Final Report

of the

2006 Language Engineering Workshop

Open Source Toolkit for Statistical Machine Translation: Factored Translation Models and Confusion Network Decoding

http://www.clsp.jhu.edu/ws2006/groups/ossmt/

http://www.statmt.org/moses/

Johns Hopkins University

Center for Speech and Language Processing

Philipp Koehn, Marcello Federico, Wade Shen, Nicola Bertoldi,
Ondrej Bojar, Chris Callison-Burch, Brooke Cowan,
Chris Dyer, Hieu Hoang, Richard Zens,
Alexandra Constantin, Christine Corbett Moran, Evan Herbst

October 24, 2006

Abstract

Acknowledgments

Team Members

- Philipp Koehn, Team Leader, University of Edinburgh
- Marcello Federico, Senior Researcher, ITC-IRST
- Wade Shen, Senior Researcher, Lincoln Labs
- Nicola Bertoldi, Senior Researcher, ITC-IRST
- Chris Callison-Burch, Graduate Student, University of Edinburgh
- Richard Zens, Graduate Student, Aachen University
- Hieu Hoang, Graduate Student, University of Edinburgh
- Brooke Cowan, Graduate Student, MIT
- Ondrej Bojar, Graduate Student, Charles University
- Chris Dyer, Graduate Student, University of Maryland
- Alexandra Constantin, Undergraduate Student, Williams College
- Evan Herbst, Undergraduate Student, Cornell
- Christine Corbett Moran, Undergraduate Student, MIT

Contents

Chapter 1

Introduction

Philipp Koehn: overview of the goals of the workshop 2-4 pages

Chapter 2

Factored Translation Models

The current state-of-the-art approach to statistical machine translation, so-called phrase-based models, are limited to the mapping of small text chunks without any explicit use of linguistic information, may it be morphological, syntactic, or semantic. Such additional information has been shown to be valuable, by integrating it in pre-processing or post-processing steps.

For instance, gains have been achieved by handling Arabic morphology through stemming or splitting off of affixes that typically translate into individual words in English. Another example is our earlier work on methods to reorder German input, so it is more similar to English output sentence order, which makes it more amendable to the phrase-based approach (CITE).

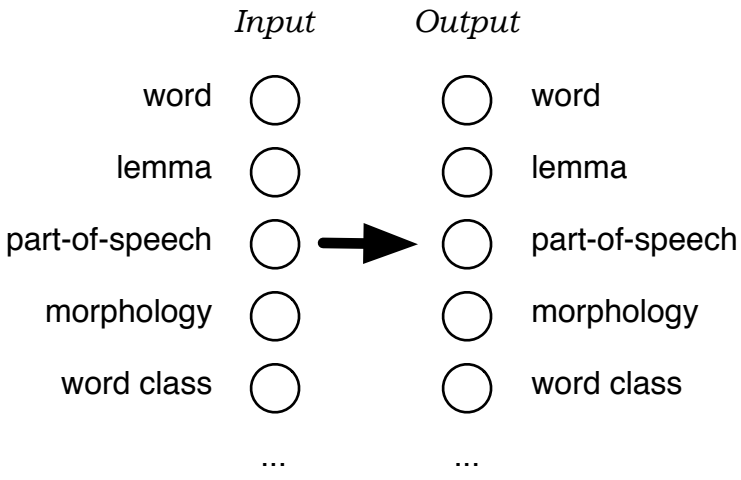
However, a tighter integration of linguistic information into the translation model is desirable for two reasons:

- Translation models that operate on more general representations, such as lemmas instead of surface forms of words, can draw on richer statistics and overcome the data sparseness problems caused by limited training data.

- Many aspects of translation can be best explained on a morphological, syntactic, or semantic level. Having such information available to the translation model allows the direct modeling of these aspects. For instance: reordering at the sentence level is mostly driven by general syntactic principles, local agreement constraints show up in morphology, etc.

Therefore, we developed a framework for statistical translation models that tightly integrates additional information. Our framework is an extension of the phrase-based model. It adds additional annotation at the word

level. A word in our framework is not anymore only a token, but a vector of factors that represent different levels of annotation.



Typical factors that we experimented with at this point include surface form, lemma, part-of-speech tag, morphological features such as gender, count and case, automatic word classes, true case forms of words, shallow syntactic tags, as well as dedicated factors to ensure agreement between syntactically related items.

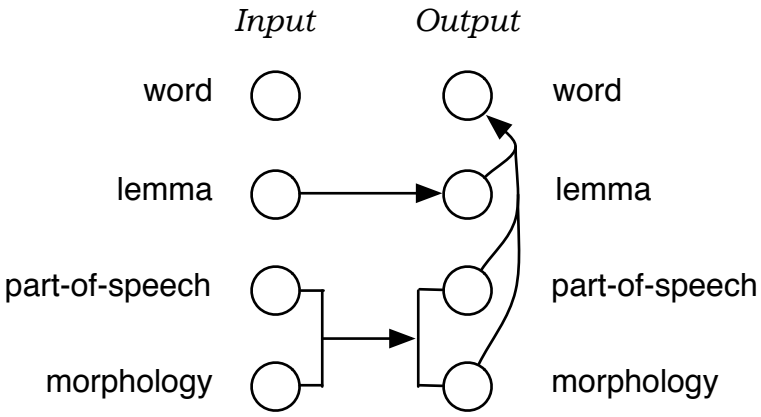
2.1 Motivation

One example to illustrate the short-comings of the traditional surface word approach in statistical machine translation is the poor handling of morphology. Each word form is treated as a token in itself. This means that the translation model treats, say, the word *house* completely independent of the word *houses*. Any instance of *house* in the training data does not add any knowledge to the translation of *houses*.

In the extreme case, while the translation of *house* may be known to the model, the word *houses* may be unknown and the system will not be able to translate it. While this problem does not show up as strongly in English — due to the very limited morphological production in English — it does constitute a significant problem for morphologically rich languages such as Arabic, German, Czech, etc.

Thus, it may be preferably to model translation between morphologically rich languages on the level of lemmas, and thus pooling the evidence for different word forms that derive from a common lemma. In such a model, we would want to translate lemma and morphological information separately, and combine this information on the target side to generate the ultimate output surface words.

