Memórias de Tradução Distribuídas

Alberto Manuel Simões, José João Almeida, and Xavier Gomez Guinovart

Departamento de Informática, Universidade do Minho
{ambs|jj}@di.uminho.pt

Universidade de Vigo
xgg@uvigo.es

Resumo Neste documento apresenta-se o conceito de memórias de tradução distribuídas, discutindo-se o seu interesse na área da tradução, bem como as vantagens que uma ferramenta de tradução pode tirar do seu uso.
É apresentada uma possível implementação de memórias de tradução distribuídas usando WebServices numa arquitectura de cooperativismo. São definidos as mensagens (API) que um serviço deste género deve implementar para que uma ferramenta de tradução possa tirar partido da colaboração entre tradutores.

1 Introdução

Na área da tradução assistida por computador usa-se memórias de tradução (MT): correspondências de frases entre duas ou mais línguas diferentes.

$$TMX \equiv \mathcal{S}_{\mathcal{L}_\alpha} \rightharpoonup \mathcal{S}_{\mathcal{L}_\beta} \times \ldots \times \mathcal{S}_{\mathcal{L}_\omega}$$

Estas memórias são usadas pelos tradutores como bases de dados onde traduções já efectuadas são armazenadas para que futuras traduções, semelhantes a outras já realizadas possam ser reutilizadas.

Para armazenar as MT cada aplicação usa o seu formato proprietário. No entanto existe um *standard* baseado em XML[8] para o intercâmbio das MT: o *Translation Memory eXchange* — TMX[7,6,4].

1.1 O Formato TMX

Uma memória de tradução TMX segue um DTD que define dois grupos distintos: cabeçalho (`header`) e o corpo (`body`).

O cabeçalho guarda meta-informação: autor, data de criação, ferramenta que o gerou e outras propriedades, todas como atributos do elemento. O atributo mais importante presente no cabeçalho é o "`srclang`" que define qual a língua original na criação da memória de tradução (semanticamente, este campo é usado para indicar qual a língua original, da qual todas as outras foram traduzidas).

O corpo do documento TMX é composto por uma sequência de unidades de tradução que correspondem a:

```
<tu>
  <tuv xml:lang="en">
    <seg>Configure window properties</seg>
  </tuv>
  <tuv xml:lang="pt">
    <seg>Configurar propriedades das janelas</seg>
  </tuv>
</tu>
```

Cada unidade de tradução (`tu`) pode conter texto em mais do que uma língua. Cada um destes textos é colocado num elemento "`seg`" dentro de um outro denominado "`tuv`". Este é identificado obrigatoriamente pelo nome de língua, com o atributo "`xml:lang`".

Embora o formato TMX suporte um conjunto mais alargado de etiquetas XML, não é importante a sua apresentação neste documento já que não são usadas para a implementação das MT distribuídas.

1.2 Conceito de MT distribuída

Para apresentar de forma mais clara o uso e implicações das MT distribuídas, consideremos duas situações:

- enquanto trabalha, o tradutor vai construindo a sua MT. No entanto, existem empresas que fornecem as suas memórias de tradução especializadas em determinada área (por exemplo, certas indústrias automóveis) e outras que as vendem (por exemplo, as empresas de software de tradução). Quando a MT é fornecida por um terceiro, esta é enviada (quer seja por correio, e-mail, ftp, etc) para o tradutor, que passa a ser seu dono[1].
- outra situação, é um grupo de tradutores que trabalham num gabinete de tradução ou numa comunidade em que mais do que um tradutor está envolvido no mesmo projecto. Nesta situação existe grande probabilidade de traduzirem porções semelhantes, não se reutilizando trabalho.
  Um bom exemplo deste tipo de comunidade é a documentação de software open-source em que cada projecto é traduzido por pessoas diferentes e que acabam por traduzir textos idênticos.

Estas duas situações podem beneficiar da existência de mecanismos de MT distribuídas[1]:

- cada fornecedor coloca as suas MT acessíveis usando uma tecnologia de partilha de dados remota, e estas passam a estar disponíveis livremente ou sujeitas a um "subscrição" ou "aluguer";
- numa comunidade de tradução torna-se possível uma melhor reutilização do trabalho realizado por cada tradutor;

[1] Embora em certos casos existam limitações no seu uso — apenas em alguns projectos, durante algum tempo, etc.

A secção seguinte explica o conceito de memória de tradução distribuída, quais os impactos do seu uso nas ferramentas de tradução e como a arquitectura de rede deve estar organizada. Termina com a definição das mensagens que um WebService para MT distribuídas deve implementar.

