

Feature Functions

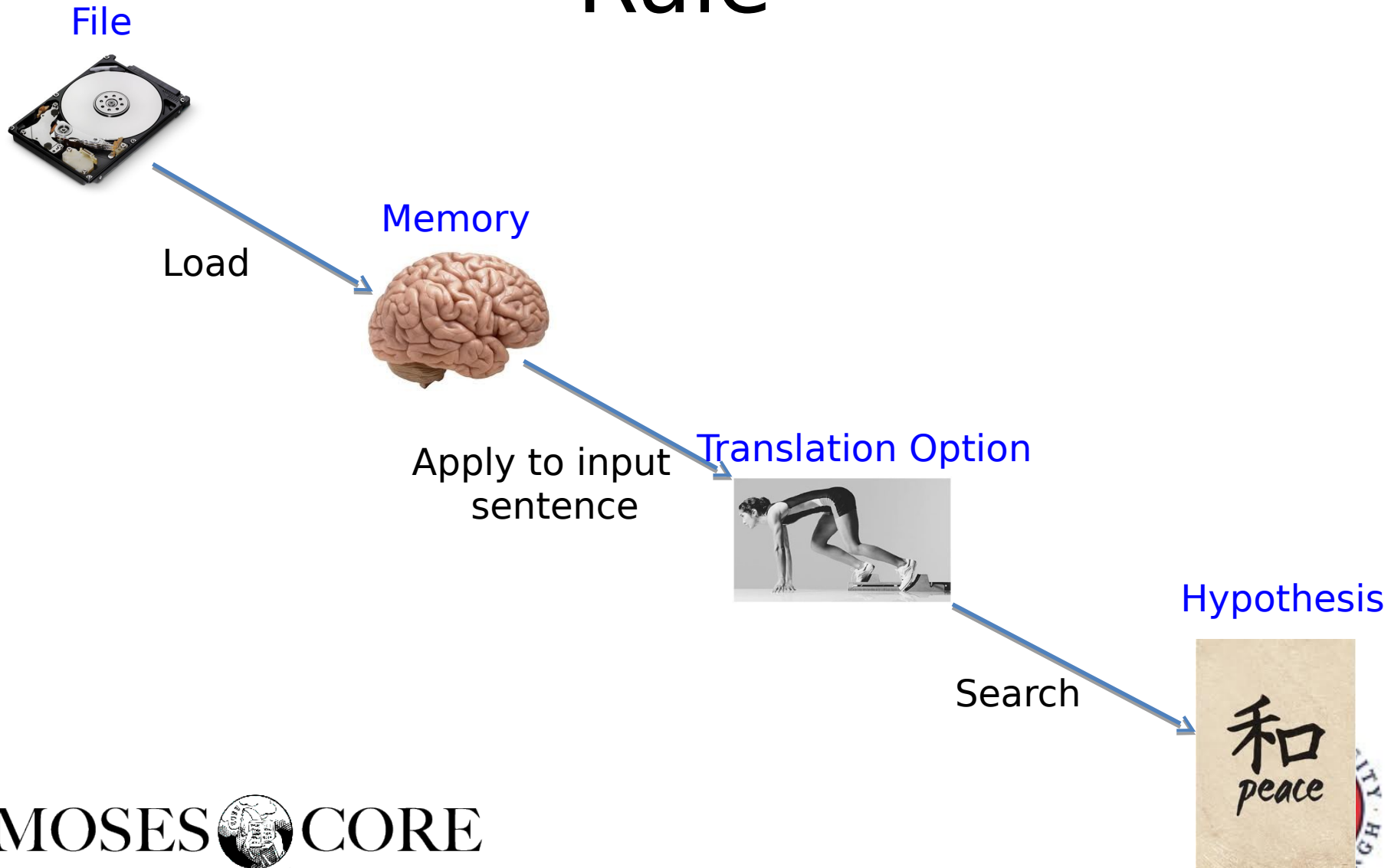
Hieu Hoang
Matthias Huck

December 2014

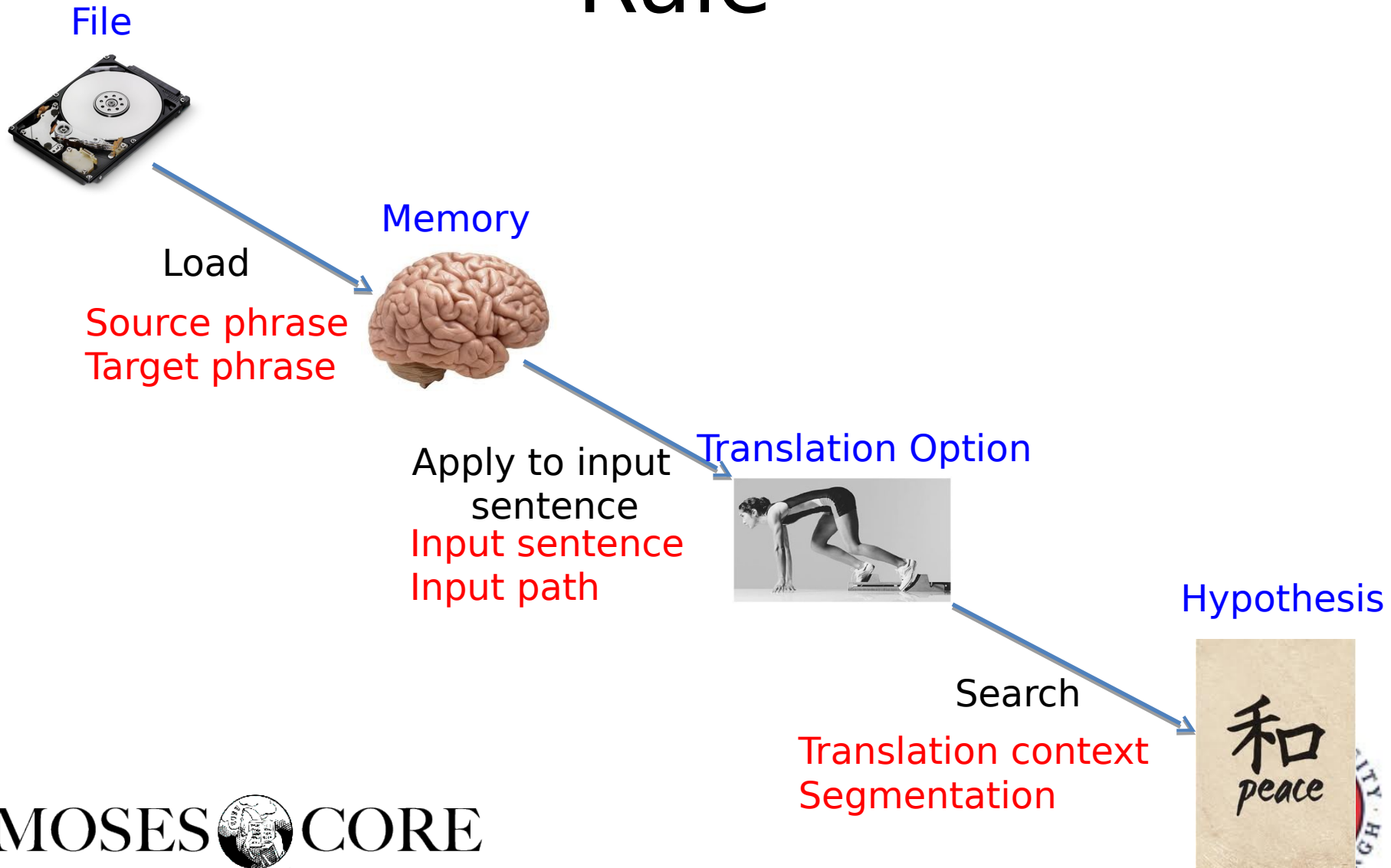
Feature Function

- Calculate score(s) for a translation rule
 - Partial translation
 - Completed translation
- Examples
 - Phrase table
 - Language model
 - Word penalty
 - Phrase penalty
- Many feature functions
 - Weighted linear combination
- What is a translation?
 - Made of multiple ***translation rules***