Such a model, which makes more efficient use of the translation lexicon, can be defined as a factored translation model. See below for an illustration of this model in our framework.



Note that while we illustrate the use of factored translation models on such a linguistically motivated example, our framework also applies to models that incorporate statistically defined word classes.

2.2 Decomposition of Factored Translation

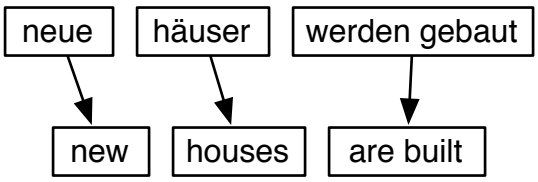
The translation of the factored representation of source words into the factored representation of target words is broken up into a sequence of **mapping steps** that either **translate** input factors into output factors, or **generate** additional target factors from existing target factors.

Recall the previous of a factored model that translates using morphological analysis and generation. This model breaks up the translation process into the following steps:

- Translating of input lemmas into target lemmas
- Translating of morphological and syntactic factors

- Generating of surface forms given the lemma and linguistic factors

Factored translation models build on the phrase-based approach that breaks up the translation of a sentence in the translation of small text chunks (so-called phrases). This model implicitly defines a segmentation of the input and output sentences into such phrases. See an example below:



Our current implementation of factored translation models follows strictly the phrase-based approach, with the additional decomposition of phrase translation into a sequence of mapping steps. Since all mapping steps operate on the same phrase segmentation of the input and output sentence into phrase pairs, we call these **synchronous factored models**.

Let us now take a closer look at one example, the translation of the one-word phrase *häuser* into English. The representation of *häuser* in German is: surface-form *häuser*, lemma *haus*, part-of-speech *NN*, count *plural*, case *nominative*, gender *neutral*.

Given the three mapping steps in our morphological analysis and generation model may provide the following applicable mappings:

- **Translation:** Mapping lemmas
  - *haus* → *house, home, building, shell*
- **Translation:** Mapping morphology
  - *NN|plural-nominative-neutral* → *NN|plural, NN|singular*
- **Generation:** Generating surface forms
  - *house|NN|plural* → *houses*
  - *house|NN|singular* → *house*
  - *home|NN|plural* → *homes*
  - ...

The German *haus|NN|plural|nominative|neutral* is expanded as follows:

- **Translation:** Mapping lemmas
  { *?|house|?|?, ?|home|?|?, ?|building|?|?, ?|shell|?|?* }

- **Translation:** Mapping morphology
  { *?|house|NN|plural, ?|home|NN|plural, ?|building|NN|plural,*
  ?|shell|NN|plural, ?|house|NN|singular, ... }

- **Generation:** Generating surface forms
  { *houses|house|NN|plural, homes|home|NN|plural, buildings|building|NN|plural,*
  shells|shell|NN|plural, house|house|NN|singular, ... }

2.3 Statistical Modeling

Factored translation models follow closely the statistical modeling methods used in phrase-based models. Each of the mapping steps is modeled by a feature function. This function is learned from the training data, resulting in translation tables and generation tables.

Phrase-based statistical translation models are acquired from word-aligned parallel corpora by extracting all phrase-pairs that are consistent with the word alignment. Given the set of extracted word pairs with counts, various scoring functions are estimated, such as conditional phrase translation probabilities based on relative frequency estimation.

Factored models are also acquired from word-aligned parallel corpora. The tables for translation steps are extracted in the same way as phrase translation tables. The tables for generation steps are estimated on the target side only (the word alignment plays no role here, and in fact additional monolingual data may be used). Multiple scoring functions may be used for generation and translation steps, we used in our experiments

- five scores for translation steps: conditional phrase translation probabilities in both direction (foreign to English and vice versa), lexical translation probabilities (foreign to English and vice versa), and phrase count;

- two scores for generation steps: conditional generation probabilities in both directions (new target factors given existing target factors and vice versa).

As in phrase-based models, the different components of the model are combined in a log-linear model. In addition to traditional components — language model, reordering model, word and phrase count, etc. — each

mapping steps forms a component with five (translation) or two (generation) features. The feature weights in the log-linear model are determined using a minimum error rate training method (cite Och, simplex).

2.4 Efficient Decoding

Compared to phrase-based models, the decomposition of the phrase translation into several mapping steps creates additional computational complexity. Instead of a simple table lookup to obtain the possible translation for an input phrase, now a sequence of such tables have to be consulted and their content combined.

Since all translation steps operate on the same segmentation, the **expansion** of these mapping steps can be efficiently pre-computed prior to the heuristic beam search, and stored as translation options (recall the example in Section 2.2, where we carried out the expansion for one input phrase). This means that the fundamental search algorithm does not change. Only the scoring of hypothesis becomes slightly more complex.

However, we need to be careful about the combinatorial explosion of the number of translation options given a sequence of mapping steps. If one or many mapping steps result in a vast increase of (intermediate) expansions, this may be become unmanageable. We currently address this problem by early pruning of expansions, and limiting the number of translation options per input phrase to a maximum number, by default 50.

2.5 Future Research

2.5.1 Smarter search for multi-factored models

Although factored translation models can be successfully used to improve translation quality (in terms of BLEU score, as well as other metrics, such as the rate of agreement errors in the output text), initial experiments suggest two changes to the translation model and decoding strategy that will enable more sophisticated models (that take advantage of linguistically motivated decomposition and generation processes, for example) and enable the models to be applied in situations where the target language is morphologically less complex (such as English).

Shorter secondary spans

One significant limiting factor in the performance of multi-factored translation models is the due to the present requirement that successive translation steps all translate identical source and target spans. If a compatible translation is not found for a secondary translation step (either because hypotheses with compatible factors were discarded earlier or because there is no possible translation in the phrase table for the secondary translation step), the hypothesis is abandoned. This has considerable benefit from a computational perspective since it constrains the search space for potential targets when translating secondary factors. However, it causes a number of significant problems:

1. In models where a secondary factor is both generated from another target factor and translated from a source factor, any pruning before both steps have completed runs the risk of producing not just degraded output, but failing to find any adequate translation.

2. Because a compatible translation must be found in secondary steps for a translation hypothesis to survive, it is difficult to filter secondary translation tables. This results in very large tables which are inefficient to load and have considerable memory overhead.

