

---

# Moses

Philipp Koehn

3 March 2016



# Who will do MT Research?

- If MT research requires the development of many resources
  - who will be able to do relevant research?
  - who will be able to deploy the technology?
- A few big labs?



- ... or a broad network of academic and commercial institutions?



# Moses



Open source machine translation toolkit  
Everybody can build a state of the art system

# Moses History



- 2002** Pharaoh decoder, precursor to Moses (phrase-based models)
- 2005** Moses started by Hieu Hoang and Philipp Koehn (factored models)
- 2006** JHU workshop extends Moses significantly
- 2006-2012** Funding by EU projects EuroMatrix, EuroMatrixPlus
- 2009** Tree-based models implemented in Moses
- 2012-2015** MosesCore project. Full-time staff to maintain and enhance Moses

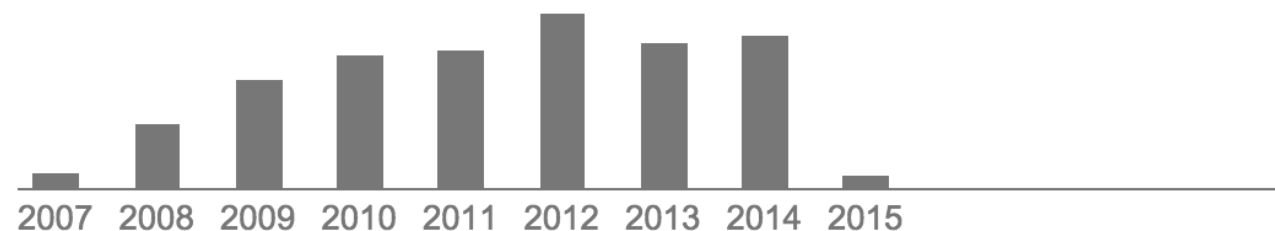
- Web site: <http://www.statmt.org/mones/>
- Github repository: <https://github.com/mones-smt/monesdecoder/>
- Main user mailing list: [mones-support@mit.edu](mailto:mones-support@mit.edu)
  - 1034 subscribers (March 2015)
  - several emails per day

# Academic Use



## Moses: Open source toolkit for statistical machine translation

Authors	Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marc Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej
Publication date	2007/6/25
Conference	Proceedings of the 45th annual meeting of the ACL on interactive poster
Pages	177-180
Publisher	Association for Computational Linguistics
Description	<p>Abstract We describe an open-source toolkit for statistical machine translation support for linguistically motivated factors, (b) confusion network decoding models and language models. In addition to the SMT decoder, the toolkit training, tuning and applying the system to many translation tasks.</p>
Total citations	<a href="#">Cited by 2651</a>



# Commercial Use



- Widely used by companies for internal use or basis for commercial MT offerings

*For this Moses MT market report we identified 22 of the 64 MT operators as Moses-based and we estimate the market share of these operators to be about \$45 million or about 20% of the entire MT solutions market.*

(Moses MT Market Report, 2015)



- Recent evaluation campaign on news translation
- Moses system better than Google Translate
  - English–Czech (2014)
  - French–English (2013, 2014)
  - Czech–English (2013)
  - Spanish–English (2013)
- Moses system as good as Google Translate
  - English–German (2014)
  - English–French (2013)
- Google Translate is trained on more data
- In 2013, Moses systems used very large English language model



- Formally in charge: Philipp Koehn
- Keeps ship afloat: Hieu Hoang
- Mostly academics
  - researcher implements a new idea
  - it works → research paper
  - it is useful → merge with main branch, make user friendly, document
- Some commercial users
  - more memory and time efficient implementations
  - handling of specific text formats (e.g., XML markup)