Na secção 3 detalha-se a implementação de um cliente e servidor de MT distribuídas. A secção seguinte termina com as conclusões que se podem retirar desta metodologia de trabalho.

2 MT distribuídas

Esta secção pretende definir a estrutura de um serviço de MT distribuídas e como deve este reagir aos diferentes tipos de pedidos que terá de responder.

2.1 Impactos na ferramenta de tradução

Quando usadas localmente, as memórias de tradução são especialmente úteis no momento da tradução: para cada frase ou segmento a traduzir, a ferramenta de apoio ao tradutor irá pesquisar a MT por esse segmento.

Este processo é realizado de forma diferente por cada ferramenta de tradução: pesquisa por um frase completa, por porções delimitadas por *markup*, ou por inclusão parcial. O processo de *matching* pode ser ainda mais complicado, como iremos ver na secção 3.2.

Num ambiente de MT distribuídas, o processo irá ser diferente. Além de consultar a sua memória de tradução local, a ferramenta irá consultar um conjunto de servidores de MT. Esta paralelização irá produzir respostas concorrentes das quais se terá de escolher uma, ou de as conciliar.

Ao optarmos pela escolha de uma resposta apenas, a ferramenta pode usar várias heurísticas, das quais salientamos:

- escolher a resposta mais rápida — se a MT local responder, a ferramenta poderá mesmo não chegar a consultar qualquer servidor de MT;
- dada uma associações de classificação ou *ranking* de servidores de MT distribuídas, escolher a resposta do servidor mais cotado;
- escolher de acordo com uma classificação da tradução — implica que cada servidor etiquete a sua tradução com uma medida de qualidade. Esta opção obrigará à adopção pelos servidores de MT de um método comum de classificação da tradução.

2.2 A arquitectura de rede

De um ponto de vista macroscópico, a arquitectura de rede para um sistema de MT distribuídas pode ser construída de duas formas diferentes, como apresentado na figura 1, correspondentes às duas situações apresentadas na introdução.

Para cada um dos casos de estudo, estamos a utilizar uma arquitectura diferente: no primeiro caso baseada em *cliente/servidor* (C/S) e no segundo caso, baseada em *peer-to-peer* (P2P):

