基于短语的统计机器翻译系统
“丝路”1.0 版（SilkRoad V1.0）
设计与使用说明

中科院计算所 中科院自动化所 中科院软件所

厦门大学 哈尔滨工业大学

2006 年 10 月

引　言

“丝路”（SilkRoad）是一个基于短语的汉英统计机器翻译系统。该系统由中国的五家研究机构和高校联合开发（中科院计算所、中科院自动化所、中科院软件所、厦门大学、哈尔滨工业大学），并在 2006 年中国第二届统计机器翻译研讨会上发布，供国内外研究者共享使用。联合开发单位希望通过这个系统的发布，使更多的研究者能够更快、更容易地加入到统计机器翻译研究中来，推动国内统计机器翻译的迅速发展。

本文档给出了“丝路”1.0 版（SilkRoad V1.0）的设计与使用说明。

系统采用了目前主流的基于短语的统计机器翻译方法。在系统实现上充分利用了国际上目前已有的一些资源，包括一些开放源代码工具和一些可以公开获得授权的工具。在此基础上，联合开发单位分工协作，补充完成了翻译系统中尚不能公开获得的关键模块，包括语料库预处理、后处理模块，词语对齐后处理模块，短语抽取模块，解码器模块等。本文档主要介绍系统的整体设计以及这些新开发模块的实现原理和使用说明。

联合开发单位的具体分工如下：

中科院计算所：总体设计、语言模型接口设计和“骆驼 CAMEL”解码器；

中科院软件所：语料的预处理、后处理模块“仙人掌”；

中科院自动化所：词语对齐后处理模块“楼兰”和短语抽取模块“胡杨”；

厦门大学：　“商队 Caravan”解码器；

哈尔滨工业大学：　“绿洲 Oasis”解码器；

解码器是统计翻译系统的核心模块，有三家单位分别开发了自己的解码器模块。这三个解码器是相互独立的，用户可以选择使用其中任何一个解码器来完成翻译过程。

此外，该系统在实现时采用了“863 中文信息处理与智能人机接口评测”2005 年汉英机器翻译评测的数据集，包括训练集、开发集和测试集，这些数据可以通过 ChineseLDC 获得研究目的授权。

目录

1 “丝路”系统设计概述

本章主要介绍“丝路”汉英统计机器翻译系统的总体设计。

1.1 基于短语的统计机器翻译模型

基于短语的统计机器翻译（Koehn et al.,2003；Zens et al., 2002；Koehn, 2004）以短语作为翻译的基本单位。对于一个汉语句子，翻译系统将其划分为多个连续的词语串（即所谓的“短语”），然后将每一个汉语短语翻译为英语短语，最后将产生的英语短语进行顺序调整，并输出译文，请参考图 1 的例子。

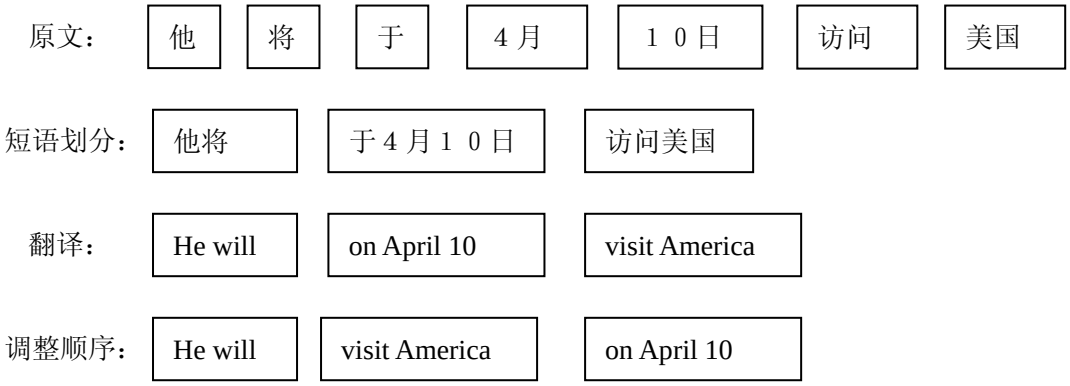

图 1：基于短语的统计机器翻译过程示例

1.2 系统流程

“丝路”系统包含以下 4 个主要部分：短语翻译模型的训练、语言模型的训练、解码、翻译结果的评价。下面以流程图的形式分别介绍这 4 部分。

1.2.1 短语翻译模型的训练

通过短语翻译模型的训练，从汉英句子对齐的语料库中学习到汉语短语到英语短语的翻译概率表，流程图参见图 2。

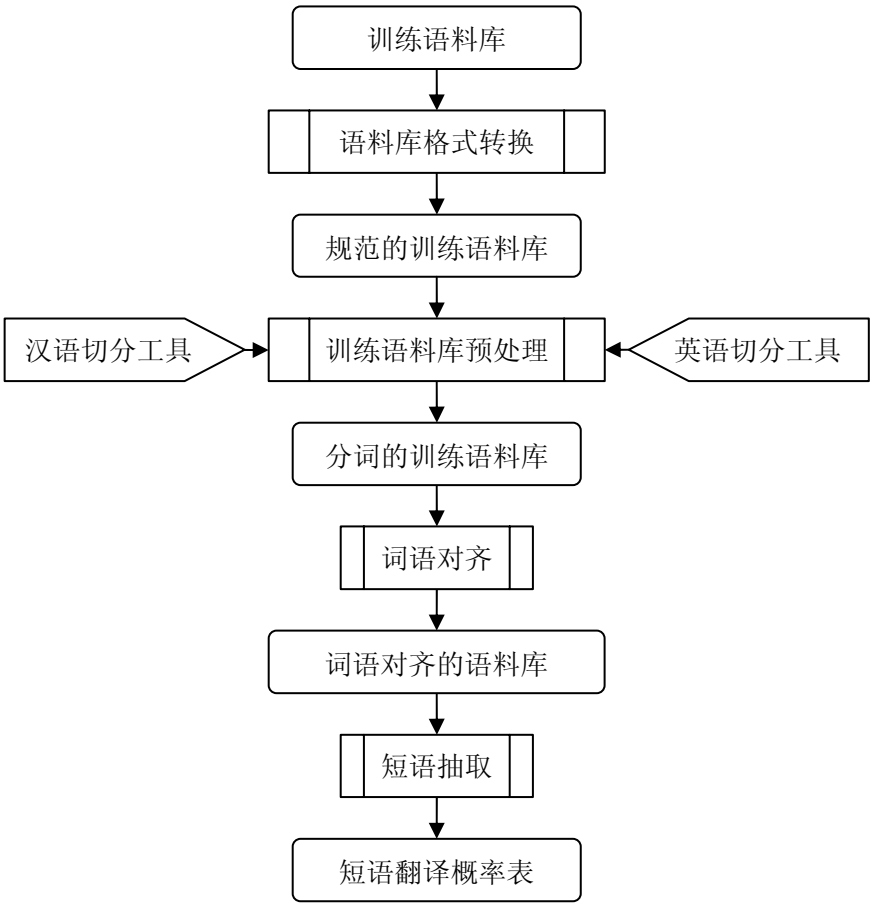

图 2：短语翻译模型训练的流程

1.2.2 语言模型的训练

在英语单语语料库上训练得到英语的语言模型，流程图参见图 3。

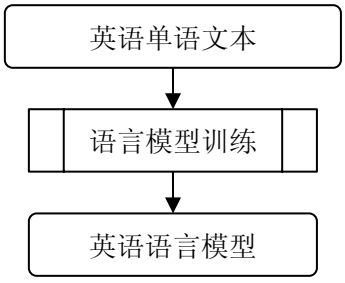

图 3：语言模型训练的流程

1.2.3 解码

解码器是系统的核心模块，通过解码，将一个输入的汉语句子翻译成英语句子，流程图参见图 4。

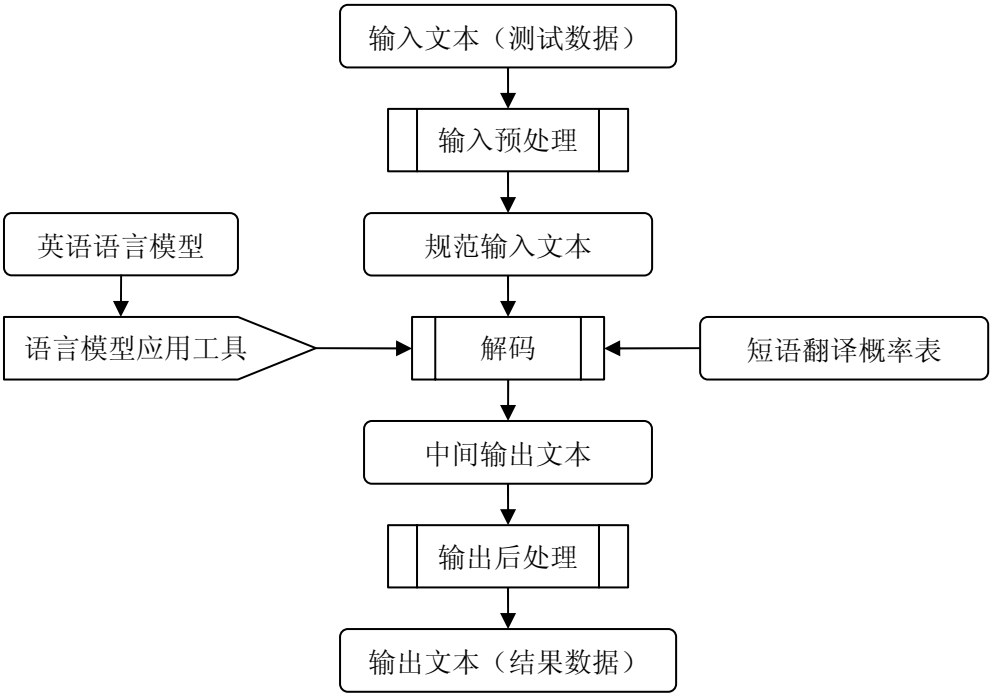

图 4：解码的流程

1.2.4 翻译结果的评价

我们采用评价工具对系统的输出结果进行自动评价，流程图参见图 5。

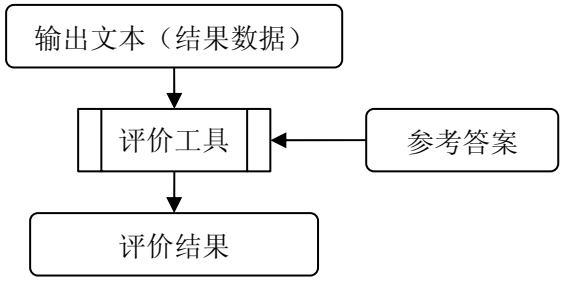

图 5：评价的流程

1.3 模块划分

整个系统由以下模块构成：

1) 训练语料库预处理模块；
2) 汉语分词工具（利用开源工具 ICTCLAS）；
3) 英语分词工具（将开源工具 tokenizeE.perl.tmpl 改写为 C++代码）；
4) 词语对齐模块（利用开源工具 GIZA++获得初始对齐，对其结果进行后处理）；
5) 短语抽取模块；
6) 语言模型训练模块（利用开源工具 SRI）；
7) 语言模型应用工具（利用开源工具 SRI）；
8) 输入预处理模块；
9) 解码模块；

10) 输出后处理模块；
11) 评价模块（利用 863 评测工具）；

每个模块都是一个可独立执行的文件，可以运行在 Windows 或者 Linux 等不同的平台上；模块之间以文件作为接口，这些文件包括：

1) 训练语料库；
2) 规范的训练语料库（采用 GIZA++定义）；
3) 分词的训练语料库（采用 GIZA++定义）；
4) 词语对齐的语料库；
5) 短语翻译概率表；
6) 语言模型（采用 SRI 定义）；
7) 输入文本（采用 863 评测定义）；
8) 输出文本（采用 863 评测定义）；
9) 参考答案（采用 863 评测定义）；
10) 评价结果（采用 863 评测定义）；

1.4 已有资源和工具简介

从上面的介绍可以看到，这个系统相当一部分的工作都可以利用已有的资源，这里对这些资源作一个简单介绍。

1.4.1 汉语分词工具 ICTCLAS

汉语词法分析系统 ICTCLAS 是由中国科学院计算技术研究所自然语言处理课题组开发的一套开放源代码的汉语分词和词性标注工具，可以在“中文自然语言处理开放平台”上下载，该平台网址为：http://www.nlp.org.cn。注意下载之前要使用真实信息注册一个用户名。

