Refactored Moses

Hieu Hoang
MT Marathon
Prague
August 2013
What did you Refactor?

• Decoder
  – Feature Function Framework
    • moses.ini format
  – Simplify class structure
  – Deleted functionality
  – NOT decoding algorithms
• Tuning
  – NOT tuning algorithm
• EMS (Experiment Management System)
Why did you Refactor?

• Feature Function Framework
  – easier to implement new features
  – use sparse features

• Simplify class structure
  – easier to develop with Moses

• Delete functionality
  – easier to refactor code
  – very little deletion
Why did you Refactor?

• Feature Function Framework
  – easier to implement new features
  – use sparse features

• **Simplify class structure**
  – easier to develop with Moses

• Delete functionality
  – easier to refactor code
  – very little deletion
Why did you Refactor?

• Feature Function Framework
  – easier to implement new features
  – use sparse features

• Simplify class structure
  – easier to develop with Moses

• Delete functionality
  – easier to refactor code
  – very little deletion
Language Model

IN THE BEGINNING

LanguageModel

LanguageModelSingleFactor

SRI

IRST

KEN

LanguageModelMultiFactor
Language Model

THEN....

LM
  LMRefCount
  KEN

LMImplementation
  LMSingleFactor
  LMMultiFactor
  LMPointerState
    SRI
    IRST
Language Model

NOW

LM

LMIImplementation

LMSingleFactor

SRI

IRST

LMMultiFactor

KEN
Phrase Tables

IN THE BEGINNING

PhraseDictionary

- Memory
- TreeAdaptor
- SCFG
- OnDisk
Phrase Tables
THEN....

PhraseDictionary

Memory
TreeAdaptor
RuleTableTrie
OnDisk
Compact

SCFG
RuleTableUTrie

PDFeature
Phrase Tables

NOW

PhraseDictionary

- TreeAdaptor
- RuleTableTrie
- Memory
- RuleTableUTrie
- OnDisk
- Compact
Why did you Refactor?

• Feature Function Framework
  – easier to implement new features
  – use sparse features

• Simplify class structure
  – easier to develop with Moses

• Delete functionality
  – easier to refactor code
  – very little deletion
1. Translation Systems
   – multiple engines in 1 decoding process
   – replaced with alternative weights function
2. Distributed Language Model
   – send LM queries to different machines
   – replace with multipass/asynchronous decoding?
3. Continue Partial Translation
   – start from non-empty hypothesis
Why did you Refactor?

• **Feature Function Framework**
  – easier to implement new features
  – use sparse features

• **Simplify class structure**
  – easier to develop with Moses

• **Delete functionality**
  – easier to refactor code
  – very little deletion
Adding new Feature Function

THEN....

• Add entry to
  – Parameter class
• Read parameter in StaticData
  – add variable to hold new feature
  – load feature

ini file:
[new-feature-section]
0 0 1-1 file.name

[weight-new-feature]
Adding new Feature Function

Now....

• StaticData
  – Simple framework to load feature

ini file:

[feature]
WordPenalty
Distortion
...
NEW-FEATURE file=path factor=0

[weight]
WordPenalty0= -0.27
Distortion0= 0.14
...
NEW-FEATURE0= 0.54
• Only know about certain feature functions
  
  – Hardcoded array

  my @ABBR_FULL_MAP = qw(d=weight-d lm=weight-l tm=weight-t w=weight-w
g=weight-generation lex=weight-lex l=weight-i dlm=weight-dlm pp=weight-pp wt=weight-wt pb=weight-pb lex=weight-lex glm=weight(glm);

  - requires feature name and abbreviation
• Only know about certain feature functions
  – Hardcoded array
    ```perl
    my @ABBR_FULL_MAP = qw(d=weight-d lm=weight-l tm=weight-t w=weight-w
g=weight-generation lex=weight-lex l=weight-l dlm=weight-dlm pp=weight-pp
wt=weight-wt pb=weight-pb lex=weight-lex glm=weight(glm));
    ```
  – requires feature name and abbreviation

