Uma abordagem descentralizada para encurtamento de URLs

Wellington Dores, Pedro Freitas, Fabrício Benevenuto [1]

[1]Departamento de Ciência da Computação
Universidade Federal de Ouro Preto (UFOP)
Campus Universitário Morro do Cruzeiro
Ouro Preto, MG, Brasil

{wellingtonjdores, pedropufop, benevenuto}@gmail.com

Abstract. *Twitter is a social system designed solely to allow users to post messages containing no more than 140 characters. With the widespread use of short messages on the Web, the use of URL shorteners are increasingly becoming popular. 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 the BeShort, a decentralized algorithm for shortening URLs able to avoid such problems. The approach have obtained a competitive compression rate in a decentralized and transparent to the user.*

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. Esses encurtadores traduzem uma URL em uma nova, com poucos caracteres. Apesar de eficientes, esses serviços podem introduzir atrasos e têm sido amplamente utilizado para ofuscar spam. Esse trabalho apresenta o BeShort, um algoritmo para encurtamento de URLs de forma descentralizada capaz de evitar tais problemas. De forma descentralizada e transparente ao usuário. Os resultados contidos no trabalho mostram que o BeShort consegue taxas de encurtamento tão eficientes quanto as taxas praticadas por arquiteturas centralizadas.*

1. Introdução

Desde seu início a Internet tem sido palco de uma série de novas aplicações como a WWW e o e-mail. Atualmente, a Web tem recebido uma nova onda de aplicações diretamente associadas com o crescimento e proliferação das redes sociais online. De acordo com a Nielsen Online [Report], a mídia social já superou a troca de e-mails como a atividade online mais popular. Mais de dois terços de toda população online no mundo visita ou participa de redes sociais e blogs. No Twitter, um sistema que permite unicamente a postagem de mensagens curtas (tweets), trafegam cerca de 150 milhões de mensagens por dia, cada uma enviada a milhares de usuários.

O uso de mensagens curtas tem sido amplamente explorado em sistemas como o Twitter, permitindo que os usuários discutam sobre tudo, incluindo notícias, casualidades, suas opiniões, repercussão de eventos ou produtos [Cha et al. 2010]. Milhões de URLs são compartilhadas todos os dias nesses sistemas, mudando a forma como as pessoas descobrem conteúdo na Web [Rodrigues et al. 2011]. Várias redes sociais impõem um

limite para o tamanho da mensagem (ex. no Twitter a mensagem é limitada a 140 caracteres). Devido a essa limitação, seus usuários passaram a utilizar cada vez mais, sistemas encurtadores de URLS para assim não ocupar o espaço de suas mensagens apenas com o endereço de uma URL.

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. Sistemas encurtadores de URL 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]. Contudo, apesar de úteis, serviços encurtadores de URL podem trazer alguns problemas para seus usuários, que serão apresentados a seguir.

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 distante do usuário. 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 dos serviços encurtadores de URL não oferecem garantias sobre a qualidade de seus serviços e podem até mesmo não atender requisições, caso estejam sobrecarregados.

Facilidade para phishing e spam: Serviços encurtadores de URL vem sendo comumente utilizados como forma de esconder spam e phishing, o que vem sendo reportado em um grande número de trabalhos científicos recentes como por exemplo [Benevenuto et al. 2010, Chhabra et al. 2011, Zhang and Paxson 2011, Grier et al. 2010, Lee et al. 2011]. Em particular, phishers utilizam URLs encurtadas para ofuscar suas URLs, conforme mostrado recentemente em [Chhabra et al. 2011].

Assim, surge a necessidade do desenvolvimento de uma arquitetura que seja capaz de encurtar URLs de forma a evitar os problemas apontados acima. Nesse artigo pretende-se investigar o uso de uma abordagem descentralizada para o encurtamento de URLs dispensando assim, o uso de redirecionamento para um servidor central. A ideia é que um algoritmo de encurtamento de URL seja executado no momento de envio da mensagem na rede social e, ao ser recebida, o algoritmo tornasse a descomprimi-la. De uma forma totalmente transparente para os usuários.

