *Aleatório* e *Tamanho* foram bem inferiores se comparadas às outras. Como exemplo, enquanto 98% das URLs obtiveram mais que 60% de encurtamento para a estratégia *Frequência*, apenas $0,1\%$ e 4% das URLs conseguiram taxas de encurtamento superiores aos mesmos 60% para as estratégia *Aleatório* e $Tamanho$. Comparando as estratégias $Freq{\times}Tam$, *Frequência* e $Freq{-}Sub$, observamos que elas estão bem próximas porém, a estratégia $Freq{-}Sub$ possui um ganho

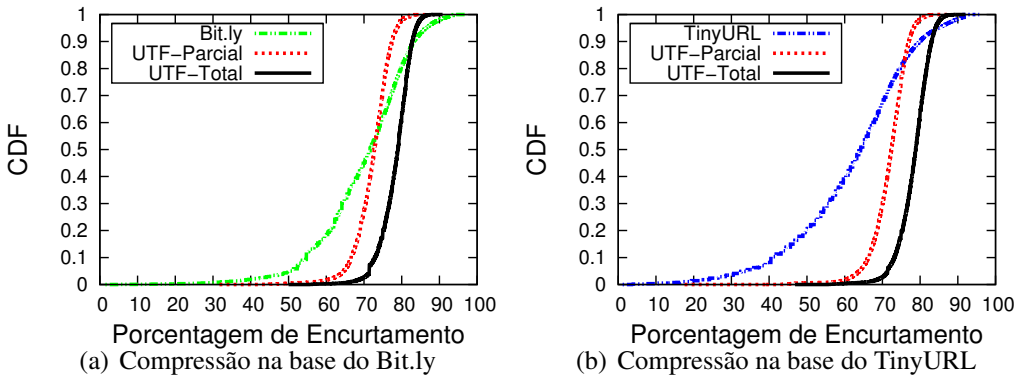
marginal em relação às outras. Como exemplo, 80% das URLs obtiveram taxas de encurtamento superiores a 69% para estratégia $Freq{-}Sub$, enquanto que nas estratégias $Freq{\times}Tam$ e *Frequência*, 80% das URLs obtiveram taxas de encurtamento superiores a 66% e 68%, respectivamente. Analisando a média das porcentagens de encurtamento, percebe-se que a estratégia $Freq{-}Sub$ foi superior às outras com valor de $72,40\%$, enquanto a estratégia $Freq{\times}Tam$ foi $69,81\%$ e a estratégia *Frequência* foi $71,49\%$.

No gráfico da figura 4(b), foi utilizado o tamanho de dicionário UTF-Total. Neste gráfico podemos observar resultados bastante semelhantes aos anteriores para o dicionário UTF-Parcial e reforçar que a estratégia $Freq{-}Sub$ é superior às demais. Sendo assim, vamos utilizá-la na criação do dicionário para os próximos experimentos.

5.2. Análise de Compressão

Nesta seção o BeShort é testado em relação a porcentagem de encurtamento das URLs, e seus resultados são comparados aos serviços Bit.ly e TinyURL. Os gráficos das figuras 5 mostram a distribuição de probabilidade cumulativa (CDF) da porcentagem de encurtamento de cada um desses serviços. O gráfico da figura 5(a) compara o BeShort com o Bit.ly, sendo que, para o BeShort foram utilizados os dois tamanhos de dicionário anteriormente discutidos: UTF-Total e UTF-Parcial. Apesar das três curvas estarem próximas e se cruzarem, podemos observar que as três abordagens conseguem resultados de compressão competitivos. Boa parte das URLs dos três sistemas obtiveram porcentagens de encurtamento entre 60% e 80% de encurtamento. Podemos notar que em alguns casos, o BeShort é superior ao Bit.ly. Como exemplo, cerca de 80% das URLs obtiveram taxas de encurtamento superiores a 75% com o dicionário UTF-Total, enquanto que apenas cerca de 40% das URLs encurtadas com o Bit.ly conseguiram porcentagens tão altas.