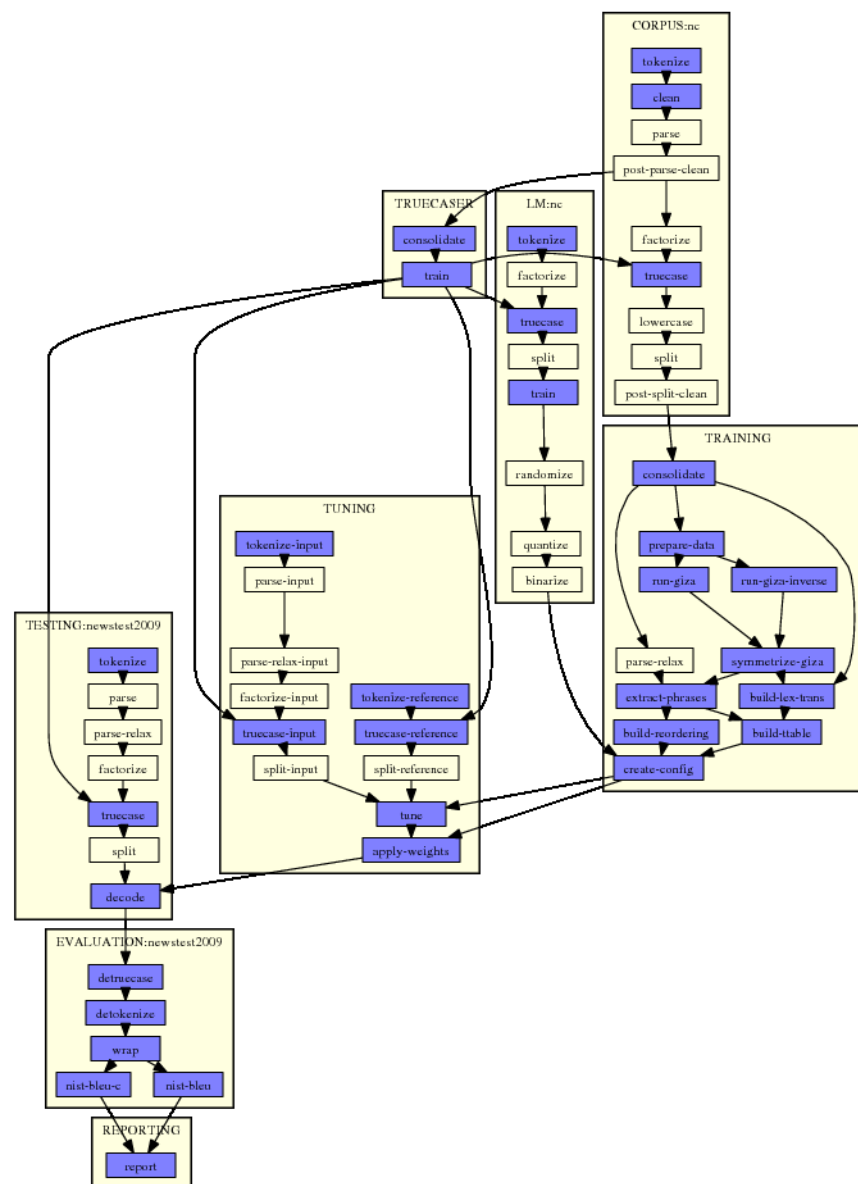
# build a system

# Ingredients

- Install the software
  - runs on Linux and MacOS
  - installation instructions  
<http://www.statmt.org/moses/?n=Development.GetStarted>
- Get some data
  - OPUS (various languages, various corpora)  
<http://opus.lingfil.uu.se/>
  - WMT data (focused on news, defined test sets)  
<http://www.statmt.org/wmt15/translation-task.html>
  - Microtopia , Chinese–X corpus extracted from Twitter and Sina Weibo  
<http://www.cs.cmu.edu/~lingwang/microtopia/>
  - Asian Scientific Paper Excerpt Corpus (Japanese–English and Chinese)  
<http://lotus.kuee.kyoto-u.ac.jp/ASPEC/>
  - LDC has large Arabic–English and Chinese–English corpora

# Steps

11





- Tokenization

The bus arrives in Baltimore .

- Handling case

- lowercasing / recasing

the bus arrives in baltimore .

- truecasing / de-truecasing

the bus arrives in Baltimore .