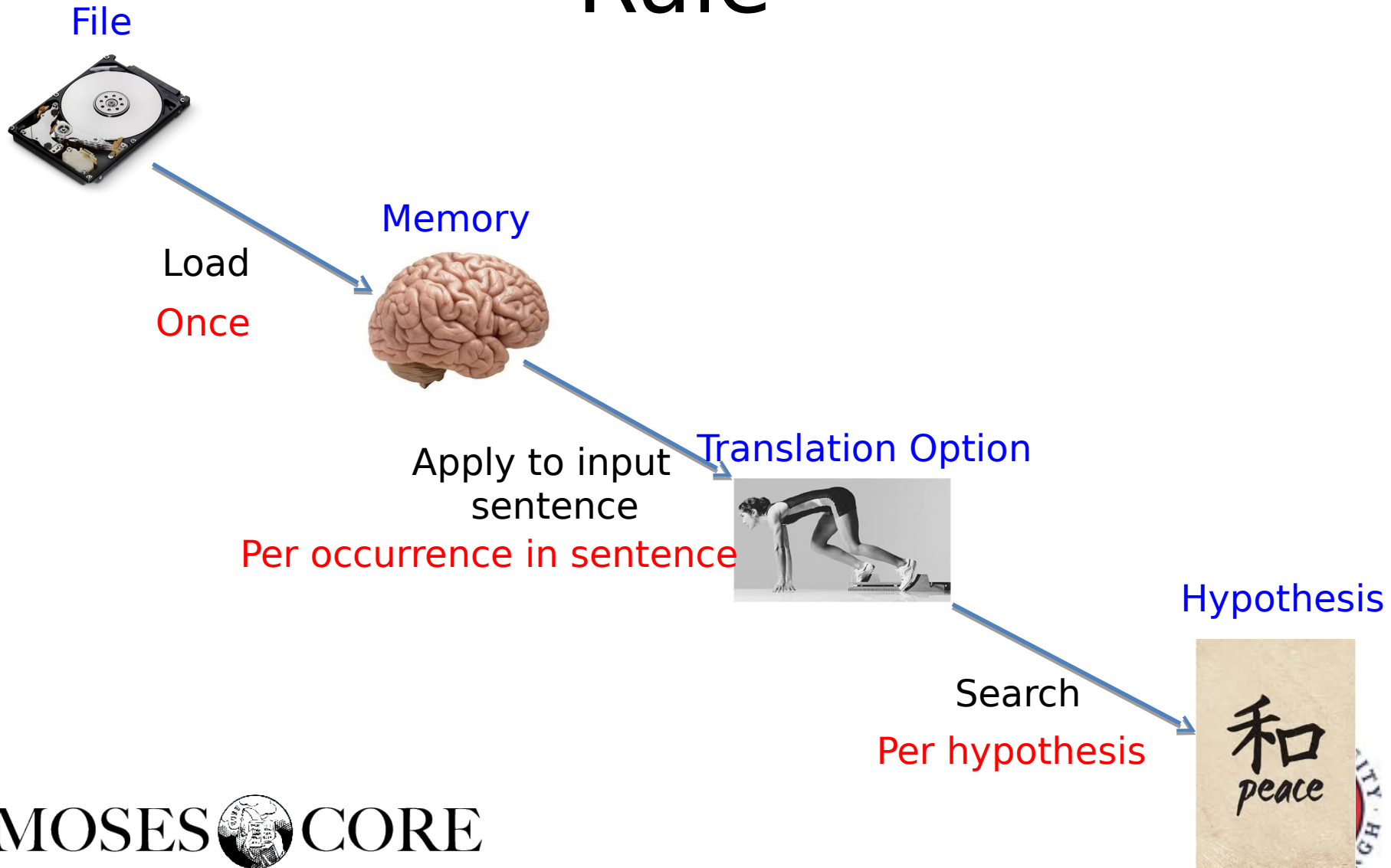
Timeline of a Translation Rule



Timeline of a Translation Rule



Timeline of a Translation Rule



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)

Feature Function API

Apply to input sentence

Memory



Translation Option



Access to: Input sentence: je suis 25 ans .
Input subphrase: je suis 25

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

Feature functions that use this:
Input feature
Bag-of-word features....

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)
```


Feature Function API

Decoding

Translation Option



Hypothesis



Feature functions that use this:

- All stateful features
 - Language models
 - Distortion model
 - Lexicalized reordering model
 - ...

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)
```

Adding a New Feature Function

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

Template Feature Functions

- SkeletonStatelessFF
- SkeletonStatefulFF
- SkeletonLM
- SkeletonPT
- Video Demo
 - on YouTube

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

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

- Calculate score(s) for a translation rule
 - Partial translation
 - Completed translation
- Examples
 - Phrase table
 - Language model
 - Word penalty
 - Phrase penalty
- Many feature functions
 - Weighted linear combination
- What is a translation?
 - Made of multiple **translation rules**

MOSES  CORE



What is the task of a feature function

- it's task is to give score to a translation rule
- the thing you see in a phrase-table
- FF calculate 1 or more scores