1.4.2 词语对齐模块 GIZA++

GIZA 是由 Och 等人在 1999 年 JHU 夏季讨论班上开发的一个开放源代码的统计机器翻译系统 Egypt 的一部分，是一个 IBM 统计翻译模型（Brown et al., 1993）的训练工具。后来，Och 对它进行了改进，这就是 GIZA++。

GIZA++的主页是：http://www.fjoch.com/GIZA++.html。

GIZA++原来是在 linux 上运行的，有个中国人（网名 Blue Gene）把 GIZA++移植到了 Windows 的 Visual Studio C++ 7.0 平台，下载地址是：

http://blogs.gcomputing.com/bluegene/archives/files/GIZA++.Win32.BlueGene.rar

关于 GIZA++的使用，中国科学院计算技术研究所的博士生刘洋写了一份详尽的报告：“利用现有软件构建统计机器翻译系统”：

http://mtgroup.ict.ac.cn/~liuyang/papers/construct_smts.pdf

这份报告介绍了如何搭建一个基于 IBM 模型的统计机器翻译系统，主要采用的工具就是 GIZA++。当然我们现在要做的是一个基于短语的系统，比 IBM 模型有了很大的改进。

1.4.3 英语分词工具 tokenizeE.perl.tmpl

tokenizeE.perl.tmpl 是一个简单的英语分词工具，和 GIZA 一样，都是 Egypt 系统的一部分。完整的 Egypt 可以在以下网页下载：

http://www.clsp.jhu.edu/ws99/projects/mt/toolkit/

里面包含了这个英语分词工具。

我们将这个工具改写为 C++代码，由语料库预处理模块直接调用。

1.4.4 语言模型工具 SRI

SRILM 是 SRI 口语技术与研究实验室（SRI Speech Technology and Research Laboratory）发布的开源的语言模型工具包。这个工具包包含一组 C++类库、一组进行语言模型训练和应用的可执行程序、其它相关工具。

SRILM 的主页是：http://www.speech.sri.com/projects/srilm/

SRILM 的最新版本是 1.5.0 版。下载地址是：

http://www.speech.sri.com/projects/srilm/download.html

考虑到可移植性，训练出的语言模型数据采用 ARPA 标准格式。细节请参考：（Jurafsky & Martin,2000），（Manning & Schütze,1999），（Stolcke,2002）。

1.4.5 863 评测资源

“丝路”系统采用 2005 年 863 中文信息处理与智能人机接口中的机器翻译评测资源，具体包括：

评测大纲：包括输入输出数据格式定义；

评测数据：包括训练集、开发集和测试集；

评测工具；

以上数据可以通过 ChineseLDC 授权购买。ChineseLDC 的网址是：

http://www.chineseldc.org

评测大纲可以直接从 863 评测网站上下载，网址是：

http://www.863data.org.cn

1.5 数据格式定义

本节对第 1.3 节中提出的文件格式进行统一定义和说明：

1.5.1 训练语料库格式定义

本次讨论班采用 2005 年 863 中文信息处理与智能人机接口中机器翻译评测的训练语料库。这些语料库没有统一的格式，需要分别处理成规范的训练语料库。

1.5.2 规范的训练语料库格式定义

包括两个文件，分别存放中文语料及英文语料。文件的一行为一个独立的句子，中文文件及英文文件的相同行为对应句子。

例如：

中文文件

中国化工工业保持稳定增长。 万里会见泰国客人

表 1：规范的训练语料库中文文件示例

英文文件

China’s chemical industry maintains steady growth. Li Wan meets with guests from Thailand

表 2：规范的训练语料库英文文件示例

1.5.3 分词的训练语料库格式定义

GIZA++的标准输入文件格式，包括两个文件，分别存放分词以后的源语言语料及目标语言语料。

例如：

中文文件

中国 化工 工业 保持 稳定 增长 。 万里 会见 泰国 客人

表 3：分词的训练语料库中文文件示例

英文文件

China ’s chemical industry maintains steady growth . Li Wan meets with guests from Thailand

表 4：分词的训练语料库英文文件示例

1.5.4 词语对齐的语料库格式定义

采用 xml 格式。文件包含一个< parallel_corpus>元素。< parallel_corpus >元素由若干个<bead>元素（由<bead …>和</bead>括起来的部分）组成，其中<bead>元素的属性 id 的值是正整数。每个<bead>元素的 id 各不相同，但不一定是连续的数值。每个<bead>元素包含源语言句子、目标语言句子以及对齐信息，其中源语言句子由<srctext>标记括起，目标语言句子由<tgttext>标记括起，对齐信息由<wordalignment>元素括起，为若干词的序号对，格式为“源词序号:目标词序号”。各个对齐对之间用空格分隔。第一个词的序号为 1。

定义：

```
<?xml version="1.0" encoding=" gbk"?>
<parallel_corpus>
<bead id="1">
<srctext>源语言句子</srctext>
<tgttext>目标语言句子</tgttext>
<wordalignment>源词序号:目标词序号 源词序号:目标词序号…</wordalignment>
</bead>
……
</parallel_corpus>
```

表 5：词语对齐的语料库格式定义

例如：

```
<?xml version="1.0" encoding="gbk"?>
<parallel_corpus>
<bead id="1">
<srctext> 中国 化工 工业 保持 稳定 增长 </srctext>
<tgttext> China ’s chemical industry maintains steady  growth </tgttext>
<wordalignment> 1:1 1:2 2:3 3:4 4:5 5:6 6:7 </wordalignment>
</bead>
<bead id="2">
<srctext> 世界 游泳 锦标赛 </srctext>
<tgttext> world Swimming Championship </tgttext>
<wordalignment> 1:1 2:2 3:3 </wordalignment>
</bead>
</parallel_corpus>
```

表 6：词语对齐的语料库示例

1.5.5 短语翻译概率表格式定义

短语抽取模块的输出文件，每行包含 3 部分：汉语短语，英语短语，翻译概率值。由连续的“|||”分开：

定义：

汉语短语 $\tilde{c}$ ||| 英语短语 $\tilde{e}$ ||| $p(\tilde{c}|\tilde{e})$ $lex(\tilde{c}|\tilde{e})$ $p(\tilde{e}|\tilde{c})$ $lex(\tilde{e}|\tilde{c})$

其中：

$p(\tilde{c}\,|\,\tilde{e})$ 表示 $\tilde{e}$ 翻译为 $\tilde{c}$ 的概率， $p(\tilde{c}\,|\,\tilde{e}) = \dfrac{N(\tilde{c},\tilde{e})}{\sum_{\tilde{c}'} N(\tilde{c}',\tilde{e})}$ （公式 1）

$lex(\tilde{c}\,|\,\tilde{e})$ 表示词汇化的翻译概率，

$$lex(c_1^J\,|\,e_1^I, a) = \prod_{j=1}^{J} \frac{1}{|\{i\,|\,(j,i)\in a\}|} \sum_{\forall(j,i)\in a} p(c_j\,|\,e_i) \qquad \text{（公式 2）}$$

$p(\tilde{e}\,|\,\tilde{c}), lex(\tilde{e}\,|\,\tilde{c})$ 表示另外一个方向的翻译概率。

例如：

银行 和 保险 公司 ||| banks and insurance companies ||| 1 0.105599 1 0.0257825
坚持 改革 开放 ||| of reform and opening up ||| 0.333333 0.00103337 1 0.014045

表 7：短语翻译概率表示例

1.5.6 语言模型格式定义

SRILM 工具包的输出结果文件。采用 ARPA 标准格式，主要包括三部分内容：

1. 开始和结束标记："*\data*""*\end*"
2. 长度从 1 到 N 的 N 元语法项的数目
3. 长度从 1 到 N 的 N 元语法项的具体信息，包括该项的条件概率的对数值（以 10 为底），组成该项的相应长度个单词及回退权重（backoff weight）的对数值（以 10 为底）。最后一项为可选。

语言模型文件中可能包括三个特定的词：<s>，</s>和<unk>，它们分别代表"句子的开始"、"结束标记"和"未登录词"。

例如：

```
\data\
ngram 1=23
ngram 2=57
ngram 3=15

\1-grams:
-0.6981126      </s>
-99   <s>  -1.226533
-1.357798       I       -0.7723913
-1.155147 a     -0.9225594
-1.45373  am    -0.6464851
-1.357798       are   -1.048023
-2.041393       beijing     -0.6230347
-1.45373  boy   -0.6018454
…
\2-grams:
-0.90309  <s> I       -0.2552725
-0.90309  <s> he      -0.30103
-0.90309  <s> she     -0.30103
…
\3-grams:
-0.1760913      <s> I am
-0.1760913      not a boy
-0.1760913      I am a
-0.1760913      where are you
…
\end\
```

表 8：语言模型文件示例

1.5.7 输入文本格式定义

输入文本即为测试数据文件，该文件采用 xml 格式。每个文件包含一个<srcset>元素，这个元素包含 setid、srclang、tgtlang 属性，分别说明测试集的 id、源语言和目标语言代码。<srcset>包含若干<doc>元素（由<doc …>和</doc>括起来的部分），其中<doc>元素的属性说明文档相关信息。docid 给出文档名称，属性值用双引号引起。语言代码中，英语用“en”表示，汉语用“zh”表示。

每个<doc>元素由若干个 元素（由 和 括起来的部分）组成。每个 元素由若干个<s>元素（由<s …>和</s>括起来的部分）组成，其中<s>元素的属性 id 的值是正整数。每个<s>元素的 id 各不相同，但不一定是连续的数值。每个<s>元素可能包含一个或多个句子。也可能没有 元素，而<s>元素直接包含在<doc>元素中。

定义：

```
<?xml version="1.0" encoding="UTF-16"?>
< srcset setid="测试集 id" srclang="源语言代码" tgtlang="目标语言代码">
<doc docid="文档名称">
<p>
<s id="1"> 源语言句子 </s>
</p>
…
<p>
<s id="n"> 源语言句子 </s>
</p>
</doc>
</ srcset >
```

表 9：输入文本格式定义

例如：

```
<?xml version="1.0" encoding="gbk"?>
<srcset setid="2004_zh_en_dial" srclang="zh" tgtlang="en">
<doc docid="d1">
<p>
<s id="1"> 上海浦东开发与法制建设同步 </s>
</p>
<p>
<s id="2"> 新华社上海二月十日电（记者谢金虎、张持坚）</s>
</p>
<p>
<s id="3"> 外商投资企业成为中国外贸重要增长点 </s>
</p>
</doc>
</srcset>
```

表 10：输入文本示例

1.5.8 输出文本格式定义

输出文本即机器翻译产生的结果文件，该文件也采用 xml 格式。结果文件的格式与测试文件相同，其中 doc 中要增加一个 site 属性，给出翻译单位名称。结果文件中，<doc>元素、<p>元素、<s>元素及其结构关系应与测试文件一一对应，但是<p>元素保留或不保留皆可。对应的<doc>元素的 docid 属性和<s>元素的 id 属性应与测试文件相同。但是<srcset>元素应改为对应的<tstset>元素，各属性不变。<s>元素中的内容是对应测试文件中的<s>元素的翻译结果。

定义：

```
<?xml version="1.0" encoding="UTF-16"?>
<tstset setid="测试集 id" srclang="源语言代码" tgtlang="目标语言代码">
<doc docid="文档名称" site="单位名称">
<p>
<s id="1"> 目标语言句子 </s>
</p>
…
<p>
<s id="n"> 目标语言句子 </s>
</p>
</doc>
</tstset>
```

表 11：输出文本格式定义

例如：

```
<?xml version="1.0" encoding="UTF-16"?>
<tstset setid="2004_zh_en_dial" srclang="zh" tgtlang="en">
<doc docid="d1" site="ICT">
<p>
<s id="1">The development of Shanghai 's Pudong is in step with the establishment of its legal system </s>
</p>
<p>
<s id="2"> Xinhua News Agency , Shanghai , February 10 , by wire ( reporters Jinhu Xie and Chijian Zhang ) </s>
</p>
<p>
<s id="3"> Foreign-invested enterprises have become a major focus of growth in China 's foreign trade</s>
</p>
</doc>
</tstset>
```

表 12：输出文本示例

1.5.9 参考答案格式定义

参考答案文件采用 xml 格式，格式与结果文件类似，区别主要有两点。一是<doc>元素中的 site 属性表示翻译者，如果有 4 个翻译者，则参考答案中同一 docid 的<doc>元素会出现 4 次。二是<tstset>元素应改为对应的<refset>元素。

