Feature Functions

Hieu Hoang
Matthias Huck

December 2014

Feature Function

- Examples
  - Phrase-table
  - Language model
  - Word penalty
  - Phrase penalty
- Gives a score to a translation
  - Partial translation
  - Completed translation
- Many feature functions
  - Weighted linear combination
- What is a translation?
  - Made of multiple ***translation rules***

Timeline of a Translation Rule

File

Load

Memory

Apply to input sentence

Translation Option

Search

Hypothesis

MOSES CORE

Timeline of a Translation Rule

File

Load

Source phrase
Target phrase

Memory

Apply to input sentence

Input sentence
Input path

Translation Option

Search

Translation context
Segmentation

Hypothesis

MOSES CORE

Timeline of a Translation Rule

File

Load

Once

Memory

Apply to input sentence

Per occurrence in sentence

Translation Option

Search

Per hypothesis

Hypothesis

MOSES CORE

Feature Function API

Loading

File

Memory

je suis ||| I am

Access to: Source phrase: je suis

Target phrase: I am

```
void EvaluateInIsolation(
          source,
          target,
          scores,
          estimated future scores)
```

Feature functions that use this:

- Word Penalty
- Phrase penalty
- Language model (partial)

MOSES CORE

THE UNIVERSITY OF EDINBURGH

Feature Function API

Apply to input sentence

Memory

Access to: Input sentence: je suis 25 ans .

Input subphrase: je suis 25

```
void EvaluateWithSourceContext(
        input,
       input path,
       target,
        scores,
        estimated future scores)
```

Feature functions that uses this:

- Input feature
- Bag-of-word features....

Translation Option

MOSES CORE

THE UNIVERSITY OF EDINBURGH

Feature Function API
Search

Translation Option

Hypothesis

Access to: Current rule (hypothesis)
Previous rules
Segmentation

Stateful features:

```
state EvaluateWhenApplied(hypo,
        previous state,
        scores)
state EvaluateWhenApplied(hypo
        previous state,
        scores)
```

Stateless features:

```
void EvaluateWhenApplied(hypo, scores)
void EvaluateWhenApplied(hypo, scores)
```

MOSES CORE

THE UNIVERSITY OF EDINBURGH

Feature Function API

Decoding

Translation Option

Hypothesis

Feature functions that uses this:

- All stateful features
  - Language models
  - Distortion model
  - Lexicalized distortion
  - …

MOSES CORE

THE UNIVERSITY OF EDINBURGH

Feature Function

Loading:

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

Apply to Input:

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

Search:

Stateful features:

```
state Evaluate(hypo,
         previous state,
         scores)
state EvaluateChart(hypo
           previous state,
           scores)
```