3. When secondary translation steps fail and hypotheses are abandoned, the model is forced to rely on shorter translation units for the primary translation step. This is in direct conflict to the potential benefits that can be gained by richer statistics.

There are several possible ways that the exact-span match requirement might be addressed. One solution that is computationally tractable is to back off to shorter spans only in the event of a failure to find any possible translation candidates during subsequent translation steps. The problem that arises is how the spans established should be translated once multiple translation units can be used. Reordering within phrases is certainly quite common. These can be further constrained to either match alignments that are suggested by the initial span.

Translation-constrained generation with FSTs

Currently when translation hypotheses are enumerated for a particular span of the source sentence (the first step in the translation process), each step in the mapping (whether translation or generation) occurs serially and pruning

occurs after each step. Furthermore, multiple generation and translation steps frequently target the same factor in a particular model (for example, a target-side lemma may generate target-side part of speech candidates, and source-side part of speech information may also be translated into target-side part of speech sequences). When the same factor is generated in multiple mapping steps, they must all converge or the hypothesis is abandoned.

The serial approach to computing translation options has two primary drawbacks:

1. Since a translation hypotheses is abandoned unless all steps in the mapping succeed fully, it is often the case that many hypotheses that survive pruning after one step are abandoned at a later step, and that many that were pruned would have ended up being a reasonable hypothesis.

2. There are an exponential number of generation candidates available for a given target span (where the exponent is the length of the span and the base is the average number of targets from a given source in the generation table).

To mitigate both of these problems, it is possible to execute generation and translation steps concurrently. The generation table can be formalized as a finite state transducer that maps between factors on the target language side. The phrase table can be formalized as a finite state transducer (FST) that maps between source language factors to target language factors. Thus all devices that generate a given target language factor can be conceived of as FSTs. These FSTs can be composed with well-known algorithms that will generally run much more efficiently (this is exactly the method that is used to minimize the paths searched through confusion networks: the source-side of the phrase table is treated as a finite state automaton that is intersected with the confusion network).

Hybrid multi-factor and single-factor models

A characteristic feature of natural languages is that elements of a wide variety of sizes, from sub-word morphemes to complete sentential units may be lexicalized. The larger lexicalized units (for example, idioms and "stock phrases") frequently exhibit idiosyncratic- rather than compositional- meaning and are the bread and butter of conventional phrase-based machine translation systems. The phrase model can simply "memorize" the larger

units and their corresponding translations, which often tend to be idiosyncratic in the target language. This is arguably one of the significant benefits of conventional phrase-based translation models since mistranslating common stock phrases results is significantly diminished fluency and understanding, and common evaluation metrics assign a great deal of value to correctly translated stock phrases (since they are, by definition, several words in length and tend to exhibit relatively fixed word order).

Multi-factored models that analyze the source language in terms of underlying lemmas, parts of speech, and morphological information, and then translate these factors in a piecemeal fashion may actually result in a system that performs less well on commonly occurring lexicalized phrases. There are at least two reasons for this. First, the process of lexicalization results in the retention of archaic or otherwise unusual forms, possibly in unusual configurations. Thus, when these units are analyzed, they exhibit unusual morphological features and parts of speech. These unusual features introduce significant sparseness in the sequence models in the target language and reduce the overall probability that would be assigned to a correct translation. Second, single-factored translation models generally have very good data on lexicalized phrases (since they must, in order to be acquired by language learners as lexicalized elements, occur with a reasonable frequency). Therefore even if the underlying linguistic phenomena are rather unusual, they are well modeled in both translation models and target language models. Moreover, if the target translation does contain unusual items, these are more likely to occur in a very specific context, which will generally decrease the net language model cost that would otherwise be expected for infrequently occurring items.

To retain the benefits associated with multi-factored models (generalization across inflected forms, making use of data with richer statistics) but retaining the benefits of single-factored "surface" translation models (better handling of stock phrases), a more effective method would be to allow both a surface form based single-factored model to propose hypotheses for various spans in a sentence that would compete with hypotheses generated by a multi-factored model. Since multi-factored models consist of different models with different scoring functions, the costs associated with the two classes of hypotheses are not directly comparable. To mitigate this difficulty and establish a trading relation between the two classes of hypotheses, a single-factor penalty parameter will be introduced that can be tuned along with the other parameters used in decoding.

Chapter 3

Confusion Network Decoding

Marcello Federico and Richard Zens: cut and paste from your journal paper?

Chapter 4

Open Source Toolkit

4.1 Overall design

In developing the Moses decoder we were aware that the system should be open-sourced if it were to gain the support and interest from the machine translation community that we had hoped. There were already several proprietary decoders available which frustrated the community as the details of their algorithms could not be analysed or changed. However, making the source freely available is not enough. The decoder must also advance the state of the art in machine translation to be of interest to other researchers. Its translation quality and runtime resource consumption must be comparable with the best available decoders. Also, as far as possible, it should be compatible with current systems which minimize the learning curve for people who wish to migrate to Moses. We therefore kept to the following principles when developing Moses:

- Accessibility
- Easy to Maintain
- Flexibility
- Easy for distributed team development
- Portability

The number of functionality added in the six weeks by every member of the team at the Johns Hopkins University workshop, as can be seen from the figures below, is evident that many of these design goals were met.

	$BeforeJHUworkshop$	$AfterJHUworksh$
Lines of code	9000	15,6
Number of classes	30	
Lines of code attributed to original developer	100%	5

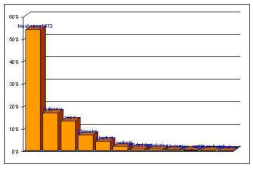


Figure 4.1: Percentage of code contribute by each developer

By adding factored translation to conventional phrase based decoding we hope to incorporate linguistic information into the translation process in order to create a competitive system.

Resource consumption is of great importance to researchers as it often determine whether or not experiments can be run or what compromises needs to be taken. We therefore also benchmarked resource usage against another phrase-based decoder, Pharaoh, as well as other decoders, to ensure that they were comparable in like-for-like decoding.

It is essential that features can be easily added, changed or replace, and that the decoder can be used as a toolkit in ways not originally envisaged. We followed strict object oriented methodology; all functionality was abstracted into classes which can be more readily changed and extended. For example, we have two implementations of single factor language models which can be used depending on the functionality and licensing terms required. Other implementations for very large and distributed LMs are in the pipeline and can easily be integrated into Moses. The framework also allows for factored LMs; a joint factor and skipping LM are currently available.