定义：

```
<?xml version="1.0" encoding="UTF-16"?>
<refset setid="测试集 id" srclang="源语言代码" tgtlang="目标语言代码">
<doc docid="文档名称 1" site="翻译者 1">
<s id="1"> 参考答案句子 </s>
…
<s id="n"> 参考答案句子 </s>
</doc>
…
<doc docid="文档名称 1" site="翻译者 n">
<s id="1"> 参考答案句子 </s>
…
<s id="n"> 参考答案句子 </s>
</doc>
…
<doc docid="文档名称 N" site="翻译者 n">
<s id="1"> 参考答案句子 </s>
…
<s id="n"> 参考答案句子 </s>
</doc>
</refset>
```

表 13：参考答案格式定义

例如：

```
<?xml version="1.0" encoding="UTF-16"?>
<refset setid="2005_zh_en_dial" srclang="zh" tgtlang="en">
<doc docid="zh_dial" site="alun">
<s id="1">Welcome to China.</s>
<s id="2">We'll go to the hotel now, the car's outside.</s>
<s id="3">We will send your luggage up to your room.</s>
…
</doc>
<doc docid="zh_dial" site="chew">
<s id="1">Welcome to China.</s>
<s id="2">We are going to the hotel now, the car is waiting outside.</s>
<s id="3">Your luggage will be sent to your rooms.</s>
…
</doc>
</refset>
```

表 14：参考答案示例

1.5.10 评价结果格式定义

使用自动评测程序比较结果文件和参考答案生成的评价文件，含有多种自动评价标准，包括： BLEU（N=4）、NIST（N=5）、GTM、mWER、mPER 等。自动评测的算法是大小写敏感的。

例如：

```
MT evaluation scorer began on 2006 Sep 4 at 16:48:02
  Evaluation of zh-to-en translation using:
    src set (1 docs, 467 segs)
    ref set (4 refs)
    tst set (1 systems)

NIST score =  6.6551 BLEU score = 0.2169 GTM score = 0.6474 mWER score = 0.6631 mPER score = 0.5309 ICT score = 0.4089 for system "CAMEL"

# ------------------------------------------------------------------------
Individual N-gram scoring
        1-gram   2-gram   3-gram   4-gram   5-gram   6-gram   7-gram   8-gram   9-gram
        ------   ------   ------   ------   ------   ------   ------   ------   ------
 NIST:  5.2769   1.1282   0.1984   0.0403   0.0113   0.0035   0.0009   0.0000   0.0000 ""

 BLEU:  0.6710   0.3020   0.1461   0.0747   0.0406   0.0218   0.0108   0.0042   0.0015 ""


# ------------------------------------------------------------------------
Cumulative N-gram scoring
        1-gram   2-gram   3-gram   4-gram   5-gram   6-gram   7-gram   8-gram   9-gram
        ------   ------   ------   ------   ------   ------   ------   ------   ------
 NIST:  5.2769   6.4052   6.6035   6.6438   6.6551   6.6586   6.6594   6.6594   6.6594 ""

 BLEU:  0.6710   0.4502   0.3094   0.2169   0.1552   0.1119   0.0801   0.0554   0.0370 ""

MT evaluation scorer ended on 2006 Sep 4 at 16:48:07
```

表 15：评价结果示例

2 训练模块设计与使用

2.1 训练语料预处理

负责人：张大鲲 dakun04@iscas.cn

本小节介绍对训练语料的预处理。一般的，训练语料都是汉英句子对齐的语料，在进行GIZA++训练之前，需要对这些语料进行加工处理。

2.1.1 模块定义

1) 输入文件：

规范的训练语料库

2) 输出文件：

处理后的训练语料库

3) 需要调用的库函数：

汉语分词工具 ICTCLAS

英语分词工具，将 tokenizeE.perl.tmpl 改写成 C++

4) 功能说明：

对训练数据进行预处理，完成以下工作：

- 中文分词；
- 英文分词；
- 英文句子首字母大写还原。
- 将中文中的 A3 区全角字符替换为半角字符

2.1.2 实现原理

1) 中文分词

中文分词模块利用开源工具 ICTCLAS（参见 1.4.1），可以对多个文档进行处理，分词后的文件保持原文件名（filename.txt），同时产生原文件的备份文件（filename.cla.bak）。

例： 中文分词模块调用开源工具 ICTCLAS

处理后： 中文 分词 模块 调用 开源 工具 ICTCLAS

2) 中文 A3 区全角字符转换

将中文的 A3 区全角符号 A - Z， a - z， 0 - 9，共 62 个字符，转换为相应的半角符号（英文 ASCII 码）A-Z，a-z，0-9，系统采用查表的方式实现。

例： 经常有人把年份写成２００６年

处理后： 经常有人把年份写成 2006 年

3) 英文分词

英文分词部分，实现了 1999 年在 JHU 夏季机器翻译讨论班上发布的英文分词工具 tokenizeE.perl.tmpl（参见 1.4.3）的 C++代码转写，输出结果和 perl 程序保持一致。主要处理英文的标点，数字，缩写形式和缩略语等。

例： Mr. and Mrs. shouldn't be separated.

处理后： Mr. and Mrs. shouldn ' t be separated .

4) 英文句子首字母大写还原

统计英文句首词出现的次数以及句首词首字母小写形式在文中出现的次数，如果句首词首字母小写形式出现的次数多，则将句首词转换为首字母小写形式，否则保持不变

2.1.3 使用说明

1) A3 区全角字符转换程序

SBC2DBC.exe File1.txt File2.txt …

2) 中文分词程序

SPLIT.exe -操作类型参数 -输出格式参数 File1.txt File2.txt ...

默认参数为 0 0，可以省略，即

SPLIT.exe -0 -0 Chinese.txt == SPLIT.exe Chinese.txt

参数说明参见表 16：

操作类型参数	对应的分词操作	输出格式参数	对应的输出格式
0	词语切分	0	北大标准
1	一级标注	1	973 标准
2	二级标注	2	XML

表 16：中文分词参数说明

3) 英语分词程序

EnglishToken.exe File1.txt File2.txt …

4) 英文句子首字母大写转换程序

UpperLowerTrans.exe File1.txt File2.txt …

生成中间文件 ult_record.txt，保存经过转换的句首词

以上每个程序都可以同时处理多个文件，处理后的结果仍然使用源文件名命名，同时将源文件生成备份文件（.bak）。

2.2 词语对齐“楼兰”

负责人：魏玮 weiwei@hitic.ia.ac.cn

本小节介绍词语对齐模块，对训练语料库中的句子自动进行词语对齐。

2.2.1 模块定义

1) 输入文件：

GIZA++双向训练的对齐结果文件：从汉语到英语方向，从英语到汉语方向

2) 输出文件；

词语对齐的语料库

3) 功能说明：

对 GIZA++的对齐结果进行优化

2.2.2 实现原理

其基本思想是利用 GIZA++进行汉语到英语、英语到汉语两个方向的训练，再对两个方向的对齐结果按照 Och 等提出的 Heuristic (Och and Ney, 2000)的思路进行优化。

Giza++实现了 IBM 统计翻译模型，但得到的对齐结果忽略了多对多及多对一的情况。为了解决 Giza++词语对齐的问题，通常利用双向对齐的结果进行优化。优化的方法采用了以两个方向对齐结果的交集为中心点，检查其上下左右(grow)及对角(diag)相邻的 8 个点，若在并集中，则作为扩展的对齐点加入对齐序列中。

具体实现见以下伪代码和图示例句。

```
Neighboring =((-1,0),(0,-1),(1,0),(0,1),(-1,-1),(-1,1),(1,-1),(1,1));
Align_Intersect=intersect(e2f,f2e);
Align_Union=union(e2f,f2e);
Heuristic();

Heuristic():
  for english word e=0…..en
      for chinese word f=0…..fn
         if ( e aligned with f)
             for each neighboring point ( e-new,f-new)
                 if ( ( e-new not aligned and f-new not aligned )
                    and (e-new,f- new) in Align_Union )
                           add alignment point ( e-new,f-new );
```

算法 1：词语对齐优化算法

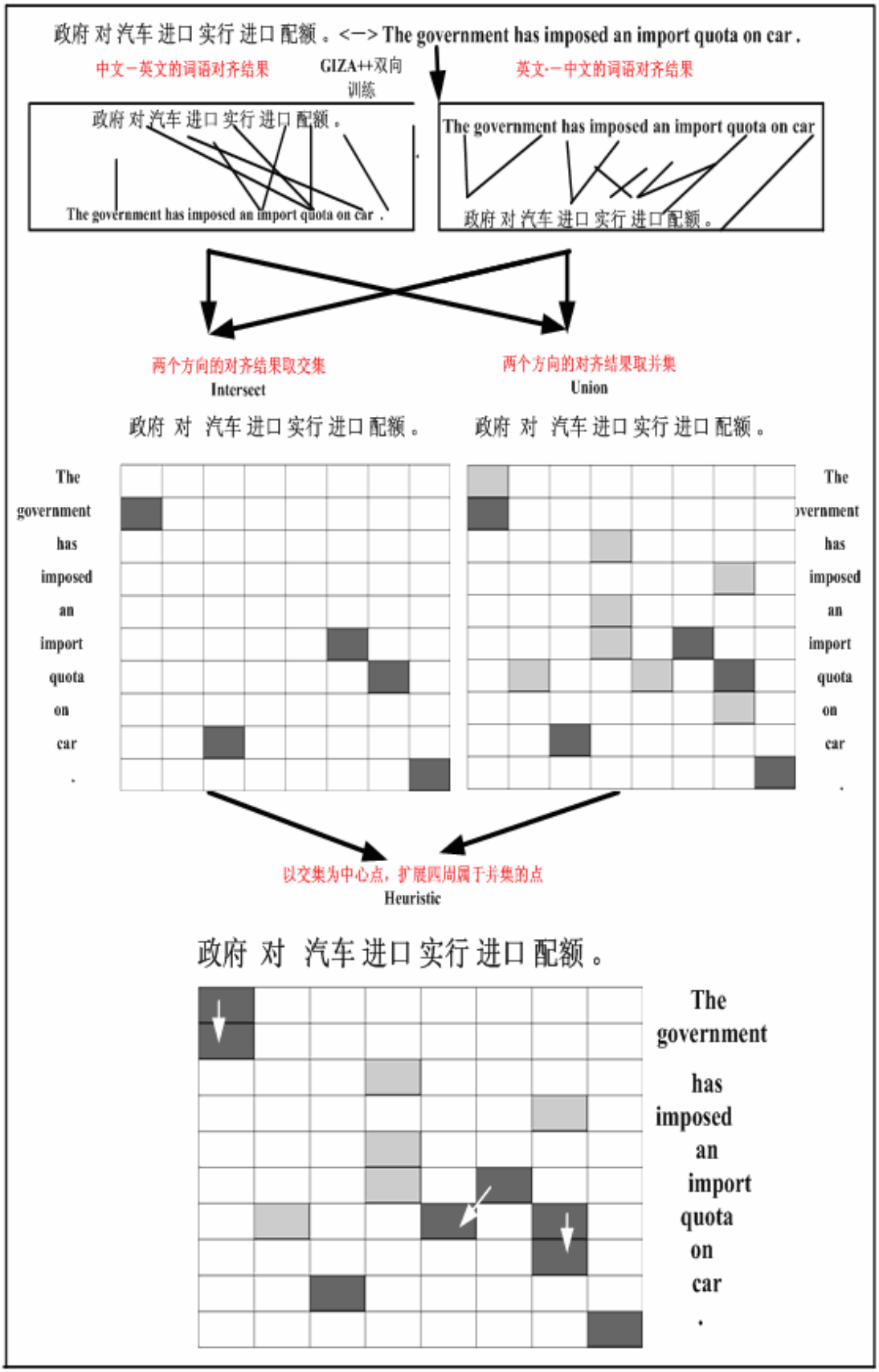

图 6：生成 Heuristic 对齐结果的具体过程

2.2.3 使用说明

1) 输入：

- 文件：Giza++双向训练的对齐结果文件：从汉语到英语方向，从英语到汉语方向(默认文件为当前目录下: f2e_giza_alignment.txt, e2f_giza_alignment.txt)
- 参数选择：在程序运行初始，可以选择输出 5 种不同的优化结果作为参数：
  a) F2E(汉语到英语的对齐结果）
  b) E2F (英语到汉语的对齐结果)
  c) Union(两个方向取并集)
  d) Intersect(两个方向取交集)
  e) Heuristic (利用前面提到的 Och 的 Refined alignment 的思路，将交集点向其周围的“邻居”进行扩展(Och,2002; Koehn et al., 2003)