• Deleted array
  – Ask decoder for feature function
  – abbreviations deleted
Timeline of a Translation Rule

- File
- Load
- Memory
- Apply to input sentence
- Translation Option
- Search
- Hypothesis
Timeline of a Translation Rule

File

Load
Source phrase
Target phrase

Memory

Apply to input
sentence
Input sentence
Input path

Translation Option

Hypothesis

Search
Translation context
Segmentation

和
peace
Timeline of a Translation Rule

1. **File**
   - Load
   - Once

2. **Memory**
   - Apply to input sentence
     - Per occurrence in sentence

3. **Translation Option**

4. **Hypothesis**
   - Search
     - Per hypothesis
Feature Function API

Loading

$X \rightarrow \text{je suis } X_1 \text{ ||| I am } X_1$

Access to:
- Source phrase: je suis $X_1$
- Target phrase: I am $X_1$

void Evaluate(source, target, scores, estimated future scores)

Feature functions that use this:
- Word Penalty
- Phrase penalty
- Language model (partial)
Feature Function API
Apply to input sentence

Input:  je|PRO suis|VB 25|NUM ans|NNS .|

Access to:
Input subphrase:  je|PRO suis|VB 25|NUM
Input scores: 0.3 0.2 0.1

void Evaluate(input,
   input path,
scores)

Feature functions that uses this:
Input feature
Bag-of-word features....
In this chapter, we focus on the analysis of the search space phrase-based, syntactic and the relaxed syntax model of Chapter 4. We seek to explain why phrase-based model often outperform the hierarchical model, especially for closely related languages, why the pure syntax model of the previous chapter perform so poorly, and the reason for the good performance of the relaxed syntax model. By understanding interaction of the variables and heuristics that controls the translation search space, we are able to adjust the relaxed syntax model to outperform results in the previous chapter.

Statistical machine translation (SMT) can be viewed as a search process with an embedded binary classifier. Datapoints in the discriminative space consist of a tuple of a source sentence and a target sentence which the classifier labels as positive or negative depending on whether it believes the tuple represents a good or bad translation. Multiple tuples with the same source but different target strings can be positively labelled, and vice versa.

It is more usual that the classifier perform its discriminative function by being given a source and being asked to enumerate positively labelled tuples. The set of all possible enumerated tuples defines the search space for the search component. The parameters...
Feature Function API

Decoding

Feature functions that uses this:

• All stateful features
  – Language models
  – Distortion model
  – Lexicalized distortion
  – ...

• Some stateless features
  – Global lexicalized model
  – Word translation
Feature Function

**Loading:**

```java
void Evaluate(source, target, scores, estimated future scores)
```

**Apply to Input:**

```java
void Evaluate(input, input path, scores)
```

**Search:**

**Stateless features:**

```java
void Evaluate(hypo, scores)
void EvaluateChart(hypo, scores)
```

**Stateful features:**

```java
state Evaluate(hypo, previous state, scores)
state EvaluateChart(hypo previous state, scores)
```
Strange Features functions (1)

• Language model
  – implement 2 Evaluate()

1. Loading
  • evaluate full n-grams
    • reprise de la session ||| resumption of the session
  • estimate future cost
    – partial n-grams

2. Decoding
  • evaluate overlapping n-grams
Strange Feature Functions (2)

- Phrase-tables
- Unknown Word Penalty
- Generation Model
  - integral part of decoding process
  - Uses no Evaluate()
    - scores assign by decoder
Register a Feature Function

• Register
  – in moses/FF/Factory.cpp
  – add entry
    • MOSES_FNAME(ClassName);
Language Model

• Inherit from
  – LanguageModelSingleFactor

```cpp
class LanguageModelSingleFactor : ...
{
    LMResult GetValue(
        const std::vector<const Word*> &contextFactor,
        State* finalState = NULL) const = 0;
}
```
Phrase Table

• Inherit from
  – PhraseDictionary

• Legacy API:
  
  ```cpp
class PhraseDictionary : ...
{
  public:
    TargetPhraseCollection *GetTargetPhraseCollectionLEGACY(
      Phrase &src) const;

  protected:
    const TargetPhraseCollection *GetTargetPhraseCollectionNonCacheLEGACY(Phrase &src);
}
```
Phrase Table

• New API:

```cpp
class PhraseDictionary {
public:
    void GetTargetPhraseCollectionBatch(InputPathList &);
};
```

• InputPathList

Input Sentence: je suis 25 ans.