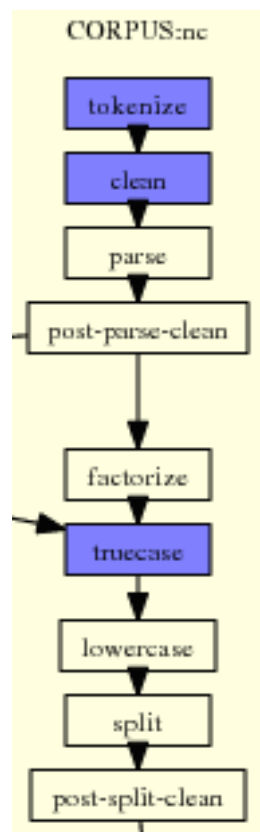
- Other pre-processing, such as

- compound splitting

- annotation with POS tags, word classes

- morphological analysis

- syntactic parsing



# Major Training Steps

- Word alignment
- Phrase table building
- Language model training
- Other component models
  - reordering model
  - operation sequence model
- Organize specification into configuration file

- Parameter tuning
  - prepare input and reference translation
  - use methods such as MERT to optimize weights
  - insert weights into configuration file
- Testing
  - prepare input and reference translation
  - translate input with decoder
  - compute metric scores (e.g., BLEU) with respect to reference

experiment.perl



- Build baseline system
- Try out
  - a newly implemented feature
  - variation of configuration
  - use of different training data
- Build new system
- Compare results
- Repeat

# Motivation

- Avoid typing many commands on command line
- Steps from previous runs could be re-used
- Important to have a record of how a system was built
- Need to communicate system setup to fellow researchers

# Experiment Management System

- Configuration in one file
- Automatic re-use of results of steps from prior runs
- Runs steps in parallel when possible
- Can submit steps as jobs to GridEngine clusters
- Detects step failure
- Provides web based interface with analysis

# Web-Based Interface

## Task: WMT10 French-English (pkoehn)

[Wiki Notes](#) | [Overview of experiments](#) | [/fs/saxnot1/pkoehn-experiment/wmt10-fr-en](#)

<input type="button" value="compare"/>	ID	start	end	<input type="button" value="newstest2009"/>	
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-12]</a>	7+Internal emplus test set	21 Apr	crashed	-	
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-11]</a>	5+interpolated-tm.lm-weighted	19 Feb	20 Feb <small>14: 0.250655 -&gt; 0.250672</small>	<a href="#">26.30 (1.027)</a> <a href="#">27.36 (1.027)</a>	<a href="#">analysis</a> <input type="checkbox"/>
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-10]</a>	5+only-un	04 Feb	07 Feb <small>14: 0.242886 -&gt; 0.242886</small>	<a href="#">25.53 (1.026)</a> <a href="#">26.56 (1.026)</a>	
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-9]</a>	5+only-ep	05 Feb	06 Feb <small>12: 0.242985 -&gt; 0.243113</small>	<a href="#">25.43 (1.034)</a> <a href="#">26.49 (1.034)</a>	
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-8]</a>	5+only-nc	04 Feb	05 Feb <small>19: 0.225010 -&gt; 0.225012</small>	<a href="#">23.62 (1.023)</a> <a href="#">24.54 (1.023)</a>	
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-7]</a>	5+pop20,000+ttl50	19 Feb	19 Feb	<a href="#">26.07 (1.027)</a> <a href="#">27.11 (1.027)</a>	<a href="#">analysis</a> <input type="checkbox"/>
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-6]</a>	5+pop20,000	19 Feb	19 Feb	<a href="#">26.11 (1.026)</a> <a href="#">27.16 (1.026)</a>	<a href="#">analysis</a> <input type="checkbox"/>
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-5]</a>	2+pos-lm	19 Feb	19 Feb <small>8: 0.247961 -&gt; 0.247992</small>	<a href="#">26.13 (1.026)</a> <a href="#">27.16 (1.026)</a>	<a href="#">analysis</a> <input checked="" type="checkbox"/>
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-4]</a>	3+w/o GoodTuring	18 Jan	18 Jan <small>6: 0.240353 -&gt; 0.240353</small>	<a href="#">25.23 (1.025)</a> <a href="#">26.29 (1.025)</a>	
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-3]</a>	2+w/o UN	17 Jan	18 Jan <small>17: 0.242442 -&gt; 0.242442</small>	<a href="#">25.37 (1.024)</a> <a href="#">26.47 (1.024)</a>	
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-2]</a>	baseline GIZA++	10 Feb	10 Feb <small>14: 0.247519 -&gt; 0.247536</small>	<a href="#">25.92 (1.025)</a> <a href="#">26.93 (1.025)</a>	<a href="#">analysis</a> <input checked="" type="checkbox"/>
<input type="checkbox"/> <a href="#">cfg par img</a> <a href="#">[1041-1]</a>	baseline Berkeley	18 Jan	crashed	-	