O presente artigo faz parte de um projeto de iniciação científica PIBIT/UFOP/CNPq do qual, o aluno de graduação do curso de ciência da Computação da Universidade federal de Ouro Preto Wellington J. Dores é bolsista e foi responsável pelas análises aqui apresentadas. Sendo o idealizador do algoritmo, participou ativamente na preparação das bases de URLs que contém suas versões longas e curtas e realizou os testes comparativos. Todo este trabalho foi orientado e supervisionado pelo Professor Fabrício Benevenuto, Ph.D. em Ciência da Computação, e com o apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

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

2. Trabalhos Relacionados

Recentemente, Rodrigues e colaboradores [Rodrigues et al. 2011] realizaram uma série de análises sobre os padrões de propagação de URLs entre os usuários do Twitter. Além de quantificar o aumento de audiência que o uso do reenvio de mensagens (retweets) pode causar a uma URL, eles mostram que domínios pouco populares na web podem se tornar populares através do espalhamento boca-a-boca no Twitter. O trabalho também identificou características típicas da estrutura de árvores de propagação de informação nesse sistema, e mostram que em sistemas como este as árvores são mais largas do que profundas. De maneira geral, esses trabalhos ressaltam o grande interesse por espalhamento de informação e quantificam o volume de URLs compartilhadas, sendo que a maioria dessas são postadas de forma encurtada. De fato, Antoniades e colaboradores [Antoniades et al. 2011] mostram que cerca de 87% das URLs do Twitter são encurtadas. Além disso, os autores quantificam o atraso imposto por encurtadores de URL e mostram que esse atraso pode ser significativo. Os autores mostram também que URLs encurtadas possuem vida curta em termos de popularidade. Tal observação sugere que sistemas encurtadores centralizados mantêm listas enormes de URLs que só possuem acessos em um curto intervalo de tempo.

Um aspecto relacionado à grande popularidade do uso de encurtadores de URLs é a proliferação de diferentes formas de spam na Web. Chhabra [Chhabra et al. 2011] e colaboradores estudaram o espalhamento de *phishing* através de URLs encurtadas. Através de várias análises os autores mostram que muitas URLs encurtadas por phishers tiveram pouca ou nenhuma redução em seu tamanho, sugerindo que o uso do sistema é simplesmente para ofuscar URLs para usuários. Outras evidências de spam no Twitter também foram apresentadas em outros estudos, entre eles pode-se citar, [Grier et al. 2010, Chhabra et al. 2011, Lee et al. 2011].

3. Base de Dados

A base utilizada no trabalho foi extraída de uma grande coleção de tweets, obtida em um trabalho anterior [Cha et al. 2010]. Após a busca por URLs nos tweets, em sua maioria encurtadas, foi preciso traduzir essas URLs para suas versões originais, ou seja, longas. Essa coleção possui **54.981.152** de usuários do Twitter, todos os elos de seguidores e seguidos (grafo com **1.963.263.821** de arestas) e todos os tweets postados por esses usuários (**1.755.925.520** de tweets). Esses tweets correspondem a todos os tweets já postados por todos os usuários do Twitter, desde a criação do Twitter em 2006 até julho de 2009. Essa base de dados é apropriada para o propósito deste estudo por se tratar de coleta completa do Twitter e não de uma amostra potencialmente tendenciosa. Para uma descrição mais detalhada desses dados e das técnicas empregadas para a realização da coleta desses dados, recomenda-se a seguinte referência [Cha et al. 2010].

Para traduzir URLs encurtadas encontradas nos tweets, foi desenvolvida uma ferramenta capaz de resolver a URL de um tweet. O sistema envia uma requisição de GET para cada URL encontrada no tweet e identifica os redirecionamentos, que é o mecanismo utilizado por sistemas encurtadores. Ao detectar um redirecionamento, é armazenada a versão longa original da URL junto a versão curta, formando um par. Esse procedimento foi realizado com todas as URLs encontradas na base de tweets e considerou-se que um serviço encurtador de URLs foi utilizado toda vez que o domínio obtido era diferente

e a URL era maior do que a versão encurtada. No total foi identificado **67.324.019** de URLs. Dessas, de forma manual, ou seja, abrindo a URL no navegador, encontrou-se 77 serviços encurtadores de URLs diferentes, que foram responsáveis pelo encurtamento de **57.736.943** (86% das URLs).