Input Path List: 
```
je
je suis
je suis 25
...
suis
suis 25
...
```
Phrase-table Optimization

Input Sentence: je suis 25 ans.
Input Path List: je
je suis
je suis 25
... 
suis
suis 25
...

Phrase-table trie

0

1
suis

2

3

4

25
Input Path List:

- einen
- einen wettbewerb
- einen wettbewerbs
- einen wettbewerbsbedingten
- einen wettbewerb bedingten
- einen wettbewerbs bedingten
- einen wettbewerb bedingten preissturz
- einen wettbewerbs bedingten preis
- einen wettbewerbsbedingten preissturz
- einen wettbewerbsbedingten preis
- ....
Phrase Table
Syntax Decoding

• New API:

    class PhraseDictionary : . . .
    {
        virtual ChartRuleLookupManager *CreateRuleLookupManager(
            const ChartParser &,
            const ChartCellCollectionBase &) = 0;
    }

    – Active Parsing
    – CKY+

    class ChartRuleLookupManager
    {
        void GetChartRuleCollection(
            const WordsRange &range,
            ChartParserCallback &outColl) = 0;
    }
Results

• Lattice decoding
  – use all phrase-table implementations
  – Syntax model (nearly...)

• One in-memory phrase-table implementation

• One on-disk phrase-table
  – Syntax models
  – AND phrase-based model
  – Zen’s binary implementation
    • phrase-based only
    • deprecated – to be removed
Results

Translation Quality

- Pass regression tests
  - tests have also changed...
- BLEU unchanged
  - subject to random variability