# Analysis

## Analysis: WMT10 English-German (pkoehn), Set newstest2010, Run 13

Precision					Metrics		Coverage			
precision	1-gram	2-gram	3-gram	4-gram	BLEU-c	BLEU	model	corpus		
correct	31719	13052	6406	3289	16.30 (0.962)	16.68 (0.962)	0	1829 (2.9%)	1486 (2.4%)	1 to 2
	52.8%	22.7%	11.6%	6.3%			1	577 (0.9%)	410 (0.7%)	2 to
wrong	28329	44507	48665	49307	length-diff: -2383 (-3.8%)		2-5	1220 (1.9%)	632 (1.0%)	3 to
							6+	59420 (94.2%)	60518 (96.0%)	4+ to
								by token / <a href="#">by type</a> /		
								<a href="#">details</a>		

### annotated sentences

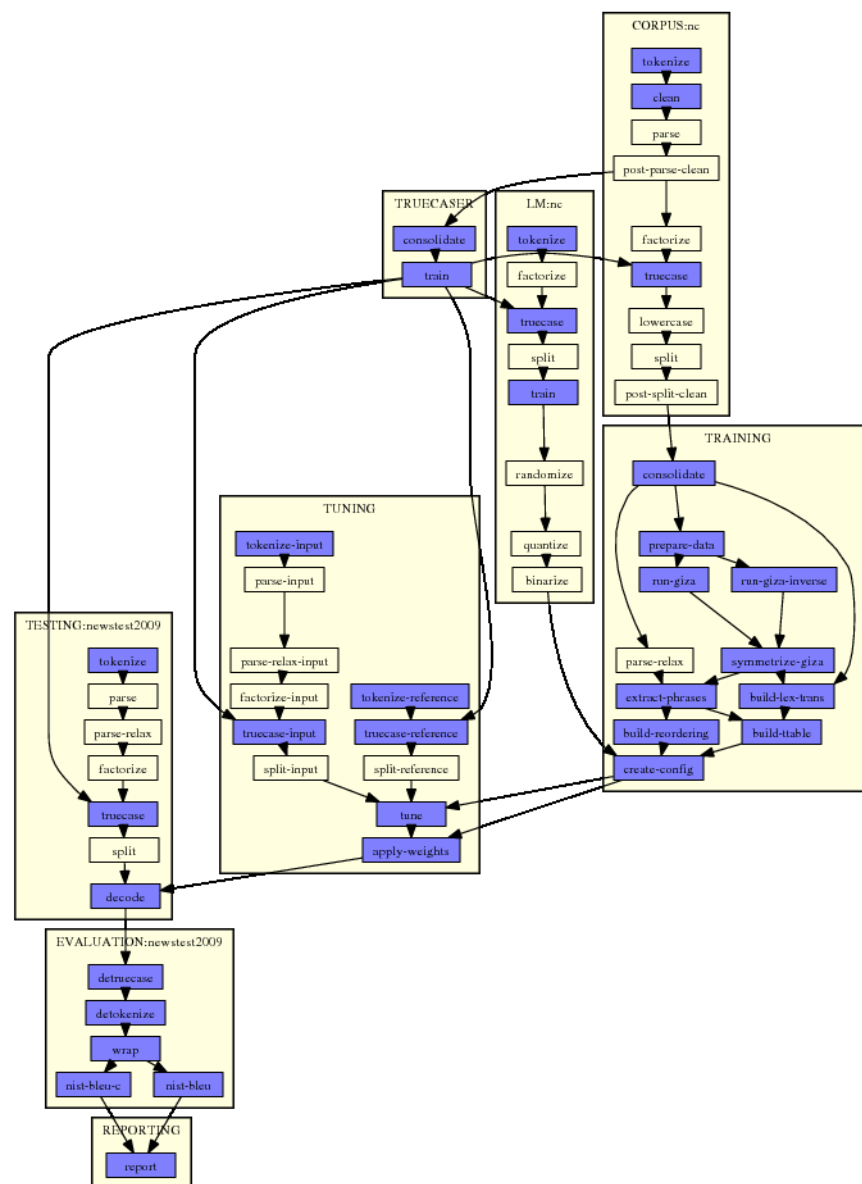
sorted by order [best](#) [worst](#) showing 5 [more](#) [all](#)