Stateless features:

```
void Evaluate(hypo, scores)
void EvaluateChart(hypo, 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. Search
     - evaluate overlapping n-grams

MOSES CORE

THE UNIVERSITY OF EDINBURGH

Strange Feature Functions (2)

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

Adding a New Feature Function

- Inherit from
  - StatefulFeatureFunction
  - StatelessFeatureFunction
- Register
  - in moses/FF/Factory.cpp
  - add entry
    - MOSES_FNAME(ClassName);

Adding a New Feature Function

- Inherit from
  - StatefulFeatureFunction
  - StatelessFeatureFunction
- Register
  - in moses/FF/Factory.cpp
  - add entry
    - MOSES_FNAME(ClassName);

Extensions

- Change input
  - Add/delete word
  - Integrate parser/tagger
- Prune
  - Hard constraint
  - Negative infinity score

Properties

- Vanilla translation rule

  je suis ||| I am ||| 0.1 0.2 0.3 0.4

- With properties

  je suis ||| I am ||| 0.1 0.2 0.3 0.4||| ||| |||
  ||| {{Key1 Value1}} {{Key2 Value2}}

- Example properties
  - Syntax structure
  - Document context
  - Orientation

MOSES CORE

THE UNIVERSITY OF EDINBURGH

Properties

- Implement class
  - Inherit from PhraseProperty
  - Override method
    - ProcessValue(string)
  - Register property
    - MOSES_PNAME2("KeyName", Property class);
- Use property
  - Inside feature function Evaluate()
  - targetPhrase.GetProperty("KeyName");

Feature Functions

Hieu Hoang
Matthias Huck

December 2014

Thanks for inviting me to come

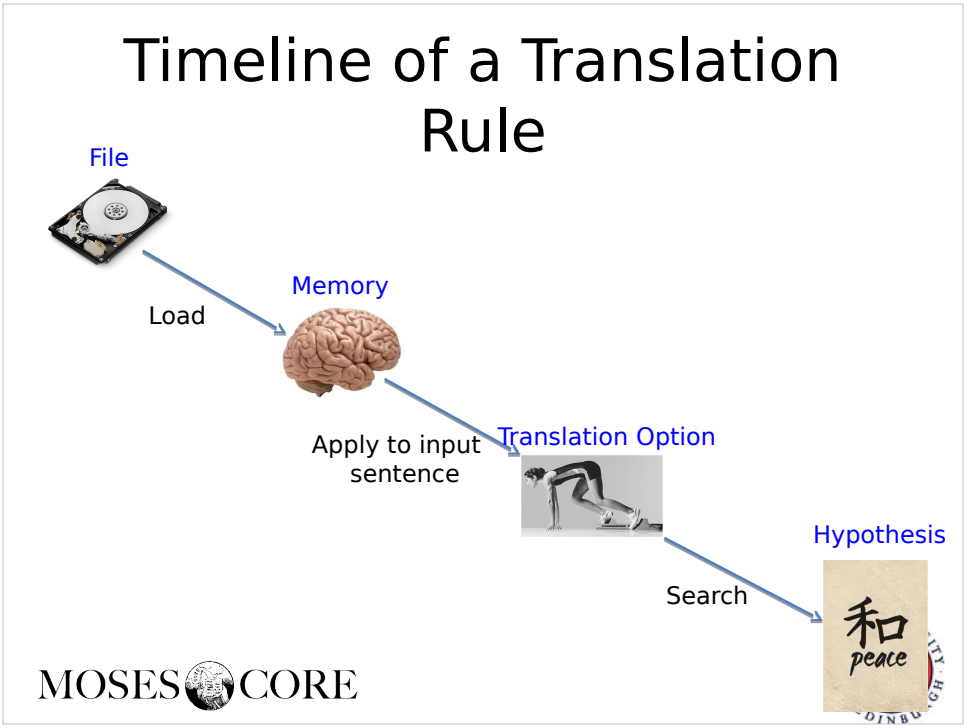
Here to tell you a little about the things I've been doing to Moses
- over the past 2 years
- mainly concentrate of the past year
  - but will quickly tell you about things I did prior to that