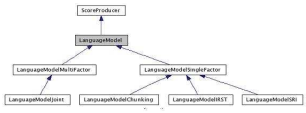


Figure 4.2: Language Model Framework

Another example is the extension of Moses to accept confusion networks as input. This also required changes to the decoding mechanism.

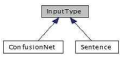


Figure 4.3: Input

Nevertheless, there will be occasions when changes need to be made which are unforeseen and unprepared. In these cases, the coding practises and styles we instigated should help, ensuring that the source code is clear, modular and consistent to enable the developers to quickly assess the algorithms and dependencies of any classes or functions that they may need to

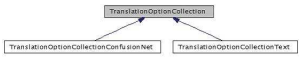


Figure 4.4: Translation Option Collection

change.

A major change was implemented when we decided to collect all the score keeping information and functionality into one place. That this was implemented relatively painlessly must be partly due to the clarity of the source code.

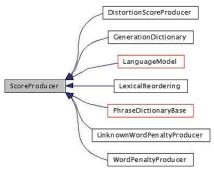


Figure 4.5: Scoring framework

The decoder is packaged as a library to enable users to more easily comply with the LGPL license. The library can also be embedded in other programs, for example a GUI front-end or an integrated speech to text translator.

4.1.1 Entry Point to Moses library

The main entry point to the library is the class

```
Manager
```

For each sentence or confusion network to be decoded, this class is instantiated and the following function called

```
ProcessSentence()
```

Its outline is shown below

```
CreateTranslationOptions()
for each stack in m_hypoStack
   prune stack
   for each hypothesis in stack
      ProcessOneHypothesis()
```

Each contiguous word coverage (span) of the source sentence is analysed in

```
CreateTranslationOptions()
```

and translations are created for that span. Then each hypothesis in each stack is processed in a loop. This loop starts with the stack where nothing has been translated which has been initialised with one empty hypothesis.

4.1.2 Creating Translations for Spans

The outline of the function

```
TranslationOptionCollection::CreateTranslationOptions()
```

is as follows:

```
for each span of the source input
   CreateTranslationOptionsForRange(span)
ProcessUnknownWord()
Prune()
CalcFutureScoreMatrix()
```

A translation option is a pre-processed translation of the source span, taking into account all the translation and generation steps required. Translations options are created in

```
CreateTranslationOptionsForRange()
```

which is out follows

```
ProcessInitialTranslation()
for every subequent decoding step
   if step is Translation
      DecodeStepTranslation::Process()
   else if step is Generation
      DecodeStepGeneration::Process()
Store translation options for use by decoder
```

However, each decoding step, whether translation or generation, is a subclass of

```
DecodeStep
```

so that the correct Process() is selected by polymorphism rather than using if statements as outlined above.

4.1.3 Unknown Word Processing

After translation options have been created for all contiguous spans, some positions may not have any translation options which covers it. In these cases, CreateTranslationOptionsForRange() is called again but the table limits on phrase and generation tables are ignored.
If this still fails to cover the position, then a new target word is create by copying the string for each factor from the untranslatable source word, or the string UNK if the source factor is null.

Source Word		New Target Word
Jimmy	→	Jimmy
Proper Noun	→	Proper Noun
-	→	UNK
-	→	UNK

This algorithm is suitable for proper nouns and numbers, which are one of the main causes of unknown words, but is incorrect for rare conjugation of source words which have not been seen in the training corpus. The algorithm also assumes that the factor set are the same for both source and target language, for instance, th list of POS tags are the same for source and target. This is clearly not the case for the majority of language pairs. Language dependent processing of unknown words, perhaps based on morphology. is a subject of debate for inclusion into Moses.
Unknown word processing is also dependent on the input type - either sentences or confusion networks. This is handled by polymorphism, the call stack is

```
Base::ProcessUnknownWord()
   Inherited::ProcessUnknownWord(position)
      Base::ProcessOneUnknownWord()
```

where

```
Inherited::ProcessUnknownWord(position)
```

is dependent on the input type.

4.1.4 Scoring

A class is created which inherits from

```
ScoreProducer
```

for each scoring model. Moses currently uses the following scoring models:

Scoringmodel	*Class*
Distortion	DistortionScoreProducer
WordPenalty	WordPenaltyProducer
Translation	PhraseDictionary
Generation	GenerationDictionary
LanguageModel	LanguageModel

The scoring framework includes the classes

```
ScoreIndexManager
ScoreComponentCollection
```

which takes care of maintaining and combining the scores from the different models for each hypothesis.

4.1.5 Hypothesis

A hypothesis represents a complete or incomplete translation of the source. Its main properties are

Variables	
m_sourceCompleted	Which source words have already been translated
m_currSourceWordsRange	Source span current being translated
m_targetPhrase	Target phrase currently being used
m_prevHypo	Pointer to preceding hypothesis that translated the other words
m_scoreBreakdown	Scores of each scoring model
m_arcList	List of equivalent hypothesis which have lower score than curre

Hypothesis are created by calling the constructor with the preceding hypothesis and an appropriate translation option. The constructors have been wrapped with static functions, Create(), to make use of a memory pool of hypotheses for performance.

Many of the functionality in the Hypothesis class are for scoring. The outline call stack for this is

```
CalcScore()
   CalcDistortionScore()
   CalcLMScore()
   CalcFutureScore()
```

The Hypothesis class also contains functions for recombination with other hypotheses. Before a hypothesis is added to a decoding stack, it is compare to other other hypotheses on the stack. If they have translated the same source words and the last n-words for each target factor are the same (where n is determined by the language models on that factor), then only the best scoring hypothesis will be kept. The losing hypothesis may be used latter when generating the n-best list but it is otherwise not used for creating the

best translation.

In practise, language models often backoff to lower n-gram than the context words they are given. Where it is available, we use information on the backoff to more agressively recombine hypotheses, potentially speeding up the decoding.