[#0]

Barack Obama becomes the fourth American president to receive the Nobel Peace Prize  
 [0.2521] Barack Obama wird der vierte amerikanische Präsident den Friedensnobelpreis erhalten.  
 Barack Obama erhält als vierter US @-@ Präsident den Friedensnobelpreis

- Create a directory for your experiment
- Copy example configuration file `config.toy`
- Edit paths to point to your Moses installation
- Edit paths to your training / tuning / test data
- Run `experiment.perl -config config.toy`

# Automatically Generated Execution Graph 22



# Configuration File

```
#####  
### CONFIGURATION FILE FOR AN SMT EXPERIMENT ###  
#####
```

```
[GENERAL]
```

```
### directory in which experiment is run
```

```
#
```

```
working-dir = /home/pkoehn/experiment
```

```
# specification of the language pair
```

```
input-extension = fr
```

```
output-extension = en
```

```
pair-extension = fr-en
```

```
### directories that contain tools and data
```

```
#
```

```
# moses
```

```
moses-src-dir = /home/pkoehn/moses
```

```
#
```

```
# moses binaries
```

```
moses-bin-dir = $moses-src-dir/bin
```

```
#
```

```
# moses scripts
```

```
moses-script-dir = $moses-src-dir/scripts
```

```
#
```

```
# directory where GIZA++/MGIZA programs resides
```

```
external-bin-dir = /Users/hieuhoang/workspace/bin/training-tools
```

```
#
```



# Specifiying a Parallel Corpus

```
[CORPUS]
### long sentences are filtered out, since they slow down GIZA++
# and are a less reliable source of data. set here the maximum
# length of a sentence
#
max-sentence-length = 80
```

```
[CORPUS:toy]
```

```
### command to run to get raw corpus files
#
# get-corpus-script =
```

```
### raw corpus files (untokenized, but sentence aligned)
#
raw-stem = $toy-data/nc-5k
```

```
### tokenized corpus files (may contain long sentences)
#
#tokenized-stem =
```

```
### if sentence filtering should be skipped,
# point to the clean training data
#
#clean-stem =
```

```
### if corpus preparation should be skipped,
# point to the prepared training data
#
#lowercased-stem =
```

- Very similar to **Makefile**
  - need to build final report
  - ... which requires metric scores
  - ... which require decoder output
  - ... which require a tuned system
  - ... which require a system
  - ... which require training data
- Files can be specified at any point
  - already have a tokenized corpus → no need to tokenize
  - already have a system → no need to train it
  - already have tuning weights → no need to tune
- If you build your own component (e.g., word aligner)
  - run it outside the EMS framework, point to result
  - integrate it into the EMS

# Execution of Step

- For each step, commands are wrapped into a shell script

```
% ls steps/1/LM_toy_tokenize.1* | cat
steps/1/LM_toy_tokenize.1
steps/1/LM_toy_tokenize.1.DONE
steps/1/LM_toy_tokenize.1.INFO
steps/1/LM_toy_tokenize.1.STDERR
steps/1/LM_toy_tokenize.1.STDERR.digest
steps/1/LM_toy_tokenize.1.STDOUT
```

- **STDERR** and **STDERR** are recorded
- **INFO** contains specification information for re-use check
- **DONE** flags finished execution
- **STDERR.digest** should be empty, otherwise a failure was detected

# Execution Plan

- Execution plan follows structure defined in **experiment.meta**

```
get-corpus
  in: get-corpus-script
  out: raw-corpus
  default-name: lm/txt
  template: IN > OUT
tokenize
  in: raw-corpus
  out: tokenized-corpus
  default-name: lm/tok
  pass-unless: output-tokenizer
  template: $output-tokenizer < IN > OUT
  parallelizable: yes
```

- **in** and **out** link steps
- **default-name** specifies name of output file
- **template** defines how command is built (not always possible)
- **pass-unless** and similar indicate optional and alternative steps