A tabela 1 mostra os 5 serviços encurtadores mais populares encontrados em termos de porcentagem de URLs únicas e em termos de ocorrência. Para as análises foram utilizados os dois serviços mais populares, o Bit.ly e o TinyURL, que juntos representam mais de 85% do total de URLs encurtadas pelos 77 serviços.

Tabela 1. Serviços de URL

Serviço	Porcentagem (%)	Ocorrência
Bit.ly	81,37	46.979.875
TinyURL	5,92	3.415.708
Is.gd	3,39	1.955.690
Tweetburner	2,88	1.662.796
Ow.ly	1,40	812.508
Outros 72	5,04	2.937.366

Cabe ressaltar que a compressão empregada por esses sistemas são ótimas, pois estes serviços utilizam tabelas hash para a criação das URLs encurtadas. Esses sistemas utilizam uma hash de tamanho entre 3 e 6 caracteres, a variação de tamanho é devido ao esgotamento de combinações que formam as novas URLs. Cada hash pode ser uma combinação com repetição de A-Z, a-z e 0-9, ou seja, são 62 caracteres, para uma hash de tamanho 3 seria possível encurtar 238.328 URLs.

4. Arquitetura do BeShort

Sistemas encurtadores de URL atuais utilizam uma arquitetura centralizada, como é ilustrado na figura 1. 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.

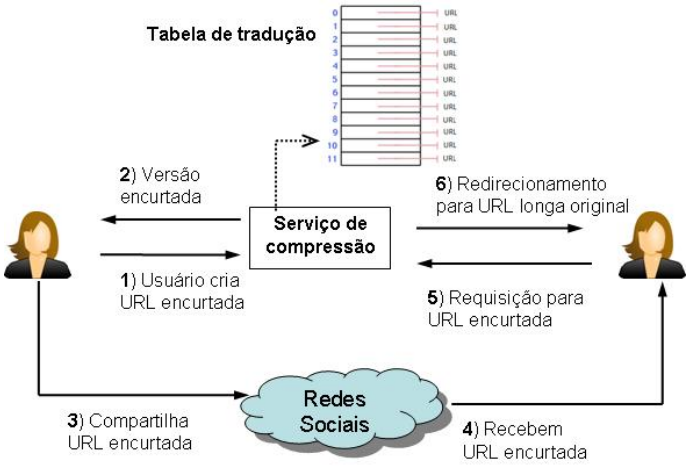


Figura 1. Arquitetura para encurtamento de URLs de forma centralizada

A figura 4 ilustra o funcionamento da arquitetura proposta, que irá se chamar BeShort. Na abordagem proposta, os usuários postam URLs em seu formato original. Essas, por sua vez, serão comprimidas ou traduzidas pelo BeShort de forma transparente para o usuário que posta ou recebe a URL.

Existem diferentes lugares onde o BeShort pode ser implantado, por exemplo nos próprios clientes através de *plugins* em navegadores ou mesmo incorporado diretamente a aplicações de redes sociais específicas. Outra alternativa é a implantação do BeShort em CNDs (content distribution networks) ou mesmo através de servidores *proxy* ativos. Este trabalho busca investigar a viabilidade dessa arquitetura descentralizada e propor uma primeira versão do algoritmo que denomina-se BeShort.

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 códigos, foi utilizado o Padrão Unicode [The Unicode Consortium 2011], 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 pode representar toda a gama de caracteres do padrão, a diferença das formas é para qual objetivo será utilizada. Para o dicionário que irá encurtar as URLs será utilizado a forma UTF-8. O Padrão Unicode contém **1.114.112** caracteres e os **65.536** primeiros representam a maioria dos caracteres comuns utilizados nos principais idiomas do mundo.