Feature Function

- Examples
  - Phrase-table
  - Language model
  - Word penalty
  - Phrase penalty
- Gives a score to a translation
  - Partial translation
  - Completed translation
- Many feature functions
  - Weighted linear combination
- What is a translation?
  - Made of multiple ***translation rules***

MOSES CORE

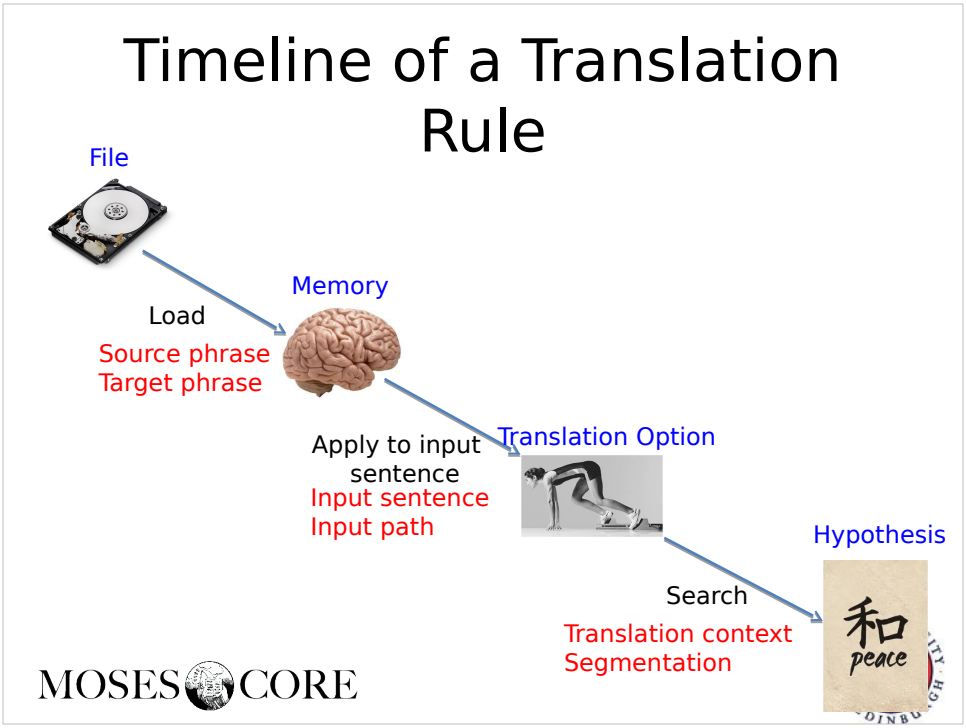
What is this feature function framework?



What is the task of a feature function
- it's task is to give a score to a translation rule

Translation rule has a lifespan
- starts off in a file on disk
- gets loaded into memory

- before a sentence is decoded
  - translation rules are looked up
    - fitted to a specific place in the source sentence
  - name of translation rules
    - changes to translation option
    - all intents and purposes
      - is a trans rule



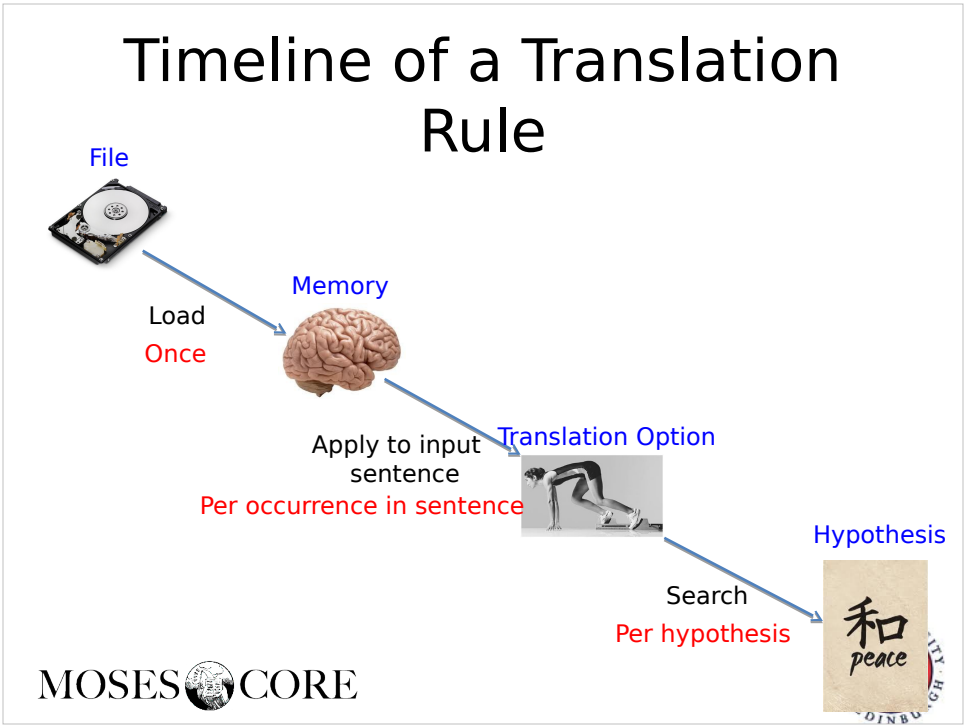
at each step
- feature function has access to different kinds of side information with which to score the rule

During loading
- only know what the rule is, without context

When it is being applied to a sentence
- it know the sentence

During search
- it know what other rules have been used

These are the information it can use to score the rule



Point of showing you this timeline

sooner you caclulate it, the better

1. efficiency
   - not repeated
2. more accurate
   - each stage – subject to pruning
     - some rules are thrown away
     - if the feature function can give a good score
     - the rule can say
       ‘hey I’ll be really useful to you, don’t throw me away!”

Feature Function API

Loading

File

Memory

je suis ||| I am

Access to: Source phrase: je suis

Target phrase: I am