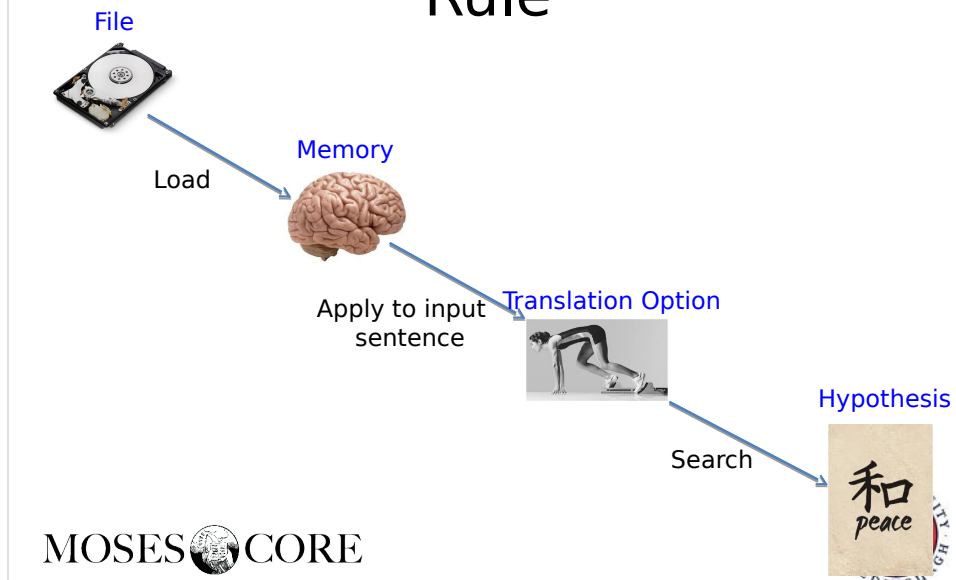
These are examples of FF

There can be many scores

Total score for the translation

- weighted sum of all scores

Timeline of a Translation Rule

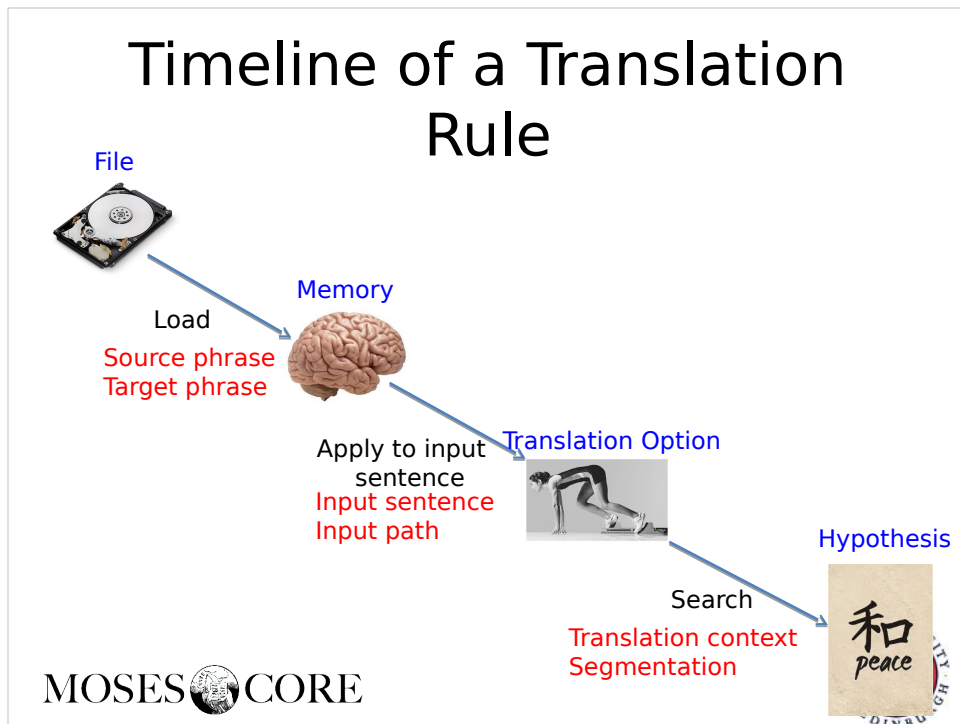


Translation rule has a lifespan

- 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

When it's being used in search

- it's name changes to hypothesis
- again, wrapper around a translation 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