2) 输出：

按照词语对齐的语料库的格式定义，以 xml 文件输出。(默认文件为当前目录下：****_Alignment.xml)

3) 注意事项：

- 在进入 Giza++训练之前，已将中文或英文大于 100 个词的句对过滤。所以在该模块默认的最大句长是 100(全局变量 MAX).
- 在进行参数选择时，要按照提示中指定格式进行输入，要包含前面的代码和具体类别。如：5)Heuristic。
- 在产生对齐结果的过程中，每处理 1000 句，屏幕显示一个‘．’。
- 具体细节详见代码中注释。

2.3 短语抽取“胡杨”

负责人：何彦青 yqhe@nlpr.ia.ac.cn

本小节介绍短语抽取模块，从词对齐的语料库中自动学习汉语短语到英语短语的翻译。

2.3.1 模块定义

1) 输入文件：

词语对齐的语料库

2) 输出文件：

短语翻译概率表

3) 功能说明：

从词语对齐的语料库中抽取双语短语，并计算翻译概率 $p(\tilde{c}\,|\,\tilde{e})$ $lex(\tilde{c}\,|\,\tilde{e})$ $p(\tilde{e}\,|\,\tilde{c})$ $lex(\tilde{e}\,|\,\tilde{c})$

2.3.2 实现原理

许多基于短语的统计机器翻译系统的短语抽取方法都很相似，根据这些方法，本模块实现了从词对齐中自动抽取双语短语的算法。基本思想就是首先根据词对齐生成最大似然词汇化词典，然后进行短语抽取，最后对每一个短语对计算 4 个翻译概率。

2.3.2.1 生成最大似然词汇化翻译表

对于已经进行了词对齐的语料，直接估计最大似然词汇化翻译表，用于短语对的词汇化概率计算。如果直接用 GIZA++产生的翻译表计算，由于有的词条在 GIZA++词典中没有出现，导致会有一些短语对的词汇化概率为 0，而且 GIZA++词典中没有 w(c|NULL)和 w(e|NULL)这两个概率值。直接估计的方法就是直接抽取对齐的词对，如果某个词没有与之对齐的翻译词，就认为它与 NULL 对齐，然后计算同现次数，按照下面公式分别计算出 w(e|f)和 w(f|e)，从而直接生成词汇化翻译表。

$$w(f\,|\,e)=\frac{count(e,f)}{count_f(e,f)} \qquad \text{（公式 3）}$$

$$w(e\,|\,f)=\frac{count(e,f)}{count_e(e,f)} \qquad \text{（公式 4）}$$

2.3.2.2 短语抽取

从词语对齐的语料库中抽取双语短语，要求短语对必须与词对齐相容。定义如下：

$$(\bar{e},\bar{f})\in BP \Leftrightarrow \quad \forall e_i\in\bar{e}:(e_i,f_j)\in A\rightarrow f_j\in\bar{f}$$

$$AND\quad \forall f_j\in\bar{f}:(e_i,f_j)\in A\rightarrow e_i\in\bar{e} \qquad \text{（公式 5）}$$

其中，A 表示词语对齐的矩阵。

抽取方法就是提取对齐矩阵中的所有以对齐点为顶点的矩形，条件是与矩形所在行范围内的源词对齐的目标词也都在这个矩形的列范围内，反之亦然。例如：

	中国	化学	工业	保持	稳定	增长
China	■					
's	■					
chemical		■				
industry			■			
maintains				■		
steady					■	
growth						■

图 7：短语抽取示例

对于上例，抽取出的短语结果为：

```
中国 ||| China ’s ||| 0-0 0-1
中国 化工 ||| China ’s chemical ||| 0-0 0-1 1-2
中国 化工 工业 ||| China ’s chemical industry ||| 0-0 0-1 1-2 2-3
中国 化工 工业 保持 ||| China ’s chemical industry maintains ||| 0-0 0-1 1-2 2-3 3-4
中国 化工 工业 保持 稳定 ||| China ’s chemical industry maintains steady ||| 0-0 0-1 1-2 2-3 3-4 4-5
中国 化工 工业 保持 稳定 增长 ||| China ’s chemical industry maintains steady growth ||| 0-0 0-1 1-2 2-3 3-4 4-5 5-6
化工 ||| chemical ||| 0-0
化工 工业 ||| chemical industry ||| 0-0 1-1
化工 工业 保持 ||| chemical industry maintains ||| 0-0 1-1 2-2
化工 工业 保持 稳定 ||| chemical industry maintains steady ||| 0-0 1-1 2-2 3-3
化工 工业 保持 稳定 增长 ||| chemical industry maintains steady growth ||| 0-0 1-1 2-2 3-3 4-4
工业 ||| industry ||| 0-0
工业 保持 ||| industry maintains ||| 0-0 1-1
工业 保持 稳定 ||| industry maintains steady ||| 0-0 1-1 2-2
工业 保持 稳定 增长 ||| industry maintains steady growth ||| 0-0 1-1 2-2 3-3
保持 ||| maintains ||| 0-0
保持 稳定 ||| maintains steady ||| 0-0 1-1
保持 稳定 增长 ||| maintains steady growth ||| 0-0 1-1 2-2
稳定 ||| steady ||| 0-0
稳定 增长 ||| steady growth ||| 0-0 1-1
增长 ||| growth ||| 0-0
```

表 17：短语抽取结果

这个结果中每一行格式如下：

中文短语 ||| 英文短语 ||| 词语对齐（中-英）

2.3.2.3 计算概率

短语抽取完后，再计算四个翻译概率，$p(\tilde{c}|\tilde{e})$，$lex(\tilde{c}|\tilde{e})$，$p(\tilde{e}|\tilde{c})$，$lex(\tilde{e}|\tilde{c})$。计算方法就是对所有生成的短语对分别计算英文短语、中文短语以及二者的同现次数，然后求商得到 $p(\tilde{c}|\tilde{e})$ 和 $p(\tilde{e}|\tilde{c})$，然后利用 2.3.2.1 中生成的词典按照

$$lex(c_1^J|e_1^I,a)=\prod_{j=1}^{J}\frac{1}{|\{i|(j,i)\in a|_{\forall(j,i)\in a}}\sum p(c_j|e_i)$$

（公式 2）计算 $lex(\tilde{c}|\tilde{e})$ 和 $lex(\tilde{e}|\tilde{c})$。

例如：

	中国	化学
China	■	
's	■	
chemical		■

$$lex(\text{中国 化工}|\text{China 's chemical})=\frac{1}{2}(w(\text{中国}|\text{China})+w(\text{中国}|\text{'s}))\times w(\text{化工}|\text{chemical})$$

图 8：短语词汇化翻译概率的计算

2.3.3 使用说明

按照 2.3.2 介绍的原理程序也相应分为三部分：生成词典，抽取短语和计算概率。该模块基于 vc 6.0 、vc.net 以及 linux 平台编写，分别放在 dict，pharaoh 和 com_prob 三个文件夹下。

2.3.3.1 程序 1：生成词典

1) 主程序为 dictionary.cpp，如要修改文件名或路径，请在 definev.h 中修改。
2) 请先在当前目录下建立一个 data 文件夹，将丝路说明中要求的 xml 格式的对齐文件放入 data 文件夹中，格式同表 6。
3) 输出文件也放在 data 文件夹中，分别为：
   en_ch.txt，英中单词概率文件，表示 w(c|e)，格式为：

```
personnel 队伍 0.0170758
constantly 经常 0.0833333
sound 声调 0.000423729
... ...
```

ch_en.txt，为中英单词概率文件，表示 w(e|c)，格式为：

```
Jakes Jakes 0.5
能 indulge 0.000129224
干警 police 0.166667
芯片 the 0.0483871
一致 NULL 0.185687
... ...
```

2.3.3.2 程序 2：短语抽取（在 pharaoh 文件夹中）

1) 主程序为 extract_phrase.cpp，如要修改文件名或路径，请在 definev.h 中修改。
2) 请先在当前目录下建立一个 data 文件夹,将丝路要求的 xml 格式的对齐文件放入 data 文件夹中，格式同表 6。
3) 输出文件在 data 文件夹下

Phrase_Extraction.txt，不带概率的短语抽取结果，格式如下：

```
BritneySpears ||| Britney Spears ||| 0-0 0-1
BritneySpears , ||| Britney Spears , ||| 0-0 0-1 1-2
BritneySpears , BackstreetBoys ||| Britney Spears , Backstreet Boys
||| 0-0 0-1 1-2 2-3 2-4
BritneySpears , BackstreetBoys , ||| Britney Spears , Backstreet
Boys , ||| 0-0 0-1 1-2 2-3 2-4 3-5
... ...
```

注：这个输出文件 Phrase_Extraction.txt 中有重复的短语。

2.3.3.3 程序 3：计算概率（在 com_prob 文件夹中）

1) 主程序为 com_prob.cpp，如要修改文件名或路径，请在 define.h 中修改。
2) 请先在当前目录下建立一个 data 文件夹,将以下运行程序所需的文件放入 data 文件夹中
   - Phrase_Extraction.txt,不带概率的短语抽取结果(也就是程序 2 的结果)
   - en_ch.txt，为英中单词概率文件（程序 1 的输出结果）
   - ch_en.txt，为中英单词概率文件（程序 1 的输出结果）
3) 输出文件在 data 文件夹下，为 Phrase_Probability.txt，是含有概率的短语抽取结果，格式如下：

```
, ||| , ||| 1 0.0683137 1 0.546206
BritneySpears , ||| Britney Spears , ||| 1 0.00406288 1 0.01701
, BackstreetBoys , ||| , Backstreet Boys , ||| 1 0.000825138 1 0.03945
BritneySpears , BackstreetBoys , ||| Britney Spears , Backstreet Boys ,
||| 1 4.90742e-005 1 0.00122855
, BackstreetBoys ||| , Backstreet Boys ||| 1 0.0120787 1 0.0722254
... ...
```

注：输出文件 Phrase_Probability.txt 剔除了重复的短语

2.3.3.4 参数说明

程序参数可以在 define.h 中修改：

- 抽取短语最大长度（只对中文短语限制了长度）：EXTRACTLEN 8
- 输入文件每行的最大长度：SENTENCELEN 3000
- 输入文件每句中的最大词数： PHRASEN 100
- 每个词中的最大字符数： PHRASEL 20

3 解码模块设计与使用

3.1 输入预处理

在进行解码前，需要对测试文件进行预处理，这一部分同 2.1 节的训练语料预处理是类似的，需要进行汉语的分词，以及汉语全角字符的替换。

3.2 语言模型接口使用说明

负责人：侯宏旭 hxhou@ict.ac.cn

我们在 SRILM 1.4.5 版本的基础上，利用原有函数库，编写了语言模型的调用接口。接口包含语言模型函数库 lmsridll.dll 和压缩函数库 zlib1.dll。编程接口包含 lmsridll.lib 接口函数库和 lmsridll.h 头文件。接口函数如下：

```
void * sriLoadLM(const char * fn, int arpa=0, int order=3, int unk=0, int tolow=0);
// 载入语言模型
// fn: 文件名，支持的文件类型有两种：普通文本文件和 gzip 压缩文件(扩展名为 gz 的
   文件)
// arpa: ARPA 格式和 BIN 格式，0 为 BIN 格式，非 0 为 ARPA 格式
// order: 读入元数，缺省为 3
// unk: 保留<unk>，缺省不保留
// tolow: 全部转化为小写，缺省不转化
// 返回语言模型的指针

int sriSetOrder(void * plm, int o);
// 设定最大元数
// plm: 语言模型的指针
// o: 新的元数，小于等于 0 时无效(不变)，大于语言模型数据的元数时按语言模型数据
   的元数计算
// 返回原来的元数

void sriUnloadLM(void * plm);
// 卸载语言模型
// plm: 语言模型的指针

double sriPerplexity(void * plm, const char * sentence);
// 计算句子困惑度
// plm: 语言模型的指针
// sentence: 句子
// 返回句子的困惑度
// perplexity(a1 a2 a3 ... an)=log10(P(a1)*P(a2|a1)*P(a3|a1a2)*...*P(an|a1...an-1))
```

```
double sriWordProb(void * plm, const char * word, const char * context);
//  计算 n-gram
// plm:  语言模型指针
//  返回值: log10(P(word|context))
```