A competitividade do BeShort com as arquiteturas centralizadas fica ainda mais evidente na comparação com o TinyURL. O gráfico da figura 5(b) apresenta a mesma análise, porém utiliza a base de URLs do TinyURL e mostra uma comparação com o encurtamento desse serviço. Enquanto 90% das URLs encurtadas com o BeShort utilizando UTF-Total obtiveram porcentagens de encurtamento superiores a 73%, ao passo que apenas 25% dos encurtamentos do TinyURL conseguiram porcentagens acima desse valor.



(a) Compressão na base do Bit.ly (b) Compressão na base do TinyURL

Figura 5. Compressão do BeShort nas bases do Bit.ly e TinyURL

Comparando os resultados do encurtamento do BeShort com o dicionário UTF-Parcial e UTF-Total, podemos notar que o UTF-Total é melhor do que o UTF-Parcial, porém ambos são competitivos. A vantagem do uso do UTF-Parcial é que ele pode simplificar a implantação do BeShort por utilizar caracteres UTF-8 normalmente suportados por bibliotecas de linguagens de programação e normalmente aceitas em navegadores e outros programas. Além disso, o UTF-Parcial reduz drasticamente a quantidade de memória necessária, o que pode ser essencial para o caso de implementar o BeShort em dispositivos móveis.

Em sumário, os resultados dessa seção mostram que o BeShort é superior ao Bit.ly e TinyURL na maioria dos casos, e que o BeShort com o dicionário UTF-Total alcança resultados melhores comparados aos resultados do BeShort com dicionário UTF-Parcial.

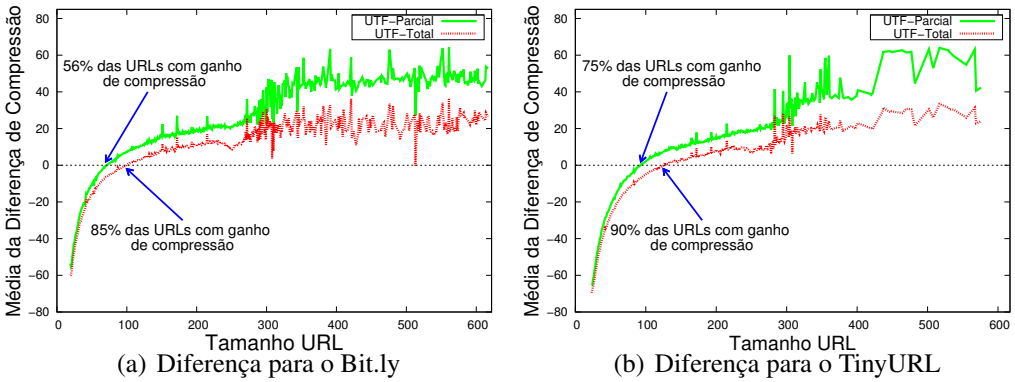
5.3. Impacto do Tamanho da URL

Com o intuito de analisar como o tamanho da URL pode afetar as porcentagens de encurtamento do BeShort e dos serviços Bit.ly e TinyURL, realizamos a seguinte análise. Para cada tamanho X das URLs longas da nossa base de testes, calculamos a média de compressão para este tamanho. Em outras palavras, nós calculamos a porcentagem de encurtamento obtidas por BeShort UTF-Total, BeShort UTF-Parcial, Bit.ly e TinyURL para as URLs da base de dados, agrupamos esses resultados de acordo com o tamanho da URL e computamos o valor médio para cada tamanho de URL.

Os gráficos da figura 6 mostram a diferença das médias de compressão em função do tamanho das URLs longas. Comparando o BeShort com os outros serviços. Quando o resultado é um valor negativo significa que o BeShort obteve um encurtamento melhor e quando o valor é positivo significa que os serviços, Bit.ly ou TinyURL alcançaram melhores resultados.