The hypothesis comparison is evaluated in

`NGramCompare()`

while the recombination is processed in the hypothesis stack class

`HypothesisCollection::AddPrune()`

and in the comparison functor class

`HypothesisRecombinationOrderer`

4.1.6 Phrase Tables

The main function of the phrase table is to look up target phrases give a source phrase, encapsulated in the function

`PhraseDictionary::GetTargetPhraseCollection()`

There are currently two implementation of the PhraseDictionary class

PhraseDictionaryMemory	Based on std::map. Phrase table loaded completely and held in memory
PhraseDictionaryTreeAdaptor	Binarized phrase table held on disk and loaded on demand.

4.1.7 Command Line Interface

The subproject, moses-cmd, is a user of the Moses library and provides an illustration on how the library functions should be called. It is licensed under a BSD license to enable other users to copy it source code for using the Moses library in their own application.

However, since most researchers will be using a command line program for running experiments, it will remain the defacto Moses application for the time being.

Apart from the main() function, there are two classes which inherites from

the moses abstract class, InputOutput:

```
IOCommandLine
IOFile (inherites from IOCommandLine)
```

These implement the required functions to read and write input and output (sentences and confusion network inputs, target phrases and n-best lists) from standard io or files.

4.2 Software Engineering Aspects

4.2.1 Regression Tests

Moses includes a suite of regression tests designed to ensure that behavior that has been previously determined to be correct does not break as new functionality is added, bugs are fixed, or performance improvements are made. The baseline behavior for the regression testing is determined in three ways:

1. Expected behavior based on off-line calculations (for example, given a small phrase table and sample input, one can work through the search space manually and compute the expected scores for a translation hypothesis).

2. Expected values based on comparisons with other systems (for example, language modeling toolkits provide the ability to score a sentence. Such a tool can be used to calculate the expected value of the language model score that will be produced by the decoder).

3. Expected values based on previous behavior of the decoder (some output behavior is so complex that it is impractical or impossible to determine externally what the expected values are; however, it is reasonable to assume that localized bug-fixes, the addition of new functionality, or performance improvements should not impact existing behavior).

The nature of statistical machine translation decoding makes achieving substantial and adequate test coverage possible with simple black-box testing. Aggregate statistics on the number of hypotheses generated, pruned, explored, as well as comparisons of the exact costs and translations for certain sample sentences provide ample evidence that the models and code that

is utilized in decoding is working adequately since these values tend to be highly sensitive to even minor changes in behavior.

How it works

The test harness (invoked with the command `run-test-suite`) runs the decoder that is to be tested (specified to the script with the `--decoder` command line option) with a series of configuration files and translation inputs. The output from the decoder, which is written to `stdout` and `stderr`, is post-processed by small scripts that pull out the data that is going to be compared for testing purposes. These values are compared with the baseline and a summary is generated.

Timing information is also provided so that changes that have serious performance implications can be identified as they are made. This information is dependent on a variety of factors (system load, disk speed), so it is only useful as a rough estimate.

Versioning

The code for the regression test suite is in the `regression/tests` subdirectory of the Subversion repository. The inputs and expected values for each test case in the test suite are stored together in `regression-tests/tests`. The test suite is versioned together with the source code for several reasons:

1. As bugs are identified and fixed that effect existing behavior, the testing code needs to be updated.

2. As new functionality is added, testing code exercising this functionality needs to be added (see below for more details).

By versioning the regression tests together with the source code, it should be possible to minimize when developers need to worry about expected test failures.

The data (language models, phrase tables, generation tables, etc.) that is used by the individual test cases is versioned along with the source code, but because of its size (currently about 60MB), it is not stored in Subversion. When test suite is run in a new environment or one with an improper version of the test data, it will fail and provide instructions for retrieving and installing the proper version of the testing data (via HTTP) from the test data repository, which is currently `http://statmt.org`.

Making changes to existing tests

As changes are made that effect the decoder's interface (output format, command line interface, or configuration file format) and bugs that effect existing decoder behavior are fixed, it will often be necessary to update either the expected values, the scripts that post-process the decoder output, or the configuration files. These files can be edited in the same manner as the rest of the source code and should be submitted along with the corresponding code changes.

If changes need to be made to the test data, a new tar-ball must be generated that contains all of the test data for all regression tests and submitted to the repository maintainer. Once it is available for download, the `TEST_DATA_VERSION` constant in `MosesRegressionTesting.pm` can be incremented to point to the new version.

Adding regression tests

As new functionality is incorporated into Moses, regression tests should be added that guarantee that it will continue to be behave properly as further changes are made. Generally, testing new models with multi-factored models is recommended since common single-factored models exercise only a subset of the logic.

If new regression tests have new data dependencies, the test data will need to be updated. For more information on this workflow, refer to the previous section.

4.2.2 Accessability

The source code for the Moses project is housed at Sourceforge.net in a subversion repository. The URL for the project is:

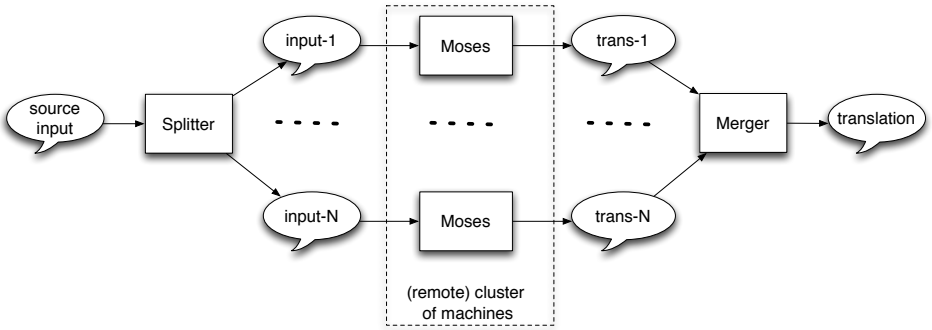
`http://sourceforge.net/projects/mosesdecoder/`

The source code is publicly accessible and in two ways:

1. Pre-packaged tar-balls are available for download directly from project page at Sourceforge.

2. The current development source code can be accessed with a subversion client (see `http://subversion.tigris.org/` for more details how to acquire and use the client software).

Figure 4.6: The parallelization module for `Moses`.



4.2.3 Documentation

Philipp Koehn and Chris Callison-Burch: Doxygen

4.3 Parallelization

The decoder implemented in `Moses` translates its input sequentially; in order to increase the speed of the toolkit a parallelization module was developed which exploits several instances of the decoder and feed them with subsets of the scource input.