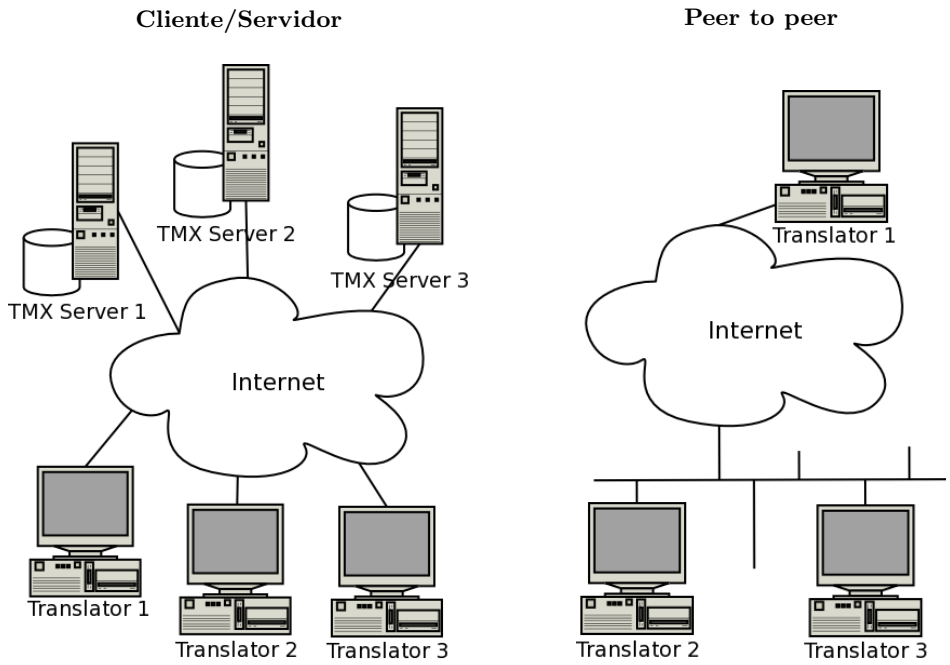

Figura 1. Arquitecturas de rede

- a arquitectura P2P é especialmente útil numa comunidade de tradução, onde vários tradutores estão a trabalhar e a partilhar automaticamente as traduções que estão a realizar;
  Pata este tipo de arquitectura podíamos dizer que uma tecnologia baseada em WebServices não é das mais adequadas já que obrigaria que cada tradutor tivesse um servidor web na sua máquina de trabalho. No entanto, não é completamente desprovido de senso a criação de um servidor dedicado apenas para a disponibilização de memórias de tradução. Neste artigo vamos supor que a arquitectura P2P é uma rede de várias máquinas, em que cada máquina é cliente e servidor ao mesmo tempo.
  Outras alternativas podiam ser baseadas em:
  - uso de uma tecnologia proprietária;
  - associação a um dos membros da rede P2P uma lista de clientes, que cada membro faz download e comunica usando, por exemplo, sockets;
  - criação de um servidor local para submissão de memórias de tradução e posterior divulgação (deixamos de ter arquitectura P2P real passando a C/S);
- Numa arquitectura C/S, será necessário um servidor dedicado para disponibilizar as unidades de tradução. Este será o método preferencial para a disponibilização de MT por terceiros.

Enquanto que na arquitectura P2P uma solução baseada em WebServices não é adequada, para uma arquitectura C/S o sistema pode ser implementado com uma qualquer tecnologia baseada em "remote procedure call", quer seja Sun RPC, Corba, Java RMI ou mesmo Web Services.

2.3 Definição do WebService

Esta secção pretende definir quais os métodos que o servidor de MT deve saber responder para a implementação de um serviço de MT distribuídas.

É usual especificar as mensagem inerentes ao processo usando a WebService Description Language (WSDL)[3]. No entanto pareceu-nos mais importante reflectir sobre a funcionalidade do WebService usando uma notação matemática do que a inclusão do documento WSDL que iria, sem dúvida, aumentar o número de páginas do artigo.

A definição deste conjunto de mensagens teve em conta o interesse de termos um servidor *stateless* — ou seja, o servidor não guarda qualquer informação sobre os vários clientes.

Num caso de subscrição de serviço em que seja necessária a autenticação do cliente, deixaremos de ter um servidor *stateless*, já que terá de armazenar informação sobre cada cliente. Também se teriam de alterar as mensagens aqui definidas, não só porque passaríamos a precisar de uma para o *login*, mas também porque teríamos de enviar em cada mensagem um *token* que identificasse o cliente. Visto que nos parece mais correcta a filosofia *open*, não nos iremos preocupar com a autenticação do serviço.

Configuração do cliente

Uma ferramenta de tradução que pretenda usar de um servidor de MT, terá de saber, para cada servidor, se o deve ou não usar, visto que as línguas disponíveis no servidor podem não ser as que o tradutor necessita.

Desta forma, o cliente deve, para cada servidor, construir uma lista de pares de MT disponíveis. Essa lista deverá ir sendo actualizada à medida que novas traduções surgem.

A configuração do cliente poderá ser feita com o envio de uma mensagem a que chamaremos `traduzes?` com um par de línguas, ao que o servidor irá responder com um valor booleano:

$$traduzes? : \mathcal{L}_\alpha \times \mathcal{L}_\beta \longrightarrow 2$$

Alem do tipo booleano que existe pre-definido, existe também um tipo chamado `Language`, sub-classe de `String` e que irá ser usado para representar línguas.

Pedido de tradução

O pedido de tradução que será dirigido a cada servidor de MT irá conter o par de línguas (língua de origem e língua destino) e uma frase na língua de

origem. O servidor irá responder com uma possível tradução e uma medida de qualidade[2] ou, nenhuma resposta.

$$traduz : \mathcal{L}_\alpha \times \mathcal{L}_\beta \times \mathcal{S}_{\mathcal{L}_\alpha} \longrightarrow \mathcal{S}_{\mathcal{L}_\beta} \times \mathcal{Q} + 1$$

Uma outra opção poderia ser a de permitir que cada servidor de MT respondesse com mais do que uma possível tradução. No entanto, parece-nos que cada servidor de MT deverá ter métodos próprios para a escolha da melhor tradução da sua base de traduções.

Concordância

Outra aplicação comum das MT a que se chama *concordância*, é o processo de, dada uma palavra ou sequência de palavras, encontrar todos os pares em que essa sequência aparece.

$$concordancia : \mathcal{L}_\alpha \times \mathcal{L}_\beta \times \mathcal{S}q_{\mathcal{L}_\alpha} \longrightarrow \left(\mathcal{S}_{\mathcal{L}_\alpha} \times \mathcal{S}_{\mathcal{L}_\beta}\right)^\star + 1$$

Embora semelhante ao pedido de tradução, a concordância é realizada com sequências muito mais pequenas de palavras e o resultado não passa pelo processo de selecção ou conciliação já que todos[3] os pares são apresentados ao utilizador.