O gráfico da figura 6(a) apresenta uma comparação da diferença das médias do Bit.ly com o Beshort, utilizando os dois tamanhos de dicionário. Com o dicionário UTF-Total, o BeShort perde para URLs com tamanho superior a 100 caracteres. Por outro lado o UTF-Parcial começa a perder para o Bit.ly a partir de URLs com 72 caracteres. Analisando o segundo gráfico da figura 6, mostramos a mesma comparação, mas agora entre o TinyURL e o BeShort, que está utilizando os mesmos dois tamanhos de dicionários. Pode-se perceber que, com o UTF-Total o BeShort não foi superior ao TinyURL apenas para a compressão de URLs de tamanho acima de 125 caracteres. Com o UTF-Parcial este valor cai para 93 caracteres.

Com essas análises podemos notar que o BeShort é, em geral, mais efetivo do que os serviços Bit.ly e TinyURL para URLs de tamanho menor. Também é importante ressaltar que, mesmo apresentando resultados piores para URLs com um número elevado de caracteres, o BeShort ainda consegue resultados competitivos comparando-o com as arquiteturas centralizadas. De maneira geral, nos casos em que o BeShort perde, as diferenças das médias ficam em sua maioria na casa dos 20%, o que ainda torna o BeShort viável. Além disso, foi observado em nossas análises que, em média 18% das URLs possuem tamanho maior que 100 e apenas 7% excedem os 140 caracteres. Em uma breve inspeção manual, analisamos 50 URLs com tamanhos superiores a 100 caracteres e notamos que elas, em geral, correspondem a endereços contendo mapas ou informações relativas a sessões de usuários.



(a) Diferença para o Bit.ly (b) Diferença para o TinyURL

Figura 6. Diferença de compressão entre BeShort e os demais serviços

5.4. Tamanho Máximo dos Termos

Finalmente, um parâmetro importante do nosso algoritmo para a criação do dicionário é o tamanho máximo dos termos utilizados. Os três gráficos mostrados nas figuras 7 e 8 oferecem análises referentes a esse parâmetro, e medem o quanto o tamanho do termo pode influenciar na porcentagem de encurtamento. O gráfico da figura 7 mostra a média de encurtamento obtida as URLs da base em função do tamanho máximo do termo. O primeiro tamanho máximo avaliado começa com 4 pois, com os tamanhos 2 e 3 não foi possível criar um dicionário com o número de termos que pudesse preencher o dicionário com UTF-Total, ou seja, **1.114.112** termos. Pode-se observar que a média de compressão é crescente entre os tamanhos máximos 4 e 8, e após este valor a média começa a estabilizar.

De fato, os gráficos das figuras 8(a) e 8(b) mostram a distribuição de probabilidade cumulativa (CDF) da porcentagem de encurtamento variando o tamanho do termo na construção do dicionário do BeShort. Tanto no UTF-Parcial quanto no UTF-Total podemos perceber que as curvas estão muito próximas para os valores 10 e 15.

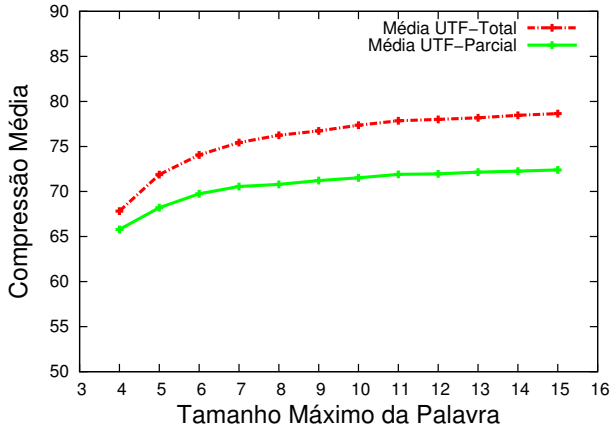


Figura 7. Média de compressão à medida que varia o tamanho máximo do termo

A partir das análises mencionadas, podemos concluir que se o tamanho máximo do termo for próximo a 8 já é suficiente para obter resultados tão precisos quanto os apresentados nas seções anteriores, calculados com tamanho máximo igual a 15.