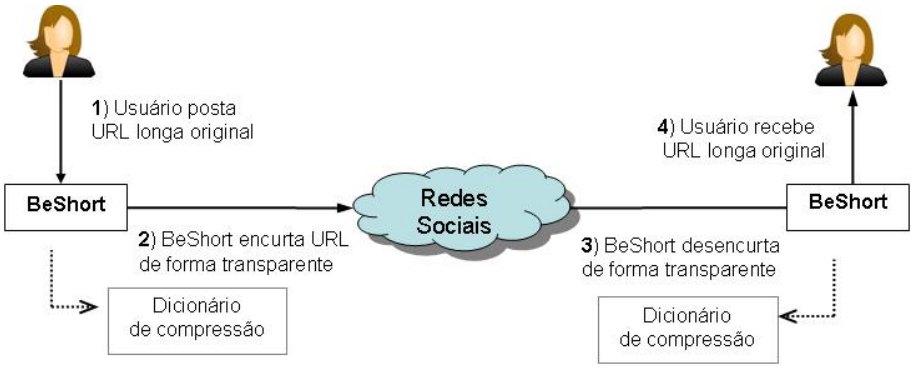


Figura 2. Arquitetura para encurtamento de URLs de forma descentralizada

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 comprimida pelo BeShort, dos demais caracteres postados em mensagens nas quais o BeShort foi compartilhado. Dentro da arquitetura, uma URL encurtada pelo BeShort será identificada pelo radical "b://" no inicio da URL. A seguir será apresentado um exemplo de compressão de uma URL através do BeShort.

Suponha um dicionário contendo os termos *http://*, *www.*, *ufop* e *.br* e que serão traduzidos, respectivamente, pelos termos: $\heartsuit$, ∇, $\bowtie$, e Ψ. Sendo assim, a URL *http://www.ufop.br* seria traduzida pelo BeShort para b://$\heartsuit\nabla \bowtie \Psi$.

Finalmente, definir exatamente quais os termos devem ser substituídos por caracteres do padrão UTF-8, que não utilizados em URLs, não é um problema trivial, visto que

termos de tamanhos diferentes podem possuir popularidades diferentes. Como exemplo, qual é a forma ótima, utilizar um caractere UTF-8 para o termo ".com" e outro para o termo ".br", ou utilizar um único caractere UTF-8 para ".com.br"?

4.1. Construção do Dicionário

Para a construção do dicionário decidiu-se por desenvolver um novo algoritmo que cria um dicionário fixo, que considera não só os termos mais frequentes, mas também o tamanho desses termos. Formando-se assim uma base com as partes que mais aparecem nas URLs.

Assim, um conjunto de URLs foi separado para determinar quais os termos que farão parte do dicionário. Tal conjunto será chamado de conjunto treino, T. Para cada URL de T, é extraído um termo que possui um tamanho i variando de 2 até M, onde M é um valor máximo definido empiricamente e discutido na seção 5.3. O tamanho começa em 2, pois, é o menor tamanho em que ainda se pode ter um ganho na compressão. Assim, para cada tamanho, o algoritmo extrai todas as combinações de tamanho i da URL u retirada da base T.

Ao percorrer a base, a frequência de ocorrência dos termos identificados é contabilizada. Próximo ao fim do algoritmo, a frequência e o tamanho i do termo são multiplicados, essa multiplicação será chamada de $peso$. A multiplicação pelo tamanho da palavra é uma maneira de conceder uma maior prioridade para as palavras maiores e mais frequentes, pois, quanto maior o tamanho da palavra, melhor será a taxa de compressão. Os passos para a construção de um ranking de termos candidatos a compor o dicionário estão descrito no Algoritmo 1.