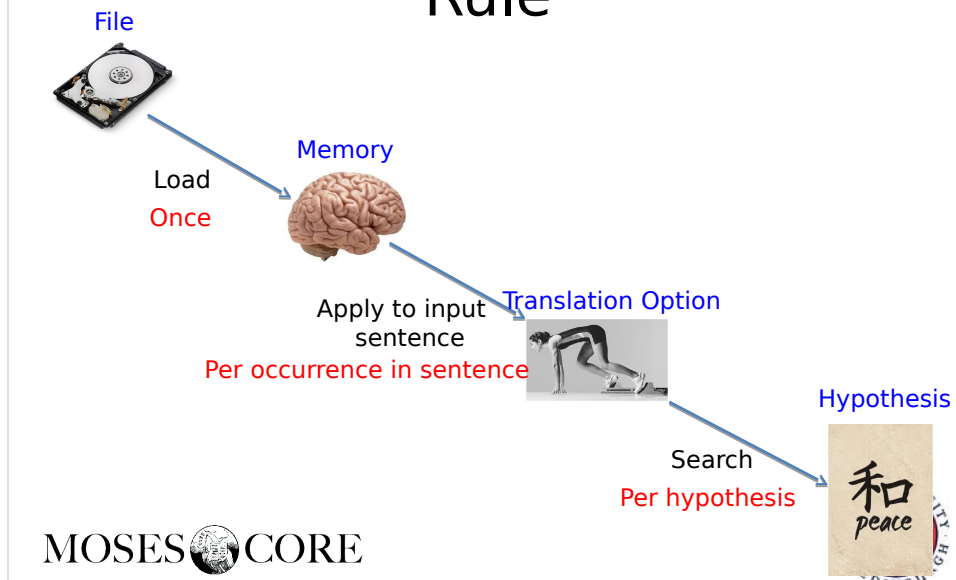
- it know the sentence

During search

- it know what other rules have been used

These are the information it can use to score

Timeline of a Translation Rule



Point of showing you this timeline

sooner you calculate 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



↓


Translation Option




Access to: Input sentence: je suis 25 ans .
Input subphrase: je suis 25

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

Feature functions that use this:
Input feature
Bag-of-word features....

MOSES  CORE

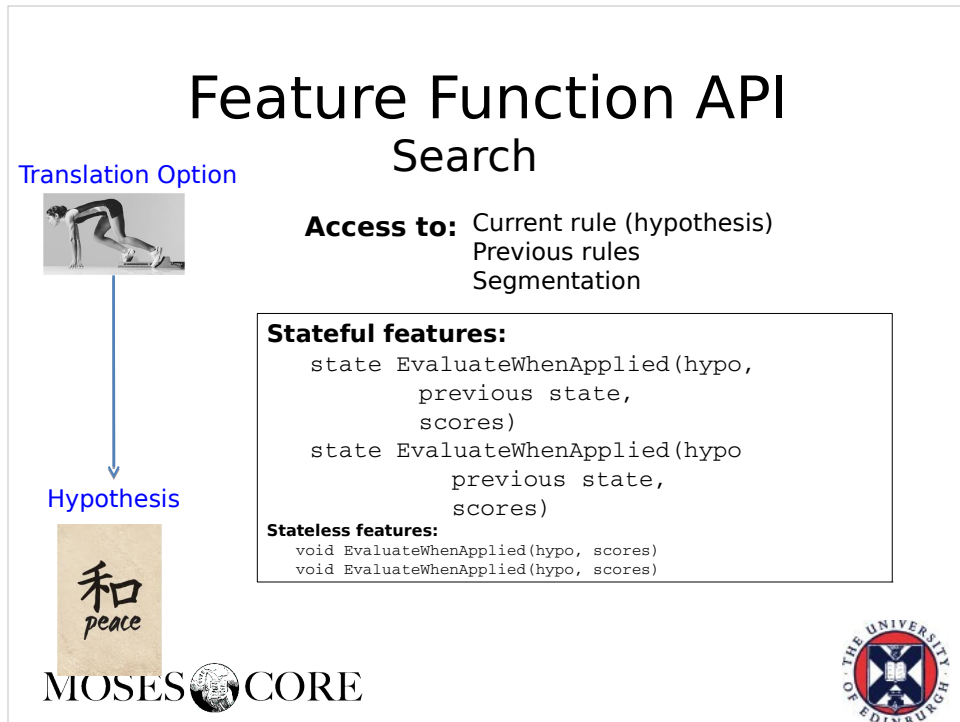


Have a sentence

- 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



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 use this:

- All stateful features
 - Language models
 - Distortion model
 - Lexicalized reordering model
 - ...

MOSES  CORE



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

- derivation tree if hiero/syntax model

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)
```

MOSES  CORE



Recap

- you can score translation rule at 3 stages
in the decoding process

Loading

Applying 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

Adding a New Feature Function

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



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

Template Feature Functions

- SkeletonStatelessFF
- SkeletonStatefulFF
- SkeletonLM
- SkeletonPT
- Video Demo
 - on YouTube

MOSES  CORE



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");