

Utilização de *smileys* no Twitter
Relatório de projecto individual - Yahoo Pipes!

Eduardo Morais de Sousa
edsousa@gmail.com

Mestrado em Multimédia 2008/09:
 Universidade do Porto - FEUP
Laboratório II

Introdução

Quis com este trabalho elaborar uma visualização que comparasse, em duas localidades à escolha, o uso de *smileys* positivos (como :-)) com o uso de *smileys* negativos (como :-() em mensagens colocadas no Twitter. O meu objectivo é registar a evolução da disposição dos utilizadores do Twitter ao longo do dia.

Esta visualização pode ser utilizada em <http://www.asseptic.org/3M/pipes/>.

Ferramentas utilizadas

Eis as ferramentas utilizadas na produção desta visualização:

- **Twitter Search API:** <http://apiwiki.twitter.com/Twitter-Search-API-Method:-search>
Interface para a pesquisa de mensagens colocadas no Twitter.
- **Yahoo Pipes:** <http://pipes.yahoo.com/pipes/>
Ferramenta de agregação e manipulação de conteúdo na Web.
- **Google Visualization API:** <http://code.google.com/intl/pt-PT/apis/visualization/>
Interface para a criação e aplicação de visualizações em páginas Web.
- **PHP:** <http://www.php.net/>
Linguagem de programação orientada para a Web.

Funcionamento

O seguinte esquema resume o funcionamento da visualização.

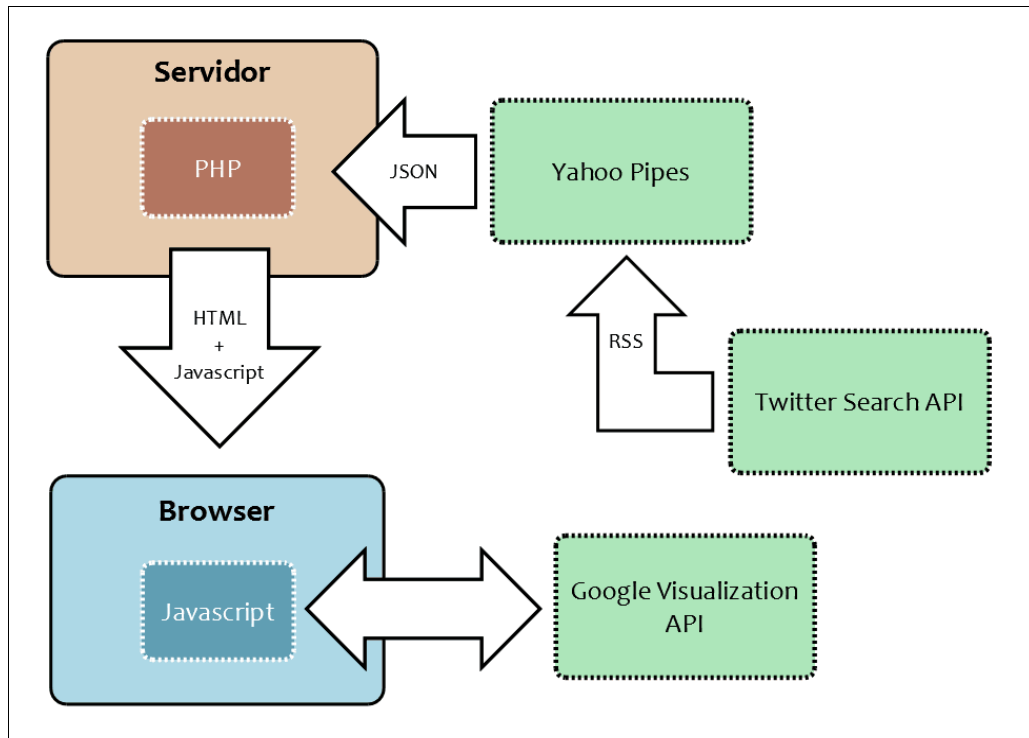


Figura 1: Funcionamento da visualização

Construção – Yahoo Pipes

O primeiro passo na construção da visualização consistiu na criação de um *mashup* no Yahoo Pipes que servisse de mediador entre a API de pesquisa do Twitter e o processamento posterior. O *mashup* acessível em <http://pipes.yahoo.com/edsousa79/twitterlocalsearch> foi criado da seguinte forma:

1. Com o módulo *URL Builder* é preparada a URL que permite interagir com a API do Twitter. De acordo com a documentação desta API, são adicionados os parâmetros *rpp* (resultados por página, num máximo de 100), *geocode* (código de latitude/longitude da área a pesquisar, e *q* (o *string* a pesquisar propriamente dito). Estes dois últimos terão de poder ser fornecidos pelo utilizador (fig 2).