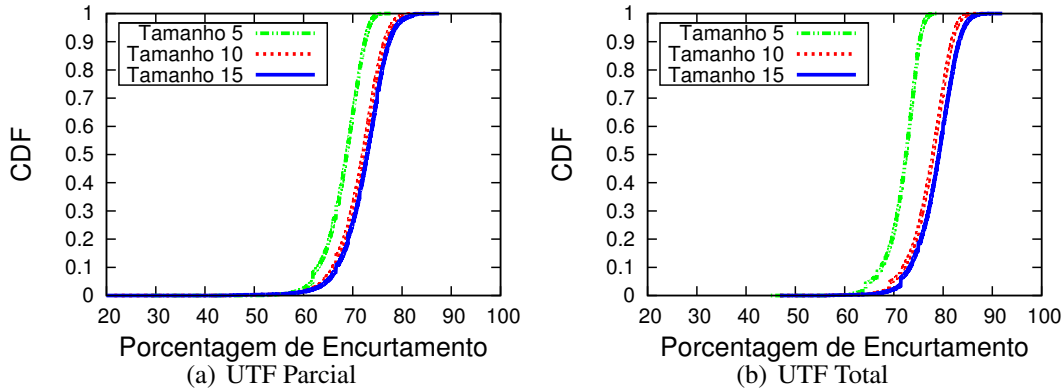


Figura 8. Compressão em Função do Tamanho Máximo do Termo

6. Conclusão

Neste trabalho analisamos a viabilidade de uma abordagem descentralizada para encurtar URLs. Esta abordagem executa um algoritmo de encurtamento de URL no momento do envio da mensagem à rede social que, ao ser recebida, é expandida para sua forma original e exibida aos usuários. Nossa abordagem é baseada na substituição de termos (partes da URL) frequentes encontrados em URLs por caracteres UTF-8, onde esse caracteres podem ser enviados através de APIs de redes sociais e normalmente não são encontrados em URLs.

Nossos resultados mostram que em relação as taxas de compressão nossa abordagem é competitiva aos serviços praticados atualmente. Com o BeShort cerca de 80% das URLs analisadas obtiveram taxas de encurtamento superiores a 75%, já no Bit.ly cerca de 40% das URLs conseguiram a mesma marca. Comparando o BeShort ao TinyURL os resultados são ainda mais favoráveis. Cerca de 90% das URLs encurtadas com o BeShort obtiveram porcentagens de encurtamento superior a 73%, ao passo que apenas 25% dos encurtamentos do TinyURL conseguiram porcentagens acima desse valor. O BeShort se mostrou inferior ao Bit.ly e ao TinyURL para URLs com um número elevado de caracteres, mas foi observado em nossas análises que em média 18% das URLs possuem tamanho maior que 100 e apenas 7% excedem os 140 caracteres. Essas análises sugerem que nossa abordagem é viável como estratégia de encurtamento.

Agradecimentos

Esse trabalho é apoiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) através do projeto universal número 472594/2012-5.

Referências

Antoniades, D., Polakis, I., Kontaxis, G., Athanasopoulos, E., Ioannidis, S., Markatos, E. P., and Karagiannis, T. (2011). we.b: The web of short urls. In *ACM Int'l conference on World Wide Web (WWW)*, pages 715–724.

Benevenuto, F., Magno, G., Rodrigues, T., and Almeida, V. (2010). Detecting spammers on twitter. In *Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*.

Benevenuto, F., Rodrigues, T., Almeida, V., Almeida, J., and Gonçalves, M. (2009). Detecting spammers and content promoters in online video social networks. In *Int'l ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 620–627.

Cha, M., Haddadi, H., Benevenuto, F., and Gummadi, K. P. (2010). Measuring User Influence in Twitter: The Million Follower Fallacy. In *Int'l AAAI Conference on Weblogs and Social Media (ICWSM)*.

Chhabra, S., Aggarwal, A., Benevenuto, F., and Kumaraguru, P. (2011). Phi.sh/$social: The phishing landscape through short urls. In *Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*.

