1．Red主要思想：在参考译文依存树中抽取headword chain和fixed/floating结构，再使用这些结构与机器译文进行匹配，匹配算法是与距离相关的。其中还加入了stem，synonym，paraphrase和function word。

2．Red代码使用：

代码分为两部分，第一部分生成机器译文与参考译文的匹配结果，第二部分使用匹配结果给每个句子打出一个分数，并且最终与人工评价结果对比，输出相关性。

第一部分代码red.match

第二部分代码red.score

**第一部分输入**：./redMatch -Train –Entropy file –Function file –Para file –Align file –HWC –FF –Fmean –r file –h file –D len –refnum 1

-Train 表示输入匹配结果

-Entropy 使用熵

-Function 功能词文件

-Para 只用paraphrase时，METEOR生成的匹配结果，转成需要的格式

-Align 使用exat,stem,synonym时，METEOR生成的匹配结果，转成需要的格式

-HWC 使用headword chain

-FF 使用fixed/floating结构

-Fmean 使用F值

-r reference的依存树文件

-h 机器译文文件

-D headword chain 和fixed/floating的长度，一般为3

-refnum 参考译文的个数，wmt语料中都为1

其中，-Entropy –Para –Align 三个输入文件格式相同 pr-numr-ph-numh-func

pr表示reference中的词的位置（从0开始）

numr表示reference中在Pr开始的词的个数

ph表示hypothesis中词的位置（从0开始）

numh表示hypothesis中在Ph开始的词的个数

func表示这个词是否为功能词，1表示功能词，0不是功能词。

比如2-1-3-1-0表示reference中第1个词与hypothesis中第3个词匹配，这个词不能功能词。–Para中的功能词没有使用，只是为格式统一。

另外， BP效果不明显，可以不用。

**第一部分输出文件的格式**：



Exact:0 stem:1 synoym:2 paraphrase:3

Chunk--匹配上1gram的个数--chunk的个数--每个chunk中word的个数

hyp总的词数—功能词的个数—非功能词的个数

ref总的词数--功能词的个数—非功能词的个数

hyp1gram--funWord1\_matchModule--funWord2\_matchModule...... ref1gram—funWord1\_matchModule—funWord2\_matchModule...... papraphrase-Word1Func\_word2Func--3

hwc--funWord\_matchModule-…-距离的差。。。。。。

ref2gram的总的个数-ref3gram总的个数-ref4gram总的个数，不包含ref1gram。Ref2gram个数可能超过ref长度，因为还包含floating结构。

第二部分输入: ./redScore –t filename –D 3 –tag tag –wmatch weight1 –wngram weight1 –params paramValue –wentry 1

-t filename 各个系统的机器翻译结果(第一部分输出)

-D ngram长度，设为3

-tag tag 用于区分wmt语料的年份，由于用的wmt语料，每年文件名格式不太一致，做了下区分

-wmatch weight 四种匹配模式的权重

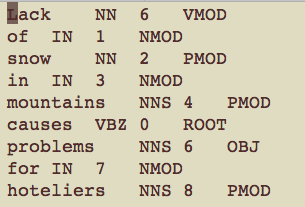
-wngram weight 三种不同长度ngram的权重

-params paramValue， Fmean和function word的参数值

-wentry 1 由于目前没有用到，固定为1.

注意：

1、依存树的格式：  
word1 词性 第i个词(为word1的一个子节点) relation(word1与第i个词的依存关系)

  
 只用到了第一部分word1和第三部分第i个词，即第几个词依存于哪个词。

2、代码根据wmt语料文件名的格式写的，使用时需要把待评价文件的文件名改为与wmt数据类似的，比如newstest2015.UU-unconstrained.3977.fi-en（年份.系统名.语言对）。或者也可以把代码中读取文件的部分稍微改一下。