As shown in Figure 4.3, the procedure we implemented is reasonably easy: first, the source input is equally divided into N parts, then N instances of the `Moses` translate them; finally, the full translation is obtained by ordering and merging the translation of all input parts.

All `Moses` instances are assumed to run on a (possibly remote) cluster. No restriction on the number of `Moses` instances is given.

Time to perform a full translation with one `Moses` instance comprises the time to load data, which is constant, and time to translate the input, which is proportional to its size. The parallelization module requires an additional time to access the cluster, which is strictly related to the real load of the cluster itself and hardly forecastable. Time to split the input and merge the output can be considered negligible with respect to the translation time. Moreover, an ending delay can be observe because the merging module should wait that all decoders have completed their translations, and this does not necessarily happen at the same time. A good splitting policy which allows a balanced translation time among all decoders, improves the

	standalone	1 proc	5 proc	10 proc	20 proc
10 sentences	6.3	13.1	9.0	9.0	–
100 sentences	5.2	5.6	3.0	1.7	1.7
1000 sentences	6.3	6.5	2.0	1.6	1.1

Table 4.1: Average time (seconds) to translate 3 input sets with a standalone `Moses` and with its parallel version.

effciency of the whole parallelization module.

We tested the gain in time that the parallelization module can provide to the toolkit on the Spanish-English EuroParl task. 3 input sets were created of 10, 100 1000 sentences and translated using a standalone `Moses`, and the parallelization module exploiting difference number of `Moses` instances (1, 5, 10, 20). Decoders ran on the 18-processor CLSP cluster. As in the real situation, its load was not in control, and hence the immediate availability of the processors was not assured. Table 4.3 reports the average translation times for all conditions.

Some considerations can be drawn by inspecting these figures.

- Parallelization is uneffective if source input is small, because time to access the cluster becomes prevalent.
- Trivially, there is no reason of using the parallelization module if just one processor is required.
- Parallelization is beneficial if more instances of `Moses` are exploited.
- The gain in time is not exactly proportional to the number of decoder instances, mainly due to the effect of ending delay.

In conclusion, the choice of the number of splits N is essential for a good efficiency of the parallelization module, and depends on the available computational power, the cluster load, and the average translation time of the standalone decoder.

4.4 Tuning

As described in Section (reference to the correct section), `Moses` decoder relies on a log-linear model to search for the best translation $\mathbf{e}^*$ given an

input string $\mathbf{f}$:

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} \Pr(\mathbf{e} \mid \mathbf{f}) = \arg\max_{\mathbf{e}} p_\lambda(\mathbf{e} \mid \mathbf{f}) = \arg\max_{\mathbf{e}} \sum_i \lambda_i h_i(\mathbf{e}, \mathbf{f}) \quad (4.1)$$

Main components of a log-linear model are the real-valued feature functions h_i and their real-valued weights λ_i. To get the best performance from this model all components need to be estimated and optimized for the specific task the model is applied to.

Feature functions model specific aspects of the translation process, like the fluency, the adequacy, the reordering. Features can correspond to any function of $\mathbf{e}$ and $\mathbf{f}$, and there is no restriction about the values they assume. Some features are based on statistical models which are estimated on specific training data.

Feature weights are useful to balance the (possibly very different) ranges of the feature functions, and to weigh their relative relevance. The most common way to estimate the weights of a log-linear model is called Minimum Error Rate Training (MERT). It consists in an automatic procedure which search for the weights minimizing translation errors on a development set.

Let $\mathbf{f}$ be a source sentence and $\mathbf{ref}$ the set of its reference translations; let $\mathbf{Err}(\mathbf{e}; \mathbf{ref})$ be an error function which measures the quality of a given translation $\mathbf{e}$ with respect to the references $\mathbf{ref}$. The MERT paradigm can be formally stated as follows:

$$\mathbf{e}^* = \mathbf{e}^*(\lambda) = \arg\max_{\mathbf{e}} p_\lambda(\mathbf{e} \mid \mathbf{f}) \quad (4.2)$$

$$\lambda^* = \arg\min_{\lambda} \mathbf{Err}(\mathbf{e}^*(\lambda); \mathbf{ref}) \quad (4.3)$$

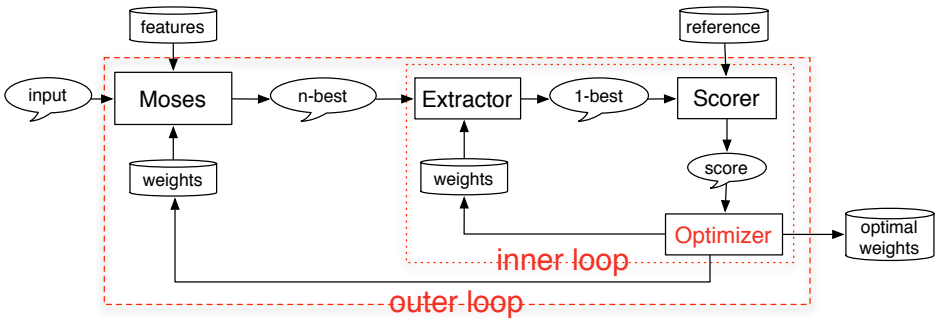
where $\mathbf{e}^*(\lambda)$ is the best translation found by the decoder exploiting a given set of weights λ.

The error function needs to be computed automatically from $\mathbf{e}$ and $\mathbf{ref}$ without human intervention. Word Error Rate (WER), Position Independent Word Error Rate (PER), (100-BLEU score), -NIST score, or any combination of them are good candidates as automatic scoring functions.

An error function is rarely mathematically sound, and hence an exact solution of the previous problem is not usually known. Hence, algorithms like the gradient descent or the downhill simplex, are exploited which iteratively approximate the optimal solution. Unfortunately, these approximate algorithms just assure to find a local optimum.

The MERT procedure we implemented during the workshop is depicted in Figure 4.4. It is based on two nested loops, which are now described.