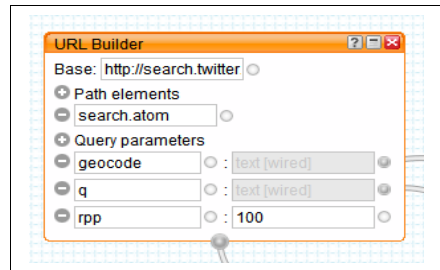


Figura 2: URL Builder

2. O *geocode* é um *string* formatado de uma determinada maneira: latitude, longitude, raio (em km). O módulo *String Builder* cria o *string* com esta formatação, recebendo a latitude e a longitude do módulo *Location Input*, que converte automaticamente uma localização (do género 'Lisboa, Portugal') inserida pelo utilizador num conjunto de variáveis, incluindo as referidas latitude e longitude (fixei o raio arbitrariamente em 20km). A saída do *String Builder* é ligada ao *geocode* do *URL Builder* referido no passo anterior.

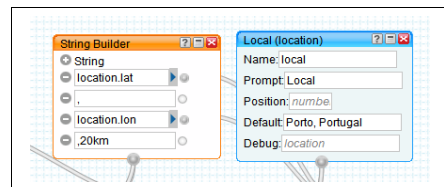


Figura 3: Location Input e String Builder

3. Criei de seguida um *Text Input*, que liguei ao parâmetro *q* (pesquisa a efectuar) do *URL Builder*. Embora a minha intenção relativamente à visualização seja pesquisar o uso de *smileys*, este *mashup* permite que sejam pesquisados quaisquer outros termos. É importante referir que o sistema de pesquisa do Twitter é bastante inteligente relativamente aos *smileys*: Uma pesquisa de ':)' devolve *tweets* contendo toda uma série de variações cujo significado é considerado positivo (ex. :-), ;), :D, XD, etc...).

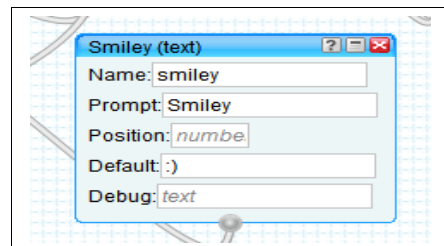


Figura 4: Text Input

4. Após alguns testes cheguei à conclusão que 100 é um número insuficiente de resultados para a visualização. Como o Twitter os devolve por ordem cronológica descendente, descobri ser frequente que os últimos 100 resultados de uma pesquisa a *tweets* do Porto ou de Lisboa se reportassem a mensagens colocadas nas últimas três ou quatro horas, insuficientes portanto para a elaboração de uma tendência diária. Decidi portanto duplicar o primeiro *URL Builder*, acrescentando o parâmetro *page* com o valor 2, de modo a devolver os cem resultados seguintes. Um total de 200 resultados já me pareceu um número mais adequado, embora seja possível alargar ainda mais o conjunto de resultados através de mais duplicados do *URL Builder*.

5. Para finalizar, ambos os *URL Builder* foram ligados a um *Fetch Feed*, que foi ligado ao *Pipe Output*. O resultado final é um *mashup* que permite fazer uma pesquisa localizada no Twitter, devolvendo os duzentos resultados mais recentes.

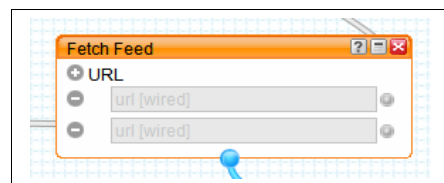


Figura 5: Fetch Feed

Eis o aspecto final do *mashup*:

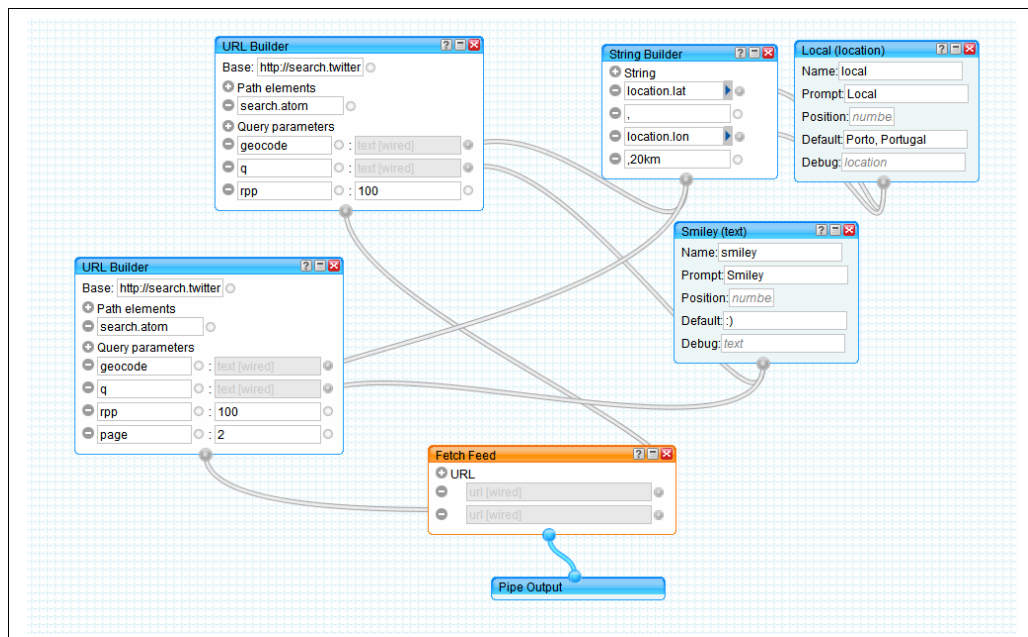


Figura 6: Yahoo Pipes – mashup para pesquisa localizada no Twitter (até 200 resultados)

Construção – PHP (1)

Uma vez que a Google Visualization API assenta em código Javascript a executar no *browser*, é possível obter resultados do Yahoo Pipes no formato JSON (Javascript Object Notation) e fazer todo o processamento em Javascript. Decidi no entanto fazê-lo em PHP no lado do servidor por me sentir mais confortável com esta linguagem.

Optei no entanto por preferir receber dados do Pipes à mesma no formato JSON apesar deste também oferecer uma saída em PHP. Esta linguagem dispõe da função `json_decode()` que transforma a informação recebida num *array* estruturado de uma forma que achei mais compreensível, de forma de seguida programar o seu processamento. Elaborei então um *script* em PHP que interage com o Pipes da seguinte forma:

1. Descobre se o utilizador inseriu localizações no formulário (entretanto construído em HTML) presente na página. Senão utiliza as duas localizações por defeito ('Porto' e 'Lisboa').

2. Acede ao *mashup* alojado no Pipes. Irá fazê-lo quatro vezes – para cada uma das duas localizações, pesquisa ora *smileys* positivos (:)), ora negativos (:(. Esta é a fase mais longa do processamento, pois o servidor web terá de se ligar repetidamente ao Yahoo Pipes (é a razão pela demora no carregamento da página com a visualização).

3. Cada uma das vezes que recebe dados, como referi no formato JSON, estes são transformados num *array*. Através da data e hora de publicação de cada item na lista de resultados, conta o número de resultados correspondentes a cada hora. Este número é registado num *array* multidimensional, indexado inicialmente por hora, e posteriormente por local e 'emoção' (smiley positivo/negativo), conforme a pesquisa que estiver a ser processada. Este *array*, contém os dados em bruto necessários à minha visualização.

Construção – Google Visualization API e PHP (2)

A minha visualização contém dois gráficos alimentados pela Google Visualization API. No primeiro, do tipo *Line Chart*, duas linhas traçam, para cada localidade, a diferença entre *tweets* positivos e *tweets* negativos nas últimas horas. Já o segundo é uma *Annotated Timeline* que discrimina separadamente o número de *tweets* positivos e negativos em cada uma das localizações.

A implementação de cada um destes gráficos é semelhante. Algum código Javascript terá de ser inserido numa página HTML (ou no caso, no *output* HTML do meu *script* PHP), sendo que partes deste código terão de ser 'customizadas'. De especial pertinência são as linhas de código que definem o conteúdo da tabela a visualizar. No meu *script*, estas porções de Javascript são geradas pelo PHP, processando o *array* de resultados obtido anteriormente conforme o gráfico a visualizar.

A visualização está acessível em <http://www.asseptic.org/3M/pipes/>.

Conclusão

A minha visualização não está de modo algum preparada para ser algo mais do que uma curiosidade. Poderia ser melhorada de várias formas, tais como:

– Através da criação de *script* que acesse frequentemente ao *mashup* alojado no Pipes, registando os resultados numa base de dados. Isto permitiria recuar na solução pouco elegante que é ter um *mashup* que pesquisa duas páginas de resultados do Twitter (na mesma insuficientes para pesquisas a 'London' ou 'New York'), além de permitir uma visualização mais rica e numa escala temporal para além das últimas horas.

– Através da implementação de todo o processamento de resultados em Javascript. Tal como referi, implementei o processamento em PHP por uma questão de conforto pessoal, embora o processamento no lado do servidor não seja de todo necessário para reproduzir a minha visualização. No entanto, a eventual criação de uma base de dados referida no ponto anterior obrigará ao processamento no servidor.

– Através da exploração de outros gráficos e outras possibilidades de visualização.