3.3 “骆驼 CAMEL”解码器

负责人： 何中军 zjhe@ict.ac.cn

“骆驼 CAMEL”是一个统计机器翻译解码器，是“丝路”的一个组成部分。采用 Log-linear 直接翻译模型（Och & Ney, 2002）：

$$\Pr(e_1^I \mid f_1^J) = p_{\lambda_1^M}(e_1^I \mid f_1^J) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J)]}{\sum_{e'^I_1} \exp[\sum_{m=1}^{M} \lambda_m h_m(e'^I_1, f_1^J)]} \quad \text{（公式 6）}$$

CAMEL 采用了 7 个特征：

- Phrase translation probability $p(\tilde{e} \mid \tilde{f})$
- Inverse phrase translation probability $p(\tilde{f} \mid \tilde{e})$
- Phrase lexical weight $lex(\tilde{e} \mid \tilde{f})$
- Inverse phrase lexical weight $lex(\tilde{f} \mid \tilde{e})$
- English language model $lm(e_1^I)$
- English sentence length penalty I
- Distortion Model d

本小节将对 CAMEL 的模块定义、实现原理和使用方法进行详细说明。

3.3.1 模块定义

1) 输入文件：
   规范测试数据（已分词） 短语翻译概率表 英语语言模型
2) 输出文件：
   翻译结果文件（既可以输出 1-best，也可以输出 Nbest）
3) 需要调用的库函数：
   语言模型应用工具（SRI）
4) 功能说明：
   对输入的测试数据进行翻译，输出翻译结果

3.3.2 实现原理

给定一个汉语句子（已分词），CAMEL 解码过程如下：

1) Select Translation Options：穷举汉语句子的所有可能的短语，从双语短语表中读入对应的英语短语及概率
2) Compute Future Cost：根据 1）中选择的短语计算任意连续位置的汉语短语未来

概率

3) Beam Search：进行柱式搜索
4) Generate Nbest-list：回溯产生 Nbest

系统结构图如图 9：

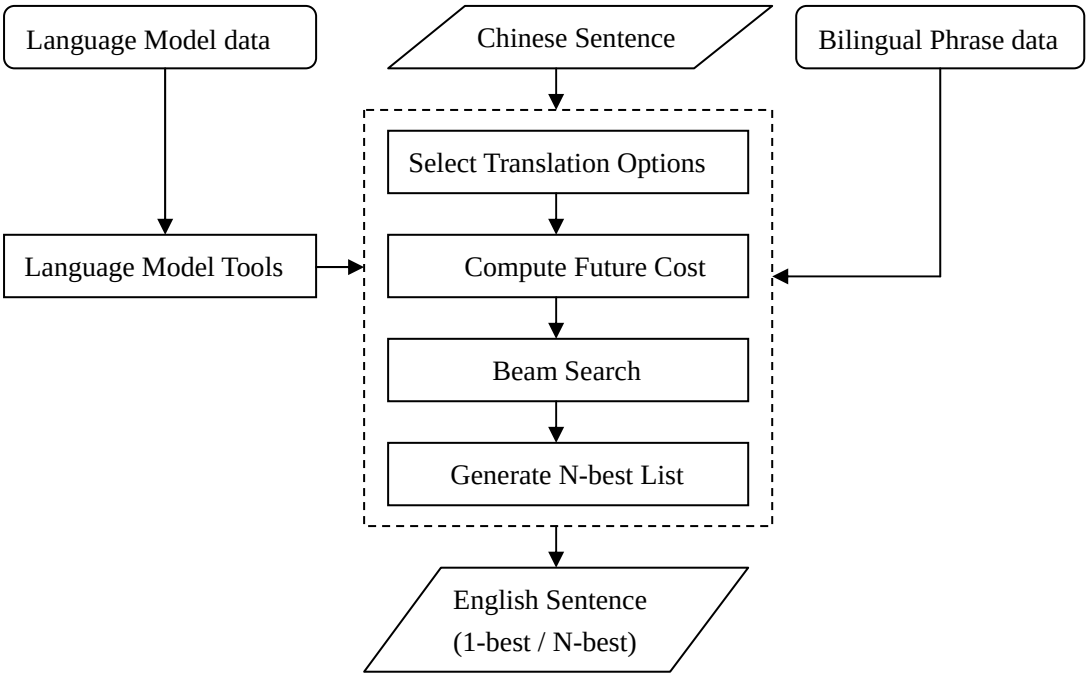

图 9：CAMEL 系统结构图

下面分别介绍这些模块的实现原理。

模块 1：选择候选短语（Select Translation Options）

一般的，一个双语短语表是非常庞大的，对于一个输入句子，我们只需要选择跟这个句子相关的短语（称为 Translation Options）就可以了，这样可以节省内存空间。为了便于解码，Translation Options 中需要记录以下信息：

- 汉语短语的起始、结束位置
- 汉语短语对应的英语翻译
- 翻译概率

算法非常简单，我们可以穷举一个汉语句子的所有可能短语，对每一个汉语短语，查找双语短语表，将双语短语表中汉语部分与之完全匹配的短语选择出来，

请参考算法 2：

```
for start=0 to ChineseWord.size()
{
    for end=start to ChineseWord.size()
   {
        if((start-end)>MAX_PHRASE_LEN)
             break;
        string  phrase = ChineseWord[start, end];
        Search translation options  for phrase from Bilingual Phrase table
   }
}
```

算法 2：Select Translation Options

模块 2：计算未来概率（Compute Future Cost）

在搜索过程中，我们以 Hypothesis 来存储英语短语翻译及概率等信息，并将已翻译相同个数汉语词语的 Hypothesis 存储到同一个栈中。为了减小搜索空间，采用宽度优先的柱式搜索（Beam Search）,这样需要对同一个栈中的 Hypothesis 进行剪枝。尽管同一个栈中存放的 Hypothesis 覆盖的汉语词个数相同，但是其覆盖的位置可能不同，在剪枝时，为了进行比较，不仅要考虑当前概率（已翻译词的概率），还要考虑未来概率（翻译目前为止还未翻译的词语需要的最大概率）。

未来概率是指如果要完成整个句子的翻译，剩余部分达到的最大概率（即最小代价），它跟当前还未翻译的词语相关，根据短语翻译概率、短语长度及语言模型进行估算，。

在 Translation Options 中，每一个汉语短语 $\tilde{f}_{start}^{end}$ 都对应一个或多个英语翻译，利用下式求得 $\tilde{f}_{start}^{end}$ 的最大翻译概率：

$$TP(\tilde{f}_{start}^{end}) = \max \sum \lambda_i \times \log(p_i(\tilde{e}, \tilde{f})) \qquad \text{（公式 7）}$$

其中，$p_i(\tilde{e}, \tilde{f})$ 指短语的 4 个翻译概率，英语短语长度，以及英语短语语言模型概率。然后，利用动态规划算法，可以很容易的求得任意连续串的未来概率，

请参考算法 1 算法 3：

```
for each ChinesePhrase in Translation Option
     TP( f̃_start^end ) = max Σ λ_i × log( p_i(ẽ, f̃) )
     FutureCost(start,end) = TP;

for len = 1 to ChineseWord.size()
     for i=0 to ChineseWord.size()-len
          for j=i to i+len
               double p= FutureCost(i,j) + FutureCost(j+1, i+len);
               if (p > FutureCost(i, i+len))
                    FutureCost(i, i+len) = p;
```

算法 3：Compute Future Cost

例如，对于汉语句子“中国 经济 发展 十分 迅速”，根据 Translation Options 和 $TP(\tilde{f}_{start}^{end}) = \max \sum \lambda_i \times \log(p_i(\tilde{e}, \tilde{f}))$ （公式 7 计算得到的汉语短语最大翻译概率如图 10：

中国 0	经济 1	发展 2	十分 3	迅速 4
-1.9118	-1.8484	-2.2069	-2.1182	-2.2024
-4.6019				
-4.7869				
	-2.5177			

图 10：汉语短语最大概率

则根据算法 3，

FutureCost(0, 1) = FutureCost(0, 0) + FutureCost(1, 1) = -3.7602 （公式 8）

同样的也可以计算其他任意连续位置的 FutureCost。

在解码时，如果未翻译的汉语词是不连续的，那么我们将各个连续位置的 FutureCost 相加，作为这个 Hypothesis 的 FutureCost。例如，如果一个假设 h 翻译了单词“发展”，则未翻译的片断为：（0, 1），（3, 4），那么

FutureCost（h）= FutureCost（0, 1）+ FutureCost（3, 4） （公式 9）

模块 3：柱式搜索（Beam Search）

搜索是解码器的核心，Translation Options 的选择和 Future Cost 的计算都是为搜索作准备的。CAMEL 采用柱式搜索（Beam Search）策略，每次只保留最好的 N 个决策，实际上，它是一种宽度优先搜索算法。

对于一个汉语句子（已分词）$c_1\ c_2\ \cdots\ c_n$，解码器每次都根据 Translation Options 选择句子中未翻译的一个片断（即短语）$c_i\ \cdots\ c_j$ 进行翻译，根据公式 4 计算相应概率，产生英语短语翻译，并将这些信息存储于 Hypothesis 中，根据已经翻译的单词个数 m 将新产生的

Hypothesis 存储到相应的栈 stack(m)中。当一个句子翻译完毕时，关于这个句子的完整翻译信息可以从最后一个栈 stack(n)中向前回溯得到。

请参考算法 4：

```
initialize HypothesisStack[0, ..., nf ]
create initial Hypothesis hp_init and push to HypothesisStack[0];
for i=0 to  nf-1
  for each hypothesis in HypothesisStack[i]
     for each new_hp that can be derived from hp
       covered_number = number of foreign words coverd so far;
       push new_hp to HypothesisStack[covered_number];
       prune HypothesisStack[covered_number];
```

算法 4：Beam Search

下面对 Hypothesis 数据结构和剪枝策略进行详细说明：

1) Hypothesis 数据结构

在搜索过程中，各种信息都存储在 Hypothesis 中，翻译一个句子的过程，也就是扩展 Hypothesis 的过程，Hypothesis 中需要包含以下信息：

- 指向父亲结点的指针（便于回溯产生路径）
- 目前为止翻译的汉语词（一个 bool 型的数组，已翻译的词标记为 1，未翻译的词标记为 0）
- 上一次翻译的汉语短语最后一个汉语词的位置（计算扭曲概率）
- 目前为止产生的最后两个英语词（计算 3-gram 语言模型）
- 本次扩展的汉语短语对应的英语翻译
- 特征函数（包括翻译概率、英语短语长度、语言模型概率、扭曲距离）
- 目前为止的概率 P
- 未来概率 FC
- 估计概率 EC（EC=P+FC）
- 被合并掉的其他边的信息 Additional Arc（父亲节点、英语译文，概率）

2) 剪枝策略 1 —— Recombining Hypothesis

第一种方法是 Hypothesis 的合并，这是一种没有风险（risk free）的剪枝策略，也即合并不会损失任何信息。如果两个假设 Hp1 和 Hp2 满足以下三个条件，我们就可以将它们合并：

- 覆盖的汉语词位置相同
- 最后产生的两个英语词相同（如果是 3-gram 语言模型）
- 前一次翻译的汉语短语最后一个词的位置相同

如果 Hp1 的估计概率 EC1 大于 Hp2 的估计概率 EC2，那么我们就保留 Hp1 继续扩展，而将 Hp2 的相关信息（父亲节点、英语译文，概率）保存到 Hp1 中的 Additional Arc 中，反之则将 Hp1 合并到 Hp2 中。这样概率低的 Hypothesis 不会再继续扩展，但是在回溯的时候仍然可以根据保留的信息将其找回。

3) 剪枝策略 2 —— Histogram Pruning

第二种方法是柱状图剪枝，这是一种有风险的剪枝策略，被剪掉的边将从栈中

删除，以后也不会再扩展，无法回溯找到。

解码器将每个栈的大小限定为一个特定值 N。搜索时，如果栈中的元素个数小于 N，则可以继续压入元素；如果栈中的元素个数等于 N，那么就将栈中概率最小（即估计概率 EC 最小）的元素删除，然后再压入元素。