Algoritmo 1: Algoritmo para criação dos termos do dicionário

```
Entrada: Entrada: Conjunto T de URLs;
Saída: Arquivo com os termos ranqueados
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
    f = frequência de s x |i|;
fim enqto
Ordena S em ordem decrescente de f;
```

Logo após, foi preciso definir o tamanho do dicionário. Como mencionado anteriormente, 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. As URLs utilizam como caracteres apenas 128 dos 256 caracteres existentes no padrão ASCII. Caracteres normalmente utilizados em URLs não podem ser usados na substituição dos termos do dicionário, pois haverá confusão na etapa de descompressão, pois o algoritmo não saberá se o termo foi substituído na compressão ou realmente pertence a URL.

Assim, no trabalho aqui apresentado foi estudado dois tamanhos. O primeiro, que será chamado de UTF-Parcial, é constituído de todos caracteres utilizados nos principais idioma do mundo, e contém **65.408** caracteres, o segundo contém todas as possíveis entradas do padrão unicode **1.113.984** de caracteres. Estes valores não correspondem aos mostrado anteriormente, pois, foi subtraído os 128 caracteres da tabela ASCII, normalmente utilizados em URLs, evitando-se assim o problema de identificação na descompressão.

5. Avaliação Experimental

Para a realização dos experimentos, foi utilizado duas bases, cada uma contendo 1 milhão de URLs que foram extraídas aleatoriamente de uma base de dados contendo todas as URLs extraídas dos tweets. Para construir o dicionário utilizou-se 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.

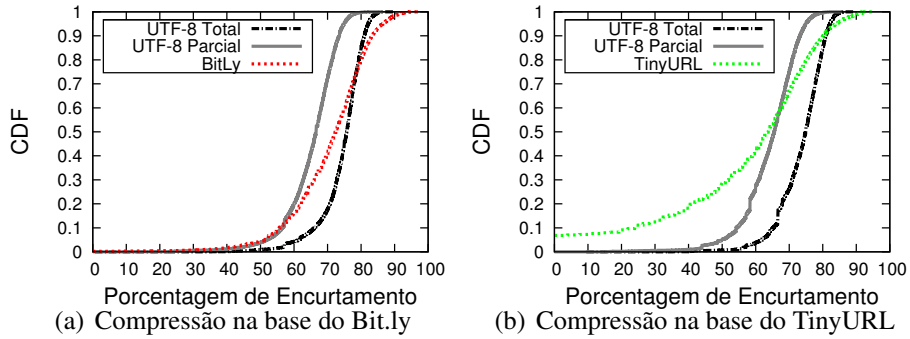
Cabe ressaltar que nos resultados de compressão apresentados, não foi excluído a parte fixa da URL encurtada (ex. b://, www.bit.ly/, www.tinyurl.com/). Em outras palavras, a compressão reportada consiste em uma simples comparação da redução no número de caracteres entre a URL longa original e a URL encurtada. A seção 5.1 reporta os principais resultados obtidos sobre a compressão, a seção 5.2 estuda o impacto do tamanho da URL na compressão e a seção 5.3 explora parâmetros da construção do dicionário utilizado pelo BeShort.

5.1. Análise de Compressão

Nesta seção o algoritmo descentralizado para encurtamento, o BeShort, será testado em relação a taxa de compressão das URLs e seus resultados serão comparados aos serviços Bit.ly e TinyURL. Os gráficos da figura 3 mostram a distribuição de probabilidade cumulativa (CDF) da porcentagem de encurtamento de cada um desses serviços. O gráfico da figura 3(a) compara o BeShort com o Bit.ly, sendo que para o BeShort foi utilizado os dois tamanhos de dicionário anteriormente discutidos, o UTF-Total e UTF-Parcial. Apesar das três curvas estarem próximas e se cruzarem, observa-se que as três abordagens conseguem resultados de compressão bons e competitivos. Boa parte das taxas de compressão dos três sistemas se concentra entre 60% e 80% de compressão. Nota-se que em alguns casos, o BeShort é superior ao Bit.ly. Como exemplo, cerca de 80% das taxas de encurtamento obtidas foram superiores a 70% de compressão com o dicionário UTF-Total, enquanto que apenas cerca 50% dos encurtamentos do Bit.ly conseguiram taxas de encurtamento tão altas.