Costa, H., Benevenuto, F., and Merschmann, L. (2013). Detecting tip spam in location-based social networks. In *28th Annual ACM Symposium on Applied Computing*.

Davis, M. (2008). Moving to unicode 5.1. `http://googleblog.blogspot.com.br/2008/05/moving-to-unicode-51.html#!/2008/05/moving-to-unicode-51.html`. Acessado em Dezembro/2012.

Facebook (2012). Facebook Press Room, Statistics. `http://www.facebook.com/press/info.php?statistics`. Acessado em Novembro/2012.

Gao, H., Hu, J., Wilson, C., Li, Z., Chen, Y., and Zhao, B. Y. (2010). Detecting and characterizing social spam campaigns. In *ACM Int'l Conference on Internet Measurement (IMC)*, pages 35–47.

Gonçalves, P., Dores, W., and Benevenuto, F. (2012). Panas-t: Uma escala psicométrica para medição de sentimentos no twitter. In *Brazilian Workshop on Social Network Analysis and Mining (BraSNAM)*.

Grier, C., Thomas, K., Paxson, V., and Zhang, M. (2010). @spam: the underground on 140 characters or less. In *17th ACM conference on Computer and communications security*, New York, NY, USA.

Hughes, A. L. and Palen, L. (2009). Twitter adoption and use in mass convergence and emergency events. In *2009 ISCRAM Conference*.

Klien, F. and Strohmaier, M. (2012). Short links under attack: Geographical analysis of spam in a url shortener network. In *23rd Conference on Hypertext and Social Media, HT' 2012*, Milwaukee, Wisconsin, USA.

Lee, K., Caverlee, J., and Webb, S. (2010). Uncovering social spammers: social honeypots + machine learning. In *33rd international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA.

Lee, K., Eoff, B. D., and Caverlee, J. (2011). Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter. In *AAAI Int'l Conference on Weblogs and Social Media (ICWSM)*.

Markines, B., Cattuto, C., and Menczer, F. (2009). Social spam detection. In *Int'l Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 41–48.

Obama, B. (2012). Utilizando twitter para campanha de 2012. `http://twitter.com/BarackObama`. Acessado em Novembro/2012.

PRLOG (2010). Just how many url shorteners are there anyway? http://www.prlog.org/10879994-just-how-many-url-shorteners-are-there-anyway.html. Acessado em Novembro/2012.

Rao, L. (2010). Twitter seeing 90 million tweets per day, 25 percent contain links. http://techcrunch.com/2010/09/14/twitter-seeing-90-million-tweets-per-day/. Acessado em Novembro/2012.

Report, N. O. (2009). Social networks & blogs now 4th most popular online activity. http://tinyurl.com/cfzjlt. Acessado em Março/2010.

Rodrigues, T., Benevenuto, F., Cha, M., Gummadi, K. P., and Almeida, V. (2011). On word-of-mouth based discovery of the web. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 381–393.

Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes twitter users: Real-time event detection by social sensors. In *In Nineteenth International WWW Conference. ACM*.

Sutton, J., Palen, L., and Shklovski, I. (2008). Backchannels on the front lines: Emergent uses of social media in the 2007 southern california wildfires. In *5th International ISCRAM Conference*.

The Unicode Consortium, editor (2012). *The Unicode Standard, Version 6.1 — Core Specification*. The Unicode Consortium, Mountain View, CA. http://www.unicode.org/versions/Unicode6.1.0/.

Thomas, K., Grier, C., Song, D., and Paxson, V. (2011). Suspended accounts in retrospect: an analysis of twitter spam. In *ACM SIGCOMM conference on Internet measurement conference, IMC '11*, New York, NY, USA.

Vasconcelos, M., Ricci, S., Almeida, J., Benevenuto, F., and Almeida, V. (2012). Tips, Dones and To-Dos: Uncovering User Profiles in FourSquare. In *ACM Int'l Conference on Web Search and Web Data Mining (WSDM)*.

Ziviani, N. (2004). *Projeto de algoritmos: com implementações em Pascal e C*. Thompson, São Paulo.