模块 4：译文的产生（Generate N-best List）

搜索完毕，可以通过回溯产生最终译文。一般的，我们只产生一个最好的译文（1-best），即从最后一个栈中找到概率最大的 Hypothesis，根据其指向父亲节点的指针向前回溯。有些时候，比如训练 Log-linear 模型的参数（Och，2003）或者用 N-best 做 Rerank 等，可能会需要解码器输出多个译文（N-bset），我们利用 A*算法来产生 N-best。下面分别对这两种方法进行介绍。

1）Generate 1-best

产生 1-best 的方法非常简单，只需要从最后一个栈中找到概率最大的 Hypothesis 向前回溯即可。

请参考算法 5：

```
find the Hypothesis with the highest probability in HypothesisStack[nf];
english_translation = hyp_best.english;
father_hyp = hyp_best.father;
while father_hyp != hyp_init
   english_translation = father_hyp.english + english_translation;
   father_hyp = father_hyp.father;
output english_translation;
```

算法 5：Generate 1-best

2）Generate N-best

有很多种方法可以产生 N-best，如利用有限状态自动机工具 Carmel（http://www.isi.edu/licensed-sw/carmel/）。CAMEL 采用 A*算法（Och et al. 2001）来产生 N-best。

3.3.3 使用说明

本节介绍 CAMEL 的使用方法。

3.3.3.1 安装

- 硬件要求：

  PC 机：P4 2.0GHz 以上 CPU，512M 以上内存，硬盘 40G 以上。

- 软件要求：

  CAMEL V1.0 使用 C++开发，可以运行在 GNU/Linux 平台和 Win32 平台。以下两种编译器都可以进行编译：

  1) GNU C++编译器 Version3.3 或更高
  2) Microsoft C++ 7.1

- Linux 下的安装

  下载 CAMEL_v1.0_Linux.tar.gz，将其拷贝到任一目录，使用下面的命令解压缩：

  ```
  tar –zxvf CAMEL_v1.0_Linux.tar.gz
  ```

  进入 CAMEL 目录，键入 make 命令进行编译：

  ```
  $ make
  ```

  如果编译过程中语言模型链接出错，可能是由于 Linux 版本不同所致。请重新编译语言模型库，覆盖程序包中 srilm-lib/下的.a 文件，并重新编译。方法如下：

  a) 将程序包中的 srilm.tar.gz 解压缩
  b) 在 srilm/目录下，键入 make 进行编译
  c) 编译完成，在 srilm/目录下会生成 lib/目录，将其下面的.a 文件拷贝到 Camel 软件包中的 srilm-lib/目录下，覆盖里面的.a 文件
  d) 重新编译 Camel

3.3.3.2 运行解码器

键入以下命令，运行解码器：

```
CAMEL –conf configurefile –test-file testfile –result-file resultfile
```

解码器读入配置文件和测试文件，输出结果文件。下面详细介绍配置文件及命令行参数。

1）配置文件

配置文件对解码器的各项参数进行设置，格式如表 18：

```
[ttable-limit] 10
[stack] 100
[nbest-list] 200
[distortion] 0

[ttable-file] bp_4k_12_11.txt
[lm-file] 30k3gram.gz
[lm-ngram] 3

[para] ##
p(c|e) 0.03
lex(c|e) 0.03
p(e|c) 0.15
lex(e|c) 0.16
len 0.48
lm 0.13
dis 0
[end] ##

[print-info] 0
[print-nbest] 0
```

表 18：解码器配置文件格式

2）文件参数

- [ttable-file] 指定双语短语文件的位置
- [lm-file] 指定语言模型的位置
- [lm-ngram] 指定语言模型的元数，默认是 3-gram

3）特征函数的权重

CAMEL 采用 Log-linear 直接翻译模型，共有 7 个特征，这些特征的权重在配置文件中设置——“[para] ##” 和 “[end] ##”中间的部分。其中前 4 个是短语翻译概率的权重，len 是英语句子长度的权重，lm 是语言模型的权重， dis 是短语扭曲模型的权重。

4）剪枝参数

CAMEL 采用 2 种类型的剪枝：对短语表进行剪枝以限制读入短语的数量，搜索过程中对搜索栈进行剪枝。

- [ttable-limit] 对双语短语表进行剪枝

由于双语短语表是从平行语料库中自动抽取得到的，同一个汉语短语可能对应多个甚至上百个不同的英语短语翻译。一种方法是，对同一个汉语短语，只取“最好”的 N 个英语短语，在我们的系统中，默认是 10。

所谓“好”的翻译，我们用短语翻译概率来衡量，概率越大越“好”：

$$P = \sum \lambda_i \times \log(p_i(\tilde{e}, \tilde{f})) \qquad \text{(公式 10)}$$

其中， $p_i(\tilde{e}, \tilde{f})$ 表示短语翻译的 4 个概率， λ_i 表示对应的权重。

这个参数也可以在命令行中进行设置：

Parameters
-ttable-limit: number of phrase translations for each foreign phrase (default 10)

- [stack] 对搜索栈进行剪枝

解码器采用 Beam-Search 方法进行搜索，数据存放于栈中。在搜索过程中，对栈的大小进行限制，以减少搜索空间，更详细的说明请参考第 2 章 Beam Search。在我们的系统中，栈的大小默认是 100。

这个参数也可以在命令行中进行设置：

Parameters
-stack: maximum size of the beam (default 100)

5）[distortion]

对短语扭曲距离进行限制。扭曲模型的计算公式如下：

$$P_d = -\sum_i d_i \qquad \text{（公式 11）}$$

其中，d = abs(last word position of previously translated phrase + 1
- first word position of newly translated phrase)

对短语扭曲距离进行限制，也可以极大的减少搜索空间。默认值是 5，即 d<=5。如果将[distortion]设为 0，则不进行调序，即单调搜索。

这个参数也可以在命令行中进行设置：

Parameters
-distortion: maximum distance between two foreign phrases (default 5, 0 for monotone search)

6）[nbest-list]

对于一个源语言句子，解码器可以产生 1 个或多个目标语言句子。默认的 Nbest 大小是 200。

这个参数也可以在命令行中进行设置：

Parameters
-nbest-list: number of candidate translations for each input sentence (default 100)

7）[print-info]

用于输出对一个句子进行解码的详细信息，包括可适用的英语短语、未来概率（Future Cost），搜索过程，以及回溯产生的 Nbest 路径。如果[print-info]设为 1， 则解码器会产生一个 search_info.xml 文件，包含 4 部分内容，格式如下：

第一部分，显示 TranslationOption，找到所有能够适用于该句子的英语短语

```
<?xml version="1.0" encoding="gbk" ?>
<translog>
<srcsent>中国 经济 发展 十分 迅速 </srcsent>
<TransOption num="8">
<src phrase="中国" beg="0" end="0">
<trans lex="China"> -0.282683 | -0.242107 | -0.995656 | -0.391329 | 1 </trans>
<trans lex="China 's"> -0.331812 | -0.69002 | -1.76154 | -1.87886 | 2 </trans>
</src>
<src phrase="中国经济" beg="0" end="1">
<trans lex="China 's economic"> -0.344841 | -0.916413 | -0.904456 | -2.43623
| 3 </trans>
<trans lex="China 's economy"> -0.241162 | -1.04799 | -1.33977 | -2.98078 | 3
</trans>
</src>…
</TransOption>
```

表 19：TranslationOption

其中，<trans lex="China">后面，前四个数值是短语翻译概率取 log 以后的数值，第 5 个数值是英语短语的长度。

第二部分，显示 FutureCost，根据 Translation Options 来估算翻译所有连续汉语片段的最大概率（即最小代价）。

```
<FutureCost>
<cost beg="0" end="0"> -4.85626</cost>
<cost beg="0" end="1"> -6.62409</cost>
<cost beg="0" end="2"> -7.11725</cost>
<cost beg="0" end="3"> -17.5915</cost>
<cost beg="0" end="4"> -27.7396</cost>
<cost beg="1" end="1"> -7.43527</cost>
<cost beg="1" end="2"> -3.31509</cost>
<cost beg="1" end="3"> -13.7893</cost>
<cost beg="1" end="4"> -23.9374</cost>
<cost beg="2" end="2"> -5.58266</cost>
<cost beg="2" end="3"> -16.0569</cost>
<cost beg="2" end="4"> -26.205</cost>
<cost beg="3" end="3"> -10.4742</cost>
<cost beg="3" end="4"> -20.6223</cost>
<cost beg="4" end="4"> -10.1481</cost>
</FutureCost>
```

表 20：FutureCost

第三部分，显示搜索栈中的信息。在搜索过程中，以 Hypothesis 来保存信息，将 Hypothesis 按照已翻译的汉语词个数保存到相应的栈中。

```
…
<Stack ID="3" size="4">
<ID stack="3" number="0">
<Hypothesis Father="(2,1)">
<Covered num="3">1 1 1 0 0 </Covered>
<CurrentTranslation> development </CurrentTranslation>
<LastEnglishWord> economy development </LastEnglishWord>
<CoveredWordPosition LastEnd="1">(2, 2) </CoveredWordPosition>
<Feature FutureCost="-20.6223" EstimateProb="-36.2644">-0.424533 -0.428234
-0.954138 -0.399975 1 -5.94549 0 # total probability:-15.6421 , sigma function:
-7.15237</Feature>
<AddArc num="1">
<Arc No="0">(1,0) | economy development | 0 -0.786202 -3.16407 -1.50189 2
-5.68787 0 # total probability:-16.3624 , sigma function: -9.14003</Arc>
</AddArc>
</Hypothesis>
</ID>
…
```

表 21：Stack information

<Stack ID="3" size="4">表示第 3 个栈（也即已经翻译了 3 个汉语词），栈中有 4 个元素；

<ID stack="3" number="0"> 表示第 3 个栈中第 0 个 Hypothesis；

<Hypothesis Father="(2,1)"> 与</Hypothesis>中间是 Hypothesis 中的信息，Father="(2,1)" 表示其父亲结点是第 2 个栈中的第 1 个 Hypothesis（注意，栈中的 Hypothesis 从 0 开始编号）；

<Covered>与</Covered>显示了已经翻译的汉语词的信息；

<CurrentTranslation>与</CurrentTranslation>表示本假设生成的英语翻译；

<LastEnglishWord>与</LastEnglishWord>表示到目前为止，最后产生的 2 个英语单词（如果是 3 元语言模型的话）；

<CoveredWordPosition LastEnd="1">(2, 2) </CoveredWordPosition>表示本假设翻译的汉语短语位置是（2，2），上一次翻译的汉语短语最后一个位置是 1，用于计算扭曲模型；

<Feature FutureCost="-20.6223" EstimateProb="-36.2644">与</Feature>显示概率信息，分别是短语翻译概率（前 4 个），短语长度，语言模型概率，扭曲距离；total probability 表示到目前为止走过的路径的概率和，sigma function 仅指本假设概率的加权和，即

$$P = \sum \lambda_i \times \log(p_i(\tilde{e}, \tilde{f})) \qquad \text{（公式 12）}$$

FutureCost 可以查表得到，EstimateProb = total probability + FutureCost；

<AddArc num=”1”> 与</AddArc>表示搜索过程中被合并掉的边，num 表示边的个数。

每一个<Arc>都是一条被合并掉的边，记录了其父亲结点，英语翻译，以及概率信息，用于生成 Nbest list。

第四部分，显示 Nbest list，即搜索产生的 N 个候选翻译。

```
<Nbest_list Number = "2">
<Candidate No="1">
<translation>China 's economic development extremely rapidly</translation>
<feature>-0.652326  -2.1119  -2.10987  -4.23355  6  -27.0231  0    #  total
probability:-30.1308 , sigma function:  -30.1308</feature>
</Candidate>
<Candidate No="2">
<translation>China 's economic development extremely rapid</translation>
<feature>-1.75094  -2.67437  -3.28852  -5.18906  6  -27.0494  0    #  total
probability:-33.9523 , sigma function:  -33.9523</feature>
</Candidate>
</Nbest_list>
```

表 22：Nbest list

这个参数也可以在命令行中进行设置：

Parameters

-print-info: print decoding information or not (default 0)

8）[print-nbest]

如果只想查看 Nbest 信息，那么可以将本参数设为 1。输出格式如下：

```
<?xml version="1.0" encoding="GB2312"?>
<text>
<sent No="1" nbest="2">
<chinese>中国 经济 发展 十分 迅速</chinese>
<candidate No="1">
<translation>China 's economic development extremely rapidly</translation>
<feature>-0.652326 -2.1119 -2.10987 -4.23355 6 -27.0231 0 </feature>
</candidate>
<candidate No="2">
<translation>China 's economic development extremely rapid</translation>
<feature>-1.75094 -2.67437 -3.28852 -5.18906 6 -27.0494 0 </feature>
</candidate>
</sent>…
<sent No="2" nbest="10">
……
```