A competitividade do BeShort com as arquiteturas centralizadas fica ainda mais evidente na comparação com o TinyURL. O gráfico da figura 3(b) apresenta a mesma análise, porém utilizando a base de URLs do TinyURL e mostrando uma comparação com a compressão desse serviço. Enquanto 90% dos encurtamentos praticados com o BeShort utilizando UTF-Total obtiveram taxas de compressão superiores a 65%, apenas 40% dos encurtamentos do TinyURL conseguiram taxas acima desse valor.

Comparando os resultados do BeShort utilizando o dicionário UTF-Parcial e o UTF-Total, percebe-se 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



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

Figura 3. Compressão do BeShort nas bases do Bit.ly e TinyURL

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 a quantidade de memória necessária para armazenamento do dicionário, o que pode ser essencial para o caso de se implementar o BeShort em dispositivos móveis.

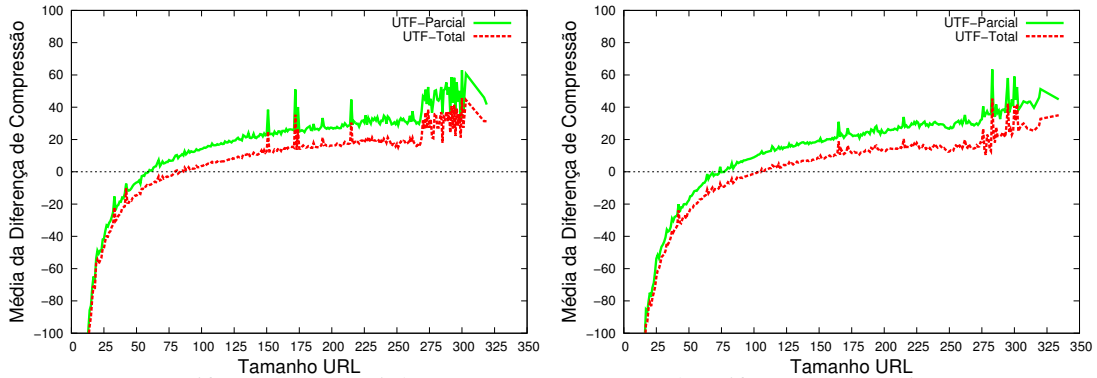
5.2. Impacto do Tamanho da URL

Com o intuito de analisar o impacto que o tamanho da URL pode afetar nas porcentagens de encurtamento do BeShort e dos serviços Bit.ly e TinyURL, a seguinte análise foi realizada. Para cada tamanho X das URLs longas da base de testes, foi calculada a média de compressão para este tamanho. Em outras palavras, calculou-se a porcentagem de compressão para o BeShort UTF-Total, BeShort UTF-Parcial, Bit.ly e TinyURL para as URLs da base. Logo após agrupando-as por seus tamanhos e dividindo pelo número de URLs correspondente ao tamanho.

A figura 4 mostra a diferença das médias de compressão em função do tamanho das URLs longas, estes gráficos comparam o BeShort com os outros serviços, Bit.ly e TinyURL. 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 4(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 84 caracteres, por outro lado o UTF-Parcial começa a perder para o Bit.ly a partir de URLs com 60 caracteres. Já o segundo gráfico da figura 4(a), mostra a mesma comparação, mas agora entre o TinyURL e o BeShort com os mesmos dois tamanhos de dicionários. Pode-se perceber que, com o UTF-Total o BeShort não obteve resultados satisfatórios na compressão para URLs de tamanho acima de 106 caracteres, já no UTF-Parcial este valor cai para 78 caracteres. Com isso nota-se que o BeShort é em geral mais efetivo do que a arquitetura do Bit.ly e TinyURL, para URLs de tamanho menor.

Após uma breve inspeção manual, notou-se que as URLs com mais de 100 caracteres, em sua maioria, correspondem a endereços contendo mapas ou informações relativas a sessões de usuários.