Contribuição

Num sistema *peer-to-peer* local, todas as comunicações são efectuadas por uma rede local sem grande tráfego. Para a partilha por *peer-to-peer* numa comunidade grande e dispersa pela Internet, as comunicações podem limitar a interactividade do software de tradução.

Para colmatar este problema pode haver interesse na criação de servidores de MT distribuídos pela Internet, ligados em *peer-to-peer* em que as MT vão sendo partilhadas, e para onde cada tradutor irá contribuindo as traduções já efectuadas.

Surge a necessidade de uma nova mensagem no WebService para a contribuição de memórias de tradução para um servidor:

$$contribui : \mathcal{L}_\alpha \times \mathcal{L}_\beta \times \mathcal{S}_{\mathcal{L}_\alpha} \times \mathcal{S}_{\mathcal{L}_\beta} \longrightarrow 2$$

Outra solução seria a contribuição usando dois textos paralelos (em que um é a tradução do outro) em que o servidor teria a necessidade de o segmentar

[2] Parece-nos importante que cada tradução devolvida pelo servidor contemple uma medida de qualidade. Tradicionalmente, esta medida calcula-se como a distância de edição entre a frase armazenada na memória de tradução e a frase solicitada (distância de edição definida polo NIST como "The smallest number of insertions, deletions, and substitutions required to change one string or tree into another"(`http://www.nist.gov/dads/HTML/editdistance.html`). No entanto, o problema complica-se devido à influência da ordem das palavras e do formato do texto, entre outros factores[2]

[3] Por vezes o número de pares é demasiado grande, pelo que este serviço deve limitar o número de pares retornados.

e alinhar à frase antes de o tornar disponível. Embora mais demorado, este tipo de contribuição pode ser importante para o reaproveitamento de traduções realizadas sem o uso de ferramentas de tradução com suporte para MT.

Embora este processo de contribuição seja importante não só para a distribuição de carga mas também para equilibrar o modelo, não nos vamos debruçar sobre ela já que implicaria a definição de políticas de contribuição (contribuição totalmente aberta, com revisão, com autenticação, etc).

3 Implementação

Esta secção irá apresentar a implementação de um protótipo do conceito de MT distribuídas.

O cliente e servidor foram implementados usando Perl e SOAP (`SOAP::Lite`). Em relação a pormenores de implementação, parece-nos mais importante a parte do servidor do que a do cliente, já que esta última limita-se ao envio e recepção de mensagens, tendo somente que conciliar as respostas recebidas.

3.1 Cliente

A implementação de um cliente deveria ser feita como *plug-in* para um sistema de tradução. No entanto, quase todos os sistemas de tradução são comerciais o que não nos permite a sua alteração. Embora existam alguns *open-source* (Frankenstein[4], OmegaT[5], ForeignDesk[6]) optamos por criar apenas um protótipo de interface com o serviço de MT distribuídas, e mais tarde tentar contribuir com um destes projectos.

O código apresentado de seguida mostra o quão simples é a interface a um WebService usando Perl e `SOAP::Lite`. Na verdade, este código iria ser muito semelhante para qualquer outro tipo de sistema Cliente/Servidor.

```
use SOAP::Lite;

my $soapserver = SOAP::Lite
  -> uri('http://localhost:80/disttmx')
  -> proxy('http://localhost:80/cgi-bin/disttmx.cgi');

my $soapresult = $soapserver->traduz("pt","en","era uma vez...");

unless($soapresult->fault) {
  my $result = $soapresult->result();

  if ($result) {
    print "$result\n";
  } else {
```

[4] http://www.sourceforge.net/projects/frankenstein/
[5] http://www.sourceforge.net/projects/omegat/
[6] http://www.sourceforge.net/projects/foreigndesk/

```
    print STDERR "** server does not know how to translate it **\n";
  }
} else {
  print $soapresult->faultcode,": ",soapresult->faultstring;
}
```

3.2 Servidor

Embora o formato de intercâmbio de MT seja o TMX, sendo este um ficheiro de texto, estruturado mas sem limites físicos bem delimitados, torna-se dificil a implementação de uma pesquisa eficiente[7].

Uma solução passa pelo uso de algum tipo de indexador da informação. Com esse objectivo usamos o *glimpse*[5], um indexador de ficheiros com pesquisas bastante eficientes. No entanto, este indexador não sabe o que o XML é, pelo que não é fácil manusear directamente a memória de tradução.

Para resolver este problema cria-se um conjunto de ficheiros, m por cada língua, em que cada linha do ficheiro corresponde a uma memória de tradução (ou seja, a linha n do ficheiro m corresponde à linha n do ficheiro $\mathcal{T}(m)$).