<table>
<thead>
<tr>
<th>Language</th>
<th>#Description</th>
<th>v 1.0</th>
<th>29th July</th>
</tr>
</thead>
<tbody>
<tr>
<td>En-es</td>
<td>1 pb</td>
<td>24.81</td>
<td>24.59</td>
</tr>
<tr>
<td></td>
<td>2 hiero</td>
<td>24.2</td>
<td>24.2</td>
</tr>
<tr>
<td></td>
<td>3(1) + CreateOnDiskPt</td>
<td>24.58</td>
<td></td>
</tr>
<tr>
<td>Es-en</td>
<td>1 pb</td>
<td>23.01</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>2 hiero</td>
<td>22.37</td>
<td>22.32</td>
</tr>
<tr>
<td></td>
<td>3(1) + CreateOnDiskPt</td>
<td>23.03</td>
<td></td>
</tr>
<tr>
<td>En-cs</td>
<td>1 pb</td>
<td>11.04</td>
<td>11.05</td>
</tr>
<tr>
<td></td>
<td>2 hiero</td>
<td>10.93</td>
<td>10.9</td>
</tr>
<tr>
<td></td>
<td>3(1) + placeholders</td>
<td>9.23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4(1) + CreateOnDiskPt</td>
<td>11.19</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5(2) + Ken's incre algo</td>
<td>10.84</td>
<td></td>
</tr>
<tr>
<td>Cs-en</td>
<td>1 pb</td>
<td>15.72</td>
<td>15.8</td>
</tr>
<tr>
<td></td>
<td>2 hiero</td>
<td>15.68</td>
<td>15.66</td>
</tr>
<tr>
<td></td>
<td>3(1) + placeholders</td>
<td>14.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4(1) + CreateOnDiskPt</td>
<td>11.66</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5(2) + Ken's incre algo</td>
<td>15.44</td>
<td></td>
</tr>
<tr>
<td>En-de</td>
<td>1 pb</td>
<td>11.87</td>
<td>11.7</td>
</tr>
<tr>
<td></td>
<td>2 hiero</td>
<td>11.62</td>
<td>11.58</td>
</tr>
<tr>
<td></td>
<td>3(1) + POS de</td>
<td>11.67</td>
<td>11.65</td>
</tr>
<tr>
<td></td>
<td>4(3) + POS en</td>
<td>11.75</td>
<td>11.65</td>
</tr>
<tr>
<td></td>
<td>5(1) + CreateOnDiskPt</td>
<td>15.75</td>
<td>11.62</td>
</tr>
<tr>
<td>De-en</td>
<td>1 pb</td>
<td>15.75</td>
<td>15.81</td>
</tr>
<tr>
<td></td>
<td>2 hiero</td>
<td>15.53</td>
<td>15.71</td>
</tr>
<tr>
<td></td>
<td>3(1) + POS de</td>
<td>15.84</td>
<td>15.9</td>
</tr>
<tr>
<td></td>
<td>4(3) + POS en</td>
<td>15.93</td>
<td>15.78</td>
</tr>
<tr>
<td></td>
<td>5(1) + CreateOnDiskPt</td>
<td>15.91</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6(3) + CreateOnDiskPt</td>
<td>15.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7(4) + CreateOnDiskPt</td>
<td>15.78</td>
<td></td>
</tr>
<tr>
<td>En-fr</td>
<td>1 pb truecase</td>
<td>24.38</td>
<td>24.38</td>
</tr>
<tr>
<td></td>
<td>2 pb recase</td>
<td>22.94</td>
<td>22.81</td>
</tr>
<tr>
<td></td>
<td>3(2) + hiero</td>
<td>22.28</td>
<td>22.36</td>
</tr>
<tr>
<td></td>
<td>4(2) + POS en</td>
<td>23.05</td>
<td>23.07</td>
</tr>
<tr>
<td></td>
<td>5(2) + kbmira</td>
<td>22.85</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6(2) + pro</td>
<td>22.64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7(2) + CreateOnDiskPt</td>
<td>22.65</td>
<td></td>
</tr>
<tr>
<td>Fr-en</td>
<td>1 pb truecase</td>
<td>24.06</td>
<td>23.97</td>
</tr>
<tr>
<td></td>
<td>2 pb recased</td>
<td>22.41</td>
<td>22.43</td>
</tr>
<tr>
<td></td>
<td>3(2) + hiero</td>
<td>18.05</td>
<td>17.75</td>
</tr>
<tr>
<td></td>
<td>4(2) + factors</td>
<td>22.55</td>
<td>22.47</td>
</tr>
<tr>
<td></td>
<td>5(2) + kbmira</td>
<td>22.44</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6(2) + pro</td>
<td>22.49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7(2) + CreateOnDiskPt</td>
<td>22.46</td>
<td></td>
</tr>
</tbody>
</table>
Results

Phrase-based

<table>
<thead>
<tr>
<th></th>
<th>speed (sec)</th>
<th>speed (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 v 1.0</td>
<td>784</td>
<td>857.12</td>
</tr>
<tr>
<td>2 master + caching</td>
<td>1,154</td>
<td>1317.07</td>
</tr>
</tbody>
</table>

-54%

With compact pt:

<table>
<thead>
<tr>
<th></th>
<th>speed (sec)</th>
<th>reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 v1</td>
<td>840.57</td>
<td>2%</td>
</tr>
<tr>
<td>4 master (with caching)</td>
<td>934.82</td>
<td>-9%</td>
</tr>
</tbody>
</table>

Hierarchical

<table>
<thead>
<tr>
<th>Pop Limit</th>
<th>100</th>
<th>200</th>
<th>500</th>
<th>1000 Loading</th>
</tr>
</thead>
<tbody>
<tr>
<td>v 1.0</td>
<td>1,044</td>
<td>1,393</td>
<td>2,620</td>
<td>4,546</td>
</tr>
<tr>
<td>13th Aug</td>
<td>670</td>
<td>1034</td>
<td>2064</td>
<td>3743</td>
</tr>
<tr>
<td>Reduction (excl load)</td>
<td>-65%</td>
<td>-26%</td>
<td>-21%</td>
<td>-18%</td>
</tr>
</tbody>
</table>
Comparison with the decoders