# Example: Corpus Tokenization

- Shell script `steps/1/CORPUS_toy_tokenize.1`

```
#!/bin/bash
```

```
PATH=/home/pkoehn/statmt/bin:/home/pkoehn/edinburgh-scripts/scripts:/home/pkoehn/edinburgh-scripts  
/scripts:/usr/lib64/mpi/gcc/openmpi/bin:/home/pkoehn/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11  
:/usr/X11R6/bin:/usr/games
```

```
cd /home/pkoehn/experiment/toy
```

```
echo 'starting at ' `date` ' on ' `hostname`
```

```
mkdir -p /home/pkoehn/experiment/toy/corpus
```

```
mkdir -p /home/pkoehn/experiment/toy/corpus
```

```
/home/pkoehn/moses/scripts/tokenizer/tokenizer.perl -a -l fr -r 1 -o /home/pkoehn/experiment/toy/  
corpus/toy.tok.1.fr < /home/pkoehn/moses/scripts/ems/example/data/nc-5k.fr > /home/pkoehn/  
experiment/toy/corpus/toy.tok.1.fr
```

```
/home/pkoehn/moses/scripts/tokenizer/tokenizer.perl -a -l en < /home/pkoehn/moses/scripts/ems/  
example/data/nc-5k.en > /home/pkoehn/experiment/toy/corpus/toy.tok.1.en
```

```
echo 'finished at ' `date`
```

```
touch /home/pkoehn/experiment/toy/steps/1/CORPUS_toy_tokenize.1.DONE
```

# decoder code

# moses.ini

### MOSES CONFIG FILE ###

[mapping]

0 T 0

[distortion-limit]

6

# feature functions

[feature]

UnknownWordPenalty

WordPenalty

PhrasePenalty

PhraseDictionaryMemory name=TranslationModel0 num-features=4 path=/home/pkoehn/experiment/toy/model/phrase-table.98

input-factor=0 output-factor=0

LexicalReordering name=LexicalReordering0 num-features=6 type=wbe-msd-bidirectional-fe-allff input-factor=0 output-factor=0

path=/home/pkoehn/experiment/toy/model/reordering-table.98.wbe-msd-bidirectional-fe.gz

Distortion

KENLM lazyken=0 name=LM0 factor=0 path=/home/pkoehn/experiment/toy/lm/toy.binlm.98 order=5

# core weights

[weight]

LexicalReordering0= 0.0664129332614665 0.0193333634837915 0.0911160439237806 0.0528731533153271

0.0538468648342602 0.0425200543795641

Distortion0= 0.0734134000992988

LM0= 0.126823453992007

WordPenalty0= -0.133801307986189

PhrasePenalty0= 0.101888283655511

TranslationModel0= 0.025090988893016 0.0854194608356669 0.0892763717037456 0.0381843196363756

UnknownWordPenalty0= 1

# Handling Settings

- Parameters from the `moses.ini` file are stored in object `Parameter`  
function `Parameter::LoadParam` (line 422+ of `Parameter.cpp`) reads in the file
- Global object `StaticData` maintains all global settings
- In function `StaticData::~~StaticData()` (line 95+ of `StaticData.cpp`),  
these settings are defined, partially based on parameters in the `moses.ini` file
- Parameter may be read  
`params = m_parameter->GetParam("stack-diversity");`  
followed by some logic what this means
- Settings may be directly set based on parameter (with default value)  
`m_parameter->SetParameter(m_maxDistortion, "distortion-limit", -1);`



- `ExportInterface.cpp` contains essentially the `main` function
- `decoder_main` (lines 222+)
  - loads configuration file `params.LoadParam(argc, argv)` (line 245)
  - sets global settings
    - `StaticData::LoadDataStatic(&params, argv[0])` (line 250)
  - checks if decoder should be run as server process or in batch mode
    - `if (params.GetParam("server"))` (line 260)
- Typically, the decoder is used in batch mode: `batch_run()` (lines 121+)
  - initialize input / output files
    - `IOWrapper* ioWrapper = new IOWrapper();` (line 132)
  - main loop through input sentences
    - `while(ioWrapper->ReadInput(staticData.GetInputType(), source))` (line 152)
  - set up task of translating one sentence
    - `TranslationTask* task = new TranslationTask(source, *ioWrapper);` (line 272)
  - execute task (may be done via threads)