表 23：单独的 Nbest List

其中，<sent No=”1” nbest=”2”> 表示第 1 个汉语句子，有 2 个英语译文。

默认的输出文件是“nbest.xml”，也可以在命令行中指定：

Parameters

-print-nbest: print nbest-list or not (default 0)

-nbest-file: specifies nbest-list file (if -print-nbest is set to 1, nbest-list will be dumped to this file)

9）输入输出参数

指定输入文件和输出文件。

Parameters

-test-file: specifies test file

-result-file; specifies result file

3.4 “商队 Caravan”解码器

负责人：陈毅东 ydchen@xmu.edu.cn

3.4.1 模块定义

1) 输入文件：
   规范测试数据（分词或未分词，目前只处理 gb 编码和 utf-16le 编码）
   短语翻译概率表
   英语语言模型
2) 输出文件：
   翻译结果文件（目前仅输出 1-best）
3) 需要调用的库函数：
   语言模型应用工具（SRI）
   Neon 系统 dll（如果希望使用史晓东教授的 neon 系统翻译未登录词则需要该模块）
   SegTag 系统 dll(如果希望使用史晓东教授的分词系统 SegTag 实现分词则需要该模块）
   ICTCLAS 系统 dll（如果希望使用计算所分词系统 ICTCLAS 实现分词则需要该模块）
4) 功能说明：
   对输入的测试数据进行翻译，输出翻译结果

3.4.2 实现原理

本节简要描述 Caravan 的的实现原理。

3.4.2.1 模型

Caravan实现了基于短语的统计机器翻译方法（Zens et al., 2002）。短语可以是任意长度的连续字符串，翻译模型采用Log-linear 直接翻译模型（Och & Ney, 2002）：

$$\Pr(e_1^I \mid f_1^J) = \frac{\exp[\sum_{m=1}^{M} \lambda_m \cdot h_m(e_1^I, f_1^J)]}{\sum_{e_1^I} \exp[\sum_{m=1}^{M} \lambda_m \cdot h_m(e_1^I, f_1^J)]} \quad \text{（公式 13）}$$

其中，Caravan采用了如下6个特征：

- 短语翻译概率： $p(\tilde{e} \mid \tilde{c})$
- 反向短语翻译概率： $p(\tilde{c} \mid \tilde{e})$
- 词汇化的短语翻译概率： $lex(\tilde{e} \mid \tilde{c})$
- 反向词汇化的短语翻译概率： $lex(\tilde{c} \mid \tilde{e})$
- 英语语言模型： $lm(e_1^I)$
- 英语句子长度： I

3.4.2.2 搜索算法

Caravan 实现了单调的基于词组的解码算法（Zens et al., 2002）。采用动态规划算法，相应的递归公式如下：

$$Q(0,\$) = 1 \qquad \text{（公式 14）}$$

$$Q(j,e) = \max_{\substack{0 \le j' < j, \\ e', \tilde{e}}} Q(j',e') \cdot p(f_{j'+1}^{j} \mid \tilde{e})^{\lambda_1} \cdot lex(f_{j'+1}^{j} \mid \tilde{e})^{\lambda_2} \cdot p(\tilde{e} \mid f_{j'+1}^{j})^{\lambda_3} \cdot lex(\tilde{e} \mid f_{j'+1}^{j})^{\lambda_4} \cdot p(\tilde{e} \mid e')^{\lambda_5} \cdot len(\tilde{e})^{\lambda_6}$$

（公式 15）

$$Q(J+1,\$) = \max_{e'} Q(J,e') \cdot p(\$ \mid e') \qquad \text{（公式 16)}$$

3.4.3 使用说明

本节介绍 Caravan 的使用方法。

3.4.3.1 安装

下载 CARAVAN.RAR 并在本地解压后即完成安装。压缩包中附带了一个双语短语文件和一个语言模型文件，双语短语文件为计算所提供的 bp_nlpr_ict.txt 经剪枝后的文件（见“数据文件的说明”）；语言模型文件为计算所提供的 all_en.bo3。请参考“配置文件格式”一节以了解如何修改配置文件以使用新的数据文件。

压缩包中的 src 目录下为源代码，可使用 Borland Delphi 7.0 或以上版本打开并修改以重新生成新版本。

3.4.3.2 数据文件的说明

双语短语翻译表是短语抽取模块的输出文件，格式见 1.5.5 节。

为了保证解码器翻译速度，双语短语翻译表中包含同一个汉语短语的条目建议不要超过 20 条。可以根据如下公式的计算结果对短语翻译表进行剪枝：

$$p = \lambda_1 \times p(\tilde{c} \mid \tilde{e}) + \lambda_2 \times lex(\tilde{c} \mid \tilde{e}) + \lambda_3 \times p(\tilde{e} \mid \tilde{c}) + \lambda_4 \times lex(\tilde{e} \mid \tilde{c}) \qquad \text{（公式 17）}$$

我们在压缩包 CARAVAN.RAR 中提供了 DELBP 工具可实现上述剪枝，使用方式如下：

DELBP <bp File> <pruned bp File> Np Lambda1 Lambda2 Lambda3 Lambda4

其中：

- <bp File>：剪枝前的双语短语文件路径
- <pruned bp File>：剪枝后的双语短语文件路径
- Np：包含同一个汉语短语的最大行数
- Lambda1~Lambda4：分别对应 $\lambda_1 \sim \lambda_4$。这四个参数应与解码器的模型参数保持一致。

3.4.3.3 命令行参数

准备好数据文件和正确的配置文件后，可以使用如下命令行格式来运行 Caravan 以实现文件翻译：

Caravan <.ini File> s-flag r-flag <Src File> <Tgt File>

其中：

- <.ini File>指定配置文件的路径
- s-flag 指定分词选项，有三种：
  - none，表示不分词（适用于输入文件已分词的情况）
  - mandel，表示使用史晓东教授的分词系统 segtag

- ict，表示使用计算所的分词系统 ICTCLAS
- r-flag 表示后处理选项，有两种
  - ppb，表示不进行后处理
  - pbnw，表示使用史晓东教授的 neon 系统进行未登录词的翻译
- <Src File>指定输入文件的路径
- <Tgt File>指定输出文件的路径

3.4.3.4 配置文件格式

配置文件对解码器的各项参数进行设置，格式如下：

```
[ce]

cemtsdk_path=cemtsdk\
segtag_path=segtag\
ictclas_path=ict\
lm_path=lm\

language_model_file=ce\all_en.bo3
bilingual_phrases_file=ce\bp_nlpr_ict.txt

p_c_e=0.03
lex_c_e=0.03
p_e_c=0.15
lex_e_c=0.16
word_penalty=0.48
language_model=0.13
```

1) 外部系统的路径
- cemtsdk_path：指定史晓东教授的 neon 系统的 dll 路径（请使用相对路径）
- segtag_path：指定史晓东教授的分词系统 segtag 的 dll 路径（请使用相对路径）
- ictclas_path：指定计算所分词系统 ICTCLAS 的 dll 路径（请使用相对路径）
- lm_path：指定计算所提供的语言模型 dll 路径（请使用相对路径）

2) 数据文件的路径
- language_model_file：指定英语言模型文件路径（请使用相对路径）
- bilingual_phrases_file：指定双语短语翻译表文件路径（请使用相对路径）

3) 模型参数
- p_c_e： $p(\tilde{c}\,|\,\tilde{e})$ 的参数
- lex_c_e： $lex(\tilde{c}\,|\,\tilde{e})$ 的参数
- p_e_c： $p(\tilde{e}\,|\,\tilde{c})$ 的参数
- lex_e_c： $lex(\tilde{e}\,|\,\tilde{c})$ 的参数
- word_penalty：英语句子长度权重
- language_model：语言模型的权重

3.5 “绿洲 Oasis”解码器

负责人：蒋宏飞 hfjiang@mtlab.hit.edu.cn

解码器在实现上，采用了基于短语的统计翻译模型， SRI 的 ARPA 标准格式的语言模型和 Bean Search 算法。输入源语言的句子，通过调用双语的短语翻译表和语言模型表，搜索产生目标语言的句子。

3.5.1 模块定义

1) 输入文件：
   规范测试数据（已分词） 短语翻译表 英语语言模型
2) 输出文件：
   翻译结果文件（既可以输出 1-best，也可以输出 Nbest）
3) 功能说明：
   对输入的测试数据进行翻译，输出翻译结果

3.5.2 实现原理

解码部分的实现主要参考了 Koehn 的 pharaoh 系统说明文档，其解码的过程可分为以下几步：

a. 获取候选短语。根据输入的已分词的汉语句子，从双语词表中顺序切分的汉语短语的片断获取对应得英语短语及其概率信息；

b. 计算未来概率。在已获取的英语候选短语的基础上，动态规划的计算任意连续位置间的概率情况；

c. 产生扩展假设。英语句子按照从左至右的顺序产生，用候选短语进行扩展，计算出相应得代价，进行 beam search。

根据 beam search 中每一步假设的代价，选择最低的假设，回溯产生英语译文。

- **假设的数据结构**
  到目前为止的概率
  新扩展短语的翻译概率
  扩展短语后的扭曲值
  对新扩展短语长度的惩罚
  扩展短语的语言概率
  此时的未来概率
  扩展短语对应的外文句子中短语的位置
  扩展的短语
  展后英语句子的最后两个词（三元模型的情况）
  ID 号
  指向来源假设的指针
- **剪枝的实现**
  包括两种类别：
  1）假设的合并，满足以下条件的假设可进行合并
  - 到当前假设为止，所覆盖的外语的短语相同
  - 最后产生的两个英语词相同（如果是 3-gram 语言模型）
  - 上一次翻译的外文短语最后一个词相同

2）beam 剪枝

通过设定阈值和栈的大小来实现。当假设入栈时，保留最佳的代价值，用该代价值和阈值的和作为后来假设入栈时的参考阈值，将低于参考阈值的假设丢弃；当栈内假设的数目高于一定值（程序设定为栈大小的 2 倍），进行修剪，丢弃最差的 n 条假设

3.5.3 使用说明

3.5.3.1 硬件环境

CPU: P4 2.0GHz 以上
内存: 512M, 推荐 1G 以上

3.5.3.2 软件环境

解码器实现采用了标准的 C ++, 可不做修改运行在 Linux 平台和 Windows 平台。
Windows 平台下采用 Visual C++ 6.0
Linux 下推荐较高版本的 C++编译器（开发时采用：gcc 3.4.2）

3.5.3.3 运行程序

Windows 下 Visual C++，新建 Win32 Console Application, 加入压缩包中的文件即可。

在命令行输入：

SilkRoad.exe –f pharaoh.ini –in input.txt –out output.txt

Linux 下，解压，进入目录 Decoder,键入 make, 即可编译

在命令行输入：

./SilkRoad –f pharaoh.ini –in input.txt –out output.txt

3.5.3.4 解码器配置文件

配置文件对解码器的参数进行设置，其余参数可在命令行进行设置，详见下文。

phrase table f, n, p(n|f)
[ttable-file]
短语翻译表的路径

language model
[lmodel-file]
英语语言模型文件的路径

vocabulary files (not really needed)
[vocab-file-foreign]
外文词表

[vocab-file-native]
英文词表

limit on how many phrase translations e for each phrase f are loaded
#ttable element load limit 0 = all elements loaded
[ttable-limit]

20

distortion (reordering) weight
[weight-d]
0.2

language model weight
[weight-l]
0.5

translation model weight (phrase translation, lexical weighting)
[weight-t]
0.2
0.2
0.2
0.2
0.2（可以不要）

word penalty
[weight-w]
-1

解码中采用了 Log-linear 直接翻译模型（Och & Ney, 2002），分别是：[weight-d], [weight-l], [weight-t], [weight-w];其中[weight-d] 是短语扭曲模型的权重，[weight-l] 是语言模型的权重，