O *glimpse* permite a pesquisa directa por um padrão ou por uma palavra, pelo que a implementação do servidor de memórias de tradução distribuída limita-se a receber o pedido, desempacotar o *query*, executá-lo usando o *glimpse*, procurar a tradução respectiva no ficheiro da língua de destino, empacotar a tradução e responder ao cliente.

Este processo não é tão eficiente quanto desejariamos já que cada execução do *glimpse* obriga-o a carregar todos os índices de consulta. Dado que a distribuição do *glimpse* incluí um servidor que carrega os índices apenas uma vez e responde a pedidos efectuados.

Usando este servidor temos uma arquitectura a dois níveis: um servidor de MT que recebe o pedido e o encaminha no formato correcto para o servidor *glimpse*, que enviará a resposta ao servidor de MT que a empacotará e a enviará ao cliente. Este processo pode ser visto na figura 2.

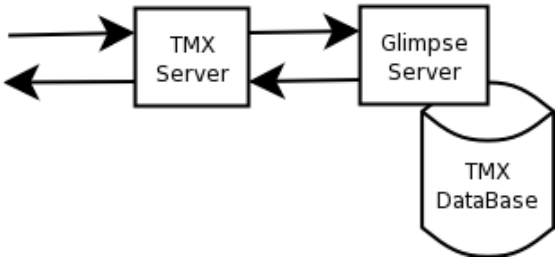

Figura 2. Arquitectura do Servidor usando o Glimpse

[7] Uma memória de tradução chega muito facilmente aos 100 Megabytes.

Embora esta implementação consiga ser eficiente, não permite o uso de algoritmos de *matching* mais complicados.

A implementação de um servidor de WebServices em Perl e `SOAP::Lite` é tão simples quanto a escrita do cliente. A CGI de tratamento de pedidos pode simplesmente invocar um módulo Perl dedicado à tarefa do servidor de forma completamente transparente:

```
use SOAP::Transport::HTTP;

SOAP::Transport::HTTP::CGI-> dispatch_to('disttmx')->handle;
```

Note-se que neste exemplo `disttmx` é o nome do módulo de gestão do servidor de MT distribuídas.

O módulo Perl que irá fazer a gestão do servidor de MT terá a seguinte estrutura:

```
package disttmx;
use strict;

sub traduzes {
    my ($self, $lang1, $lang2) = @_;
    ...}

sub traduz {
    my ($self, $lang1, $lang2, $frase) = @_;
    ...}
```

4 Conclusões

O conceito de MT distribuída pode ser importante de forma a criar utilização/construção cooperativa de recursos linguísticos. Embora não seja a nossa postura, as MT distribuídas também podem constituir uma oportunidade de negócio.

A implementação de MT distribuídas usando WebServices está ainda em fase de prototipagem, mas já demonstrou ser uma tecnologia viável. No entanto, sem a cooperação das empresas de ferramentas de tradução para a implementação desta filosofia, a construção de servidores de MT distribuídas não terá, por si só, grande repercussão no mundo da tradução.

Torna-se importante a construção de ferramentas eficientes para a indexação das MT que permitam a pesquisa por aproximação, número de palavras semelhantes e outros métodos de *matching*.

Referências

1. Joseba Abaitua. Memorias de traducción en TMX compartidas por internet. *Revista Tradumàtica*, (0), octubre 2001. `http://www.fti.uab.es/tradumatica/revista/num0/articles/jabaitua/imprim%ir.pdf`.

2. Yasuhiro Akiba, Kenji Imamura, and Eiichiro Sumita. Using multiple edit distances to automatically rank machine translation output. In *Machine Translation Summit VIII*, pages 15–25, 2001. http://www.eamt.org/summitVIII/papers/akiba.pdf.
3. Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web services description language (wsdl) 1.1, 2001. http://www.w3.org/TR/wsdl/.
4. Josu Gómez. Una guía al TMX. *Revista Tradumàtica*, (0), octubre 2001. http://www.fti.uab.es/tradumatica/revista/num0/articles/jgomez/imprimir%.pdf.
5. Udi Manber and Sun Wu. Glimpse: A tool to search through entire filesystems. Winter USENIX Technical Conference, 1994.
6. OSCAR. Open Standards for Container/Content Allowing Re-use — TMX home page, 2003. http://www.lisa.org/tmx/.
7. Yves Savourel. TMX 1.4a Specification. Technical report, Localisation Industry Standards Association, 1997.
8. *eXtended Markup Language (XML) version 1.0 recommendation*. World Wide Web Consortium, 10 February 1998. http://www.w3.org/TR/1998/REC-xml-19980210.html/.