Figure 4.7:



In the outer loop

1. initial weights λ^0, an empty list of translation hypotheses T^0, and the iteration index $t = 0$ are set;
2. `Moses` translates the input with λ^t and generates a list of N-best translation hypotheses T^t;
3. T^t are added to the previous lists $T^0, \ldots T^{t-1}$;
4. the inner loop is performed (see below) on the new list $\bigcup_{i=0}^{t} T^i$ and with the weights λ^t;
5. the new set of weights λ^{t+1} provided by the inner loop are set;
6. t is increased by 1, and the loop restarts from 2.

The outer loop ends when the list of translation hypotheses does not increase anymore.

In the inner loop which is fed with a list of hypotheses and a set of weights $\bar{\lambda}$

1. initial weights λ^0 are set to $\bar{\lambda}$, and the iteration index $s = 0$ is set;
2. all translation hypotheses in the list are rescored according with the actual weights λ^s and the best-scored hypothesis is extracted (`Extractor`);
3. the error measure of such translation is computed (`Scorer`);
4. the `Optimizer` suggests a new set of weights λ^{s+1};

5. s is increased by 1, and the loop restarts from 2.

The inner loop ends when the error measure does not improve anymore. As the `Optimizer` provides a local optimum for the weights, and strongly depends on the starting point $\bar{\lambda}$, the inner loop starts over several times with different choices of $\bar{\lambda}$. The first time the weights λ^t used by `Moses` in the last outer loop are applied; the next times random sets are exploited. The best set of weights are then provided to the outer loop again.

Instead of standard approximate algorithms like the gradient descent or the downhill simplex, in the workshop we employed an `Optimizer` which was developed by XXXX (should we cite the guy form UMaryland?) and based on the idea of **?** (Och's paper). The algorithm strongly relies on the availability of a finite list of translation alternatives, because this allows a discretization of the r-dimensional space of the weights (r is the number of weights). This makes the search of the optimum faster. The algorithm iteratively optimizes one weight at a time.

The `Scorer` employed in the workshop computes BLEU score (version xxx).

The time spent for each iteration of the outer and inner loops is basically proportional to the size of the input and the amount of translation hypotheses, respectively.

4.5 Efficient Language Model Handling

Marcello Federico

4.6 Lexicalized Reordering Models

Christine Moran

4.7 Error Analysis

We describe some statistics generally used to measure error and present two error analysis tools written over the summer.

4.7.1 Error Measurement

There are three common measures of translation error. BiLingual Evaluation Understudy (BLEU) (**?**), the most common, measures matches of short

phrases between the translated and reference text as well as the difference in the lengths of the reference and output. BLEU can be applied to multiple references, but in a way such that BLEU scores using different numbers of references are not comparable.

Word Error Rate (WER) measures the number of matching output and reference words given that if output word i is noted as matching reference word j, output word $i + 1$ cannot match any reference word before j; i.e., word ordering is preserved in both texts. Such a mapping isn't unique, so WER is specified using the maximum attainable number of single-word matches. This number is computable by some simple dynamic programming. [[[Ought I to elaborate here?]]]

Position-Independent Word Error Rate (PWER) simply counts matching output and reference words regardless of their order in the text. This allows for rearrangement of logical units of text, but allows a system to get away with poor rearrangement of function words.

All these measures are highly dependent on the level of redundancy in the target language: the more reasonable translation options, the less likely the one chosen will match the reference exactly. So the scores we use are really comparable only for a specific source text in a specific language.

Perplexity (defined in **?**), measured for a text with respect to a language model, is a function of the likelihood of that text being produced by repeated application of the model. In a shaky sense, he higher the perplexity of a text, the more complex it is, so the harder it is to produce. The perplexity of the output of a modern machine translation system is usually lower (for our test case, by a factor of two to three) than that of a reliable reference translation. This is unsurprising because the people who provide the references have at their command long-range syntactic constructs that haven't been reconstructed via computer.

Along with these statistics, we'd like some assurance that they're stable, preferably in the form of confidence intervals. We use both the paired t test and the more conservative sign test to obtain confidence intervals for the BLEU score of each translation system on a corpus.

All of these measures can be applied to a text of any size, but the larger the text, the more statistical these scores become. For detail about the kinds of errors a translation system is making, we need sentence-by-sentence error analysis. For this purpose we wrote two graphical tools.

4.7.2 Tools

While working on his thesis Dr. Koehn wrote an online tool that keeps track of a set of corpuses (a corpus is a source text, at least one system output and at least one reference) and generates various statistics each time a corpus is added or changed. Before the workshop, his system showed BLEU scores and allowed a user to view individual sentences (source, output, reference) and score the output. For large numbers of sentences manual scoring isn't a good use of our time; the system was designed for small corpuses. To replace the manual-scoring feature we created a display of the BLEU scores in detail for each sentence: counts and graphical displays of matching n-grams of all sizes used by BLEU. See figure 4.8 for screenshots.

The overall view for a corpus shows a list of files associated with a given corpus: a source text, one or more reference translations, one or more system translations. For the source it gives a count of unknown words in the source text (a measure of difficulty of translation, since we can't possibly correctly translate a word we don't recognize) and the perplexity. For each reference it shows perplexity. For each system output it shows WER and PWER, the difference between WER and PWER two for nouns and adjectives only (**?**), the ratio of PWER of surface forms to PWER of lemmas (**?**), and the results of some simple statistical tests, as described above, for the consistency of BLEU scores in different sections of the text. The system handles missing information decently, and shows the user a message to the effect that some measure is not computable. Also displayed are results of a t test on BLEU scores between each pair of systems' outputs, which give the significance of the difference in BLEU scores of two systems on the same input.

Figure 4.8: Sample output of corpus-statistics tool.

A second tool developed during the workshop shows the mapping of individual source to output phrases (boxes of the same color on the two lines in figure 4.9) and gives the average source phrase length used. This statistic tells us how much use is being made of the translation model's capabilities. There's no need to take the time to tabulate all phrases of length 10, say, in the training source text if we're pretty sure that at translation time no source phrase longer than 4 words will be chosen.

Figure 4.9: Sample output of phrase-detail tool.

Chapter 5

Experiments

5.1 English-German

Philipp Koehn, Chris Callison-Burch, Chris Dyer

5.2 English-Spanish

Wade Shen, Brooke Cowan, Christine Moran

5.3 English-Czech

This report describes in detail our experiments on Czech↔English translation with the Moses system **?** carried out at Johns Hopkins University Summer Workshop 2006 in Baltimore. The reader is expected to be familiar with factored translation models as implemented in Moses.