```
void EvaluateInIsolation(
          source,
          target,
          scores,
          estimated future scores)
```

Feature functions that use this:

- Word Penalty
- Phrase penalty
- Language model (partial)

MOSES CORE

During load

- this is the translation rule

- If you want your FF to score the rule now
  - implement this function
  - it takes are arguments
    - source + target parts of the rule
    - you return the scores and estimated future score

Feature Function API

Apply to input sentence

Memory

Access to: Input sentence: je suis 25 ans .

Input subphrase: je suis 25

```
void EvaluateWithSourceContext(
        input,
       input path,
       target,
        scores,
        estimated future scores)
```

Feature functions that uses this:

- Input feature
- Bag-of-word features....

Translation Option

MOSES CORE

Have a sentence

- looking up rules that can be used in that sentence
- once you find a rule that can be applied
  - to a specific substring in a specific sentence
  - create translation option

At this point

- have another opportunity to evaluate the scores of the rules

Feature Function API

Search

Translation Option

Access to: Current rule (hypothesis)
Previous rules
Segmentation

Stateful features:

```
state EvaluateWhenApplied(hypo,
        previous state,
        scores)
state EvaluateWhenApplied(hypo
          previous state,
          scores)
```

Stateless features:

```
void EvaluateWhenApplied(hypo, scores)
void EvaluateWhenApplied(hypo, scores)
```

Hypothesis

MOSES CORE

search

when you have a translation rule
- you know exactly where it's going to be applied to
- and you actually apply it

Implement 1 of these functions

Only place where calculating the feature function is different for phrase-based or syntax models
- slightly different for stateless and stateful features

This function has access to the hypothesis

Feature Function API

Decoding

Translation Option

Hypothesis

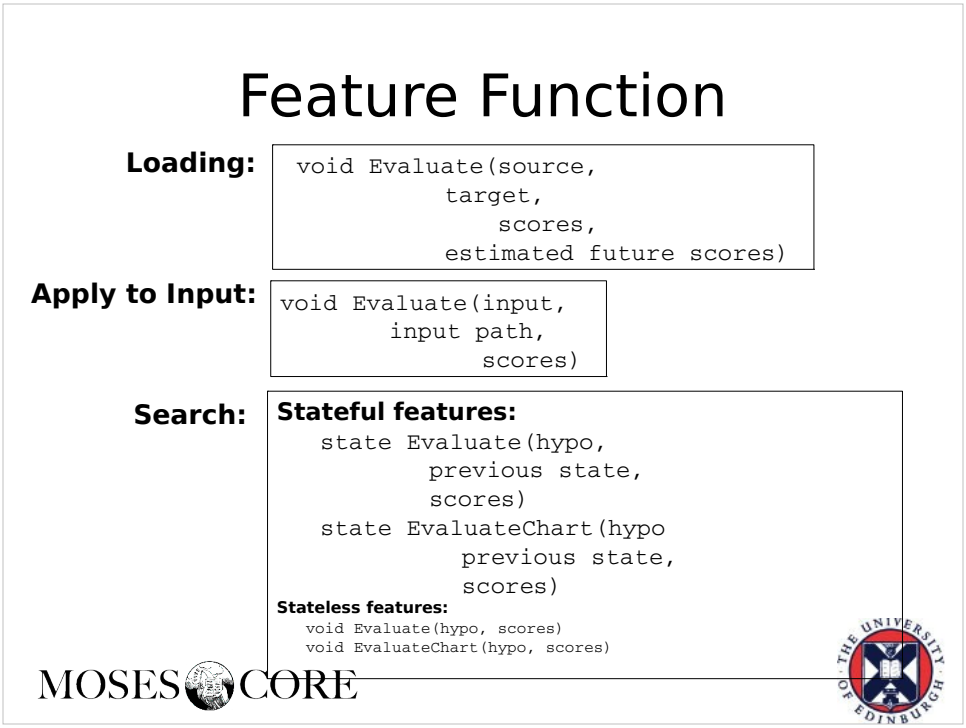
Feature functions that uses this:

- All stateful features
  - Language models
  - Distortion model
  - Lexicalized distortion
  - …

MOSES CORE

all the translation rules that were used, the total output phrase segmentation

- derivation tree if hiero/syntax model



Recap

- you can score translation rule at 3 stages in the decoding process
  - Loading
  - Appying to the input sentence
  - Search
- Implement 1 of these functions if you do

However, a FF can score the same rule in more than 1 stage

- ie. It can implement more of these functions

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. Search
     - evaluate overlapping n-grams

MOSES CORE

For those who know Moses

- this is nothing new
- this is the way Moses has always computed language model scores
  - if you had a trigram LM
  - store trigrams in the target phrase upon loading
  - store overlapping n-gram during search

- the new framework enable this optimisation to be used by every other feature function

Strange Feature Functions (2)

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

MOSES CORE

Adding a New Feature Function

- Inherit from
  - StatefulFeatureFunction
  - StatelessFeatureFunction
- Register
  - in moses/FF/Factory.cpp
  - add entry
    - MOSES_FNAME(ClassName);

MOSES CORE

What do you have to do now to add a new FF?
- only 2 things

1. Create a class that inherit from Stateful or Stateless FF
   - depending on the type of FF you want
2. Register your class

Adding a New Feature Function

- Inherit from
  - StatefulFeatureFunction
  - StatelessFeatureFunction
- Register
  - in moses/FF/Factory.cpp
  - add entry
    - MOSES_FNAME(ClassName);

MOSES CORE

What do you have to do now to add a new FF?
- only 2 things

1. Create a class that inherit from Stateful or Stateless FF
   - depending on the type of FF you want
2. Register your class

Extensions

- Change input
  - Add/delete word
  - Integrate parser/tagger
- Prune
  - Hard constraint
  - Negative infinity score

Properties

- Vanilla translation rule

  je suis ||| I am ||| 0.1 0.2 0.3 0.4

- With properties

  je suis ||| I am ||| 0.1 0.2 0.3 0.4||| ||| |||
  ||| {{Key1 Value1}} {{Key2 Value2}}

- Example properties
  - Syntax structure
  - Document context
  - Orientation

MOSES CORE

7th column

Properties

- Implement class
  - Inherit from PhraseProperty
  - Override method
    - ProcessValue(string)
  - Register property
    - MOSES_PNAME2("KeyName", Property class);
- Use property
  - Inside feature function Evaluate()
  - targetPhrase.GetProperty("KeyName");

MOSES CORE