- Class **TranslationTask** handles one input sentence based on the the search algorithm **staticData.GetSearchAlgorithm()**
- Sets implementation of the search, e.g.,
  - phrase-based: **manager = new Manager(\*m\_source);** (line 66)
  - generic syntax-based: **manager = new ChartManager(\*m\_source);** (line 95)
- Executes search algorithm  
**manager->Decode();** (line 101)
- Deals with output, such as
  - best translation
  - n-best list
  - search graph

- Class **Manager** handles phrase-based model search
- Core function **Manager::Decode()** (line 88+)
  - collects translation options for this sentence  
**m\_transOptColl->CreateTranslationOptions();** (line 110)  
how this works depends on the implementation of the phrase table
  - calls search  
**m\_search->Decode();** (line 123)
- Also implements
  - generation of n-best list
  - various operations on the search graph (e.g., MBR decoding)
  - computations of various reporting statistics

- Default search implemented in class `SearchNormal` (others, e.g., cube pruning)
- Main search loop in `SearchNormal::Decode()` (line 52+)
  - create initial hypothesis (line 58)  
`Hypothesis *hypo = Hypothesis::Create(m_manager, m_source, m_initialTransOpt);`
  - add to stack 0  
`m_hypoStackColl[0]->AddPrune(hypo);` (line 59)
  - loop through the stacks  
`for (iterStack = m_hypoStackColl.begin() ; iterStack != m_hypoStackColl.end() ; ++iterStack)` (line 63)
    - \* prune stack (line 78)  
`sourceHypoColl.PruneToSize(staticData.GetMaxHypoStackSize());`
    - \* loop through hypotheses (line 87)  
`for (iterH = sourceHypoColl.begin(); iterH != sourceHypoColl.end(); ++iterH)`
      - process each hypothesis  
`Hypothesis &hypothesis = **iterHypo;` (line 88)  
`ProcessOneHypothesis(hypothesis);` (line 89)

# Expanding One Hypothesis

- Function `ProcessOneHypothesis` (line 109+ of `SearchNormal.cpp`)
- Check which translation options can be applied
  - overlap with already translated
  - reordering restrictions
- For valid span, execute `ExpandAllHypotheses(hypothesis, startPos, endPos);`
- Function `ExpandAllHypotheses` (line 247++ of `SearchNormal.cpp`)
  - find translation options  

```
const TranslationOptionList* tol = m_transOptColl.GetTranslationOptionList(startPos, endPos);
```
  - loop through them  

```
for (iter = tol->begin() ; iter != tol->end() ; ++iter)  
    ExpandHypothesis(hypothesis, **iter, expectedScore);
```

# Expanding One Hypothesis (cnt.)

- Function `SearchNormal::ExpandHypothesis` (line 283++)
  - create new hypothesis (line 294)  
`newHypo = hypothesis.CreateNext(transOpt);`
  - how many words did it translate so far? (line 351)  
`size_t wordsTranslated = newHypo->GetWordsBitmap().GetNumWordsCovered();`
  - add to the right stack (line 355)  
`m_hypoStackColl[wordsTranslated]->AddPrune(newHypo);`

# Create New Hypothesis

- Hypothesis class **Hypothesis**
- Expanding existing hypothesis → initializer **Hypothesis::Hypothesis** (line 82+)
  - back pointer to previous hypothesis  
**m\_prevHypo(&prevHypo)** (line 84)
  - notes which translation option was used  
**m\_transOpt(transOpt)** (line 96)
  - adds translation option scores (line 100)  
**m\_currScoreBreakdown.PlusEquals(transOpt.GetScoreBreakdown());**
  - notes which words have been translated  
**m\_sourceCompleted(prevHypo.m\_sourceCompleted )** (line 85)  
**m\_sourceCompleted.SetValue(m\_currSourceWordsRange.GetStartPos(),  
m\_currSourceWordsRange.GetEndPos(), true);** (line 107)
  - ... and other book keeping

- All hypothesis are scores with feature functions
- Each is implemented with its own class (see directory **FF**)
- Scoring
  - if only depend on the translation option
    - need to implement function **EvaluateInIsolation**
  - if additionally depends on input sentence
    - need to implement function **EvaluateWithSourceContext**
  - if depends on application context
    - need to implement function **EvaluateWhenApplied**
- If stateful, **EvaluateWhenApplied** returns feature state
- YouTube video:  
<https://www.youtube.com/watch?v=x-uo522bplw>