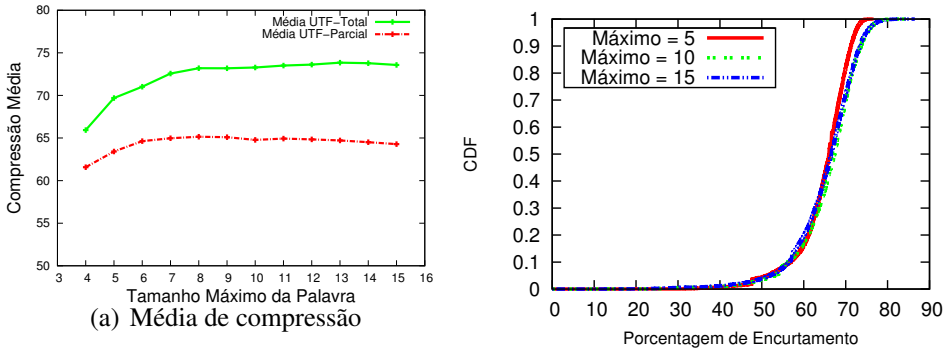


(a) Diferença para o Bit.ly (b) Diferença para o TinyURL

Figura 4. Diferença de compressão entre o BeShort e o Bit.ly. E BeShort e TinyURL

5.3. Tamanho Máximo dos Termos

Finalmente, um parâmetro importante para a criação do dicionário é o tamanho máximo dos termos utilizados. Os gráficos mostrados na figura 5 oferecem análises referentes a esse parâmetro e mede o quanto o tamanho pode influenciar na porcentagem de compressão. O gráfico da figura 5(a) mostra a média de compressão obtida para todas as URLs da base em função do tamanho da palavra. O tamanho inicial avaliado começa com tamanho 4 pois, com os tamanhos 2 e 3 não foi possível criar um dicionário com o número de palavras que pudesse preencher o dicionário UTF-Total, ou seja **1.114.112** de 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. Isso sugere que valores próximos a 8 para esse parâmetro são suficientes para obter resultados tão precisos quanto os apresentados nas seções anteriores, calculados com tamanho máximo igual a 15.



(a) Média de compressão (b) UTF Parcial

Figura 5. Compressão em Função do Tamanho Máximo da Palavra

O gráfico da figura 5(b) mostra a distribuição de probabilidade cumulativa (CDF) da porcentagem de compressão variando o tamanho da palavra na construção do dicionário do BeShort. Para as duas versões do dicionário, UTF-Parcial e UTF-Total, percebe-se que as curvas estão muito próximas para os valores 10 e 15.

6. Conclusão e Trabalhos Futuros

Este trabalho apresenta o BeShort, um algoritmo descentralizado para o encurtamento de URLs. Os resultados experimentais mostram que o BeShort consegue resultados relativamente próximos das arquiteturas centralizadas, porém sem impor os mesmos problemas existentes nessas arquiteturas. Mais importante, o trabalho mostra a viabilidade de se implementar uma arquitetura descentralizada para a solução destes problemas e apresenta as bases para a construção de uma nova abordagem.

Em trabalhos futuros pretende-se avaliar melhor o algoritmo de criação do dicionário, otimizando-o para encontrar melhores taxas de compressão. Busca-se também a criação de um protótipo do BeShort para que o mesmo seja testado também dentro das redes sociais.

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)*.

Cha, M., Haddadi, H., Benevenuto, F., and Gummadi, K. (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/$ocial: The phishing landscape through short urls. In *Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*.

Grier, C., Thomas, K., Paxson, V., and Zhang, M. (2010). @spam: the underground on 140 characters or less. In *ACM Int'l Conference on Computer and communications security (CCS)*, CCS '10, pages 27–37.

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)*.

Report, N. O. Social networks & blogs now 4th most popular online activity, 2009. `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.

The Unicode Consortium (2011). The Unicode Standard. Technical Report Version 6.0.0, Unicode Consortium, Mountain View, CA.

Zhang, C. M. and Paxson, V. (2011). Detecting and Analyzing Automated Activity on Twitter. In *Passive and Active Measurement (PAM)*.