[weight-t]是短语翻译概率的权重，[weight-w]是英语句子长度的权重。

3.5.3.5 参数说明

- **[ttable-limit]** 对双语短语表进行剪枝

由于双语短语表是从平行语料库中自动抽取得到的，同一个汉语短语可能对应多个甚至上百个不同的英语短语翻译。在解码实现时，采用了选取top n的方式。其中top n的选取综合考虑了以下几个因素：短语翻译的概率，对应英语短语的语言模型概率和英语短语的长度。对以上三个因素通过加权的方式组合，选取top n，在实现时，默认值是20。

$$c = weight\_l * lm + weight\_t * tm + weight\_w * len \qquad （公式 18）$$

该参数可以在配置文件中进行修改。

- **[stack]和[threshold]** 对栈的搜索进行剪枝

解码器采用Beam-Search 方法进行搜索，数据存放于栈中。在搜索过程中，对栈的大小进行限制，以减少搜索空间，同时采用了相关阈值来对入栈的假设进行限制。

这两个参数可以在命令行上指定，默认值分别是100和0.00001：

-s：maximum size of the beam (default 100)

-b：minimum value of a hypothesis relative to the best hypothesis in a stack (default 0.00001)

- **[distortion]** 对短语扭曲距离进行限制

扭曲模型的计算公式如下：

$$P_d = -\sum_i d_i \qquad \text{（公式 19）}$$

其中，d = abs(last word position of previously translated phrase + 1
- first word position of newly translated phrase)

通过使用配置文件中的 weight-d，对短语扭曲距离进行加权评分，计入对每条假设的评分中。

大幅度的调序会对翻译的性能产生负面的影响，对短语距离模型的限制，可以减少搜索的空间，提高效率和性能。在实现中进行了两个方面的限制：距离和阈值。距离的默认值为 9(为 0 时，不进行限制)，当距离超过设定值后，扭曲模型的值都采用默认阈值，阈值为 0.1。

这两个参数都可以在命令行进行设定。

-dl：Maximum distance between two input phrase that are translated to two neighboring output phrase

-d：Minimum of the distortion score

- **[lm-limit]** 英语语言模型值

对对应的英语短语以及未登录词的概率设定最小值，以利于评分，搜索。由于采用的是 Log-linear 直接翻译模型（Och & Ney, 2002），该值的设定参照了 pharaoh 的 release 版的设定，将最小值设定为-10。

这个参数可以在命令行中进行设定，根据语言模型的规模来设定

-m：minimum score of the language model of the phrase

- **[nbest]** 输出前 n 个最好的译文

默认输出一个，这个参数可在命令行指定

3.5.3.6 命令行参数功能说明

-f	Specify the Configuration file
-in	Specify the input data
-out	Specify the output data
-s	Specify the maximum size of the beam, default 100
-b	Specify the beam threshold, default 0.00001
-l	Specify the N-best output, default 1
-d	Specify the minimum distortion score, default -2.30259
-dl	Specify the distortion distance, default -9
-m	Specify the minimum lm score, default -10

3.5.3.7 调试信息输出

在调试模式下，可以输出很多解码的详细信息，通过在编译时定义-NDEBUG（Linux 下，Windows 下 /D NDEBUG）来实现。（辅助信息未加入 release 中，如有需要只需简单修改）

具体输出包括：候选短语表，未来概率表，每一个的假设元素的详细信息和剪枝等等。

以“同样 的 东西 有 浅 褐色 的 吗 ？”为例

- **Translation Options**

输出候选短语，其中，[]中的词是中文句子分词后短语片断，最长值设定为 10，以下第一列（以“,”分隔）为对应的抽取出的英语短语，翻译概率，加权的选择概率（详见

$$c = weight\_l * lm + weight\_t * tm + weight\_w * len$$ （公式 18））

[同样]
the same, -1.28215, -4.05183
[的]
in, -1.8011, -2.94581
right, -2.71656, -4.88024
of a, -2.82411, -4.96468
flight, -2.62181, -4.90627
's, -2.5144, -4.40283
of the, -2.21918, -3.63043
[东西]
anything, -2.03706, -4.64327
came, -2.2911, -5.35805
……
……
……
[褐色 的]
brown, -1.71482, -4.66026
[吗 ？]
, please ?, -2.64761, -2.90795
., -5.36997, -5.80588
? i, -2.6503, -4.02195
?, -0.235171, -0.85651
can have ?, -4.116, -8.81662
have ?, -3.08104, -6.56532
it ?, -4.36051, -6.22888
please ?, -2.06406, -4.79739
you ?, -2.2497, -4.72949
you have ?, -3.61164, -6.32542
[的 吗 ？]
, right ?, -3.87216, -5.2558
?, -1.20302, -1.82436
at ?, -4.04125, -7.54385
size ?, -2.50204, -5.17914

- **Future cost**

动态规划实现，用于在解码时估计假设的未来代价。

future costs from 0 to 0 is -4.05183
future costs from 0 to 1 is -4.95449
……
future costs from 0 to 7 is -22.5568
future costs from 0 to 8 is -19.8797
future costs from 1 to 1 is -2.29265
future costs from 1 to 2 is -4.18843

future costs from 1 to 3 is -6.24738
future costs from 1 to 4 is -10.3111
future costs from 1 to 5 is -14.3111
future costs from 1 to 6 is -14.9713
……
……
future costs from 6 to 7 is -5.82631
future costs from 6 to 8 is -1.82436
future costs from 7 to 7 is -3.53366
future costs from 7 to 8 is -0.85651
future costs from 8 to 8 is -0.68513

- **Hypothesis Element**

解码中各个假设的详细信息，包括当前假设的 ID，翻译和语言模型的评分，总的评分，是否丢弃，栈此时的情况等。

creating hypothesis 1 from 0
base score 0
translation cost -1.28215
distortion cost 0
language model cost for 'same' -2.57302
language model cost for 'the' -1.91582
word penalty 2
score -3.77099 + futureCost -15.8278 = -19.5988
new best estimate for this stack
merged hypothesis on stack 1, now size 1

3.6 输出后处理

后处理过程对解码器输出的译文进行以下操作：将英文句子的首字母大写。

程序使用命令行：

UpperLowerRevert.exe File.txt

4 评测工具简介

本小节介绍自动评价工具 mteval，该工具是 863 机器翻译的评测工具，由中科院计算所开发。

4.1 评价指标

- BLEU

BLEU（Papineni et al., 2001）是目前国际机器翻译评测最常用的指标。这是一种基于 N-Gram 的自动评测方法，它通过对系统译文跟参考译文进行 N-Gram 的比较，对每一个系统译文计算 N-Gram 准确率，最终对整个译文进行打分，其计算公式如下：

$$score = BP * \exp(\sum_{n=1}^{N} w_n \log p_n) \quad \text{（公式 20）}$$

其中，

$$BP = \min\{1, \exp(1 - \frac{L_{ref}}{L_{sys}})\} \quad \text{（公式 21）}$$

$$p_n = \frac{\sum_{s=1}^{S} \delta_n(e_s, r_S)}{\sum_{s=1}^{S} c_n(e_s, r_S)} \quad \text{（公式 22）}$$

其中，BP 为长度惩罚因子，L_{ref} 为被测句子最短的参考译文长度，L_{sys} 为被评测句子的译文长度（即整个系统输出译文中含有的单词数目）；$\delta_n(e_s, r_S)$ 是 e_s 与参考译文匹配的 n-gram 个数，$c_n(e_s, r_S)$ 是系统输出译文 e_s 中 n-gram 的个数；N 为最大 n-gram 长度，W_n 为 n-gram 的权重。一般的，N=4，W_n 取 1/N。

这种基于 N-Gram 共现的统计方法中，一元词的共现代表了翻译的忠实度，它表征了原文里面有多少单词被翻译了过来；而二元以上的共现词汇代表了目标语言的流利程度，阶数高的 N 元词的匹配度越高，系统译文的可读性就越好。

- NIST

NIST 是在 BLEU 基础上提出的一个改进方案，它采用各阶 N-Gram 的算术平均而不是几何平均，它的平分公式为：

$$score = \sum_{n=1}^{N} \{ \sum_{\substack{all\ w_1 \ldots w_n \\ that\ co-occur}} Info(w_1 \ldots w_n) / \sum_{\substack{all\ w_1 \ldots w_n \\ in\ sys\ output}} (1) \} * \exp\{\beta \log^2[in(\frac{Lsys}{Lref}, 1)]\} \quad \text{（公式 23）}$$

$$Info(w_1 \ldots w_n) = \log_2(\frac{number\ of\ occurrences\ of\ w_1 \ldots w_{n-1}}{number\ of\ occurrences\ of\ w_1 \ldots w_n}) \quad \text{（公式 24）}$$

可以看出 NIST 最终得分分为两个部分，一部分是各元 n-gram 的平均信息量的总和；

另一部分是长度罚分比。其中，β是一个常数，是一个经验阈值，使得在 $Lsys/\bar{L}_{ref}$ =2/3 时，β使得长度罚分率为 0.5，$\bar{L}_{ref}$ 是参考答案的平均长度。

- GTM（General Text Matcher）

GTM 是基于调和平均值的文本相似度方法，计算公式如下：

$$score = \frac{2 * Precision * Recall}{Precision + Recall} \quad \text{（公式 25）}$$

$$Precision = MMS / Lsys, \quad Recall = MMS / \bar{L}ref$$

其中 *MMS* 为最大匹配长度。GTM 得分范围在 0~1 之间，分数越高越好。详情请参考：http://nlp.cs.nyu.edu/GTM/。

- mWER（multi-reference Word Error Rate）

mWER 是基于编辑距离的最小单词错误率方法。得分范围在 0~1 之间，分数越低越好。所谓编辑距离，就是从系统输出译文到参考译文所需要进行的插入、删除和替换操作的最小代价。传统的操作有：插入、删除和替换，每个操作的代价为 1，这样计算出来的距离为 Levenshtein 距离（Levenshtein 1966）。WER（Word Error Rate）即是编辑距离除以参考译文中的单词数。一般的，一个句子的参考译文有多个，我们取 WER 最小的作为最终的分数，即 mWER。

- mPER（multi-reference Position-independent word Error Rate）

WER 的一个缺点是进行插入、删除等操作后，系统输出的译文与参考译文的词语顺序必须严格一致，然而，对于翻译来说，有多种翻译方法，并不一定对词语顺序有严格要求，仅用 WER 来衡量就有失偏颇。为了克服这一问题，引入了与词语的位置无关的 mWER 方法——mPER，它也是得分越低越好。计算方法和 mWER 类似，但是并不考虑顺序。

4.2 使用说明

命令行参数：

mteval.exe –c –s source_file –r reference_file –t result_file

参数说明：

-c：表示大小写敏感

-s：源语言文件

-r：参考译文

-t：系统生成的译文。

5 参考文献

Peter. F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Robert L. Mercer, The Mathematics of Statistical Machine Translation: Parameter Estimation, *Computational Linguistics*, Vol 19, No.2 ,1993

Stolcke,2002, SRILM -- An Extensible Language Modeling Toolkit. *Proc. Intl. Conf. on Spoken Language Processing*, vol. 2, pp. 901-904, Denver,2002

Franz Josef Och, Hermann Ney 2002. "Discriminative Training and Maximum Entropy Models for Statistical Machine Translation". In "*ACL 2002: Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*" pp. 295-302, Philadelphia, PA, July 2002.

Philipp Koehn, Franz Josef Och, and Daniel Marcu 2003. Statistical Phrase-Based Translation, *HLT/NAACL* 2003

R. Zens, F.J. Och, H. Ney, 2002. "Phrase-Based Statistical Machine Translation". In: M. Jarke, J. Koehler, G. Lakemeyer (Eds.) : *KI - 2002: Advances in artificial intelligence. 25. Annual German Conference on AI, KI 2002*, Vol. LNAI 2479, pp. 18-32, Springer Verlag, September 2002

Franz Josef Och，2003. "Minimum Error Rate Training for Statistical Machine Translation". In "*ACL 2003: Proc. of the 41st Annual Meeting of the Association for Computational Linguistics*", Japan, Sapporo, July 2003

Franz Josef Och, Nicola Ueffing, Hermann Ney, 2001. "An Efficient A* Search Algorithm for Statistical Machine Translation". In: "*Data-Driven Machine Translation Workshop*", pp. 55-62, Toulouse, France, July 2001.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the Sixth Conference of the Association for Machine Translation in the Americas*, pp. 115-124.

K. Papineni, S. Roukus, T. Ward and W. Zhu. 2001, BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computing Linguistics (ACL)*: 311-318. Philadelphia.

V. Levenshtein. *Binary codes capable of correcting deletions, insertions and reversals*. Soviet Physics Doklady, 10(8), pp. 707-710, February