Section 5.3.1 describes the data used for our experiments, including preprocessing steps and some basic statistics. Section 5.3.2 introduces the metric and lists some known result on MT quality on our dataset, including the scores of human translation. The core of this report is contained in Section 5.3.3 where all our experiments and results are described in detail.

5.3.1 Data Used

Corpus Description and Preprocessing

The data used for Czech↔English experiments are available as CzEng 0.5 **?** and PCEDT 1.0 **?**. The collection contains parallel texts from various domains, as summarized in Table 5.1.

	Sentences		Tokens	
	Czech	English	Czech	English
Texts from European Parliament	77.7%	71.7%	78.2%	75.9%
E-Books	5.5%	6.6%	7.2%	7.4%
KDE (open-source documentation)	6.9%	10.2%	2.6%	3.6%
PCEDT-WSJ (Wall Street Journal texts)	1.5%	1.7%	2.6%	2.8%
Reader's Digest stories	8.4%	9.7%	9.4%	10.3%
Total	1.4 M	1.3 M	19.0 M	21.6 M

Table 5.1: Domains of texts contained in full training data.

The texts in CzEng are pre-tokenized and pre-segmented (sentence boundaries identified) and automatically sentence-aligned using the hunalign tool **?**. The PCEDT data are manually aligned 1-1 by origin, because the Czech version of the text was obtained by translating English text sentence by sentence.

For the purposes of our experiments, we processed the data using the tools listed in Table 5.2. The English side of the corpus had to be retokenized (keeping CzEng sentence boundaries), because the original tokenization was not compatible with the tagging tool.

	Czech	English
Segmentation	CzEng	CzEng
Tokenization	CzEng	Like Europarl, **?**
Morph./POS Tagging	**?**	**?**
Lemmatization	**?**	-not used-
Parsing	**?**	-not used-

Table 5.2: Czech and English tools used to annotate CzEng data.

Baseline (PCEDT) and Large (CzEng+PCEDT) Corpus Data

The evaluation set of sentences used in our experiments (see section 5.3.1 below) comes from the very specific domain of Wall Street Journal. The PCEDT-WSJ section matches this domain exactly, so we use the PCEDT-WSJ section (20k sentences) as the training data in most of our experiments and refer to it by the term baseline corpus or simply PCEDT. In some experiments, we make use of all the training data (860k sentences) and refer to it as the large corpus. (Of course, test data are always excluded from training.)

Corpus		Sentences	Tokens
Baseline: PCEDT	Czech	20,581	453,050
	English	20,581	474,336
Large: CzEng+PCEDT	Czech	862,398	10,657,552
	English	862,398	12,001,772

Table 5.3: Data sizes available for our experiments.

Table 5.3 reports exact data sizes of the baseline and large corpora used for our experiments. (Note that the baseline corpus is a subset of the large corpus.) The data size is significantly lower than what CzEng offers, because not all of the sentences successfully passed through all our tools and also due to the removal of sentences longer than 50 words and sentences with the ratio between Czech and English number of tokens worse than 9.

Tuning and Evaluation Data

Our tuning and evaluation data consist of 515 sentences with 4 reference translations. The dataset was first published as part of PCEDT 1.0 for evaluating Czech→English translation and included original English Wall Street Journal (WSJ) texts translated to Czech (sentence by sentence) and 4 independent back-translations to English. For the purposes of English→Czech translation in our experiments, another 4 independent translations from the original English to Czech were obtained.

For our experiments we kept the original division of the dataset into two parts: the tuning (development) set and the evaluation test set. However, we retokenized all the sentences with the Europarl tokenization tool. Dataset sizes in terms of number of sentences, input tokens and input tokens never seen in the PCEDT training corpus (out-of-vocabulary, OOV) are listed in Table 5.4.

		Input Tokens When Translating from			
	Sentences	Czech	OOV	English	OOV
Tuning	259	6429	6.8%	6675	3.5%
Evaluation	256	5980	6.9%	6273	3.8%

Table 5.4: Tuning and evaluation data.

We followed the common procedure to use tuning dataset to set param-

eters of the translation system and to use the evaluation dataset for final translation quality estimate. In other words, the translation system has access to the reference translations of the tuning dataset but never has access to the reference translations of the evaluation dataset.

In the following, we use the this short notation: Dev (std) denotes results obtained on the tuning dataset with the model parameters set to the default, somewhat even distribution. Dev (opt) denotes results on the tuning dataset with the parameters optimized using minimum error rate training procedure (MERT, XXX). The Dev (opt) results are always overly optimistic, because MERT had access to the reference translations and tunes the MT output to get the highest scores possible. Test (opt) denotes results on evaluation set with model parameters as optimized on the tuning set. The Test (opt) results thus estimate the system performance on unseen data and allow for a fair comparison.

For the purposes of automatic translation, the input texts were analyzed using the same tools as listed in section 5.3.1.

5.3.2 MT Quality Metric and Known Baselines

Throughout all our experiments, we use the BLEU metric **?** to automatically assess the quality of translation. We use an implementation of this metric provided for the workshop. Other implementations such as IBM original or NIST official `mt_eval` might give slightly different absolute results, mainly due to different tokenization rules.

In all experiments reported below, we train and test the system in *case-insensitive* fashion (all data are converted to lowercase, including the reference translations), except where stated otherwise.

Human Cross-Evaluation

Table 5.5 displays the scores if we evaluate one human translation against 4 other human translations. For the sake of completeness, we report not only the default lowercase (LC) evaluation but also case sensitive (CSens) evaluation. This estimate cannot be understood as any kind of a bound or limit on MT output scores, but it nevertheless gives some vague orientation when reading BLEU figures.

As expected, we observe a higher variance (standard deviation) when evaluating translation to English. The reason is that one of the five English versions of the sentences is the original, while the other four were back translated from Czech. It is therefore quite likely for the four back transla-

		To Czech			To English		
		Min	Average	Max	Min	Average	Max
Evaluation	LC	38.5	43.1±4.0	48.4	41.6	54.5±8.4	62.9
	CSens	38.1	42.5±4.0	47.8	41.1	53.8±8.4	62.4
Tuning	LC	39.0	46.3±4.3	49.3	45.8	55.3±6.0	61.7
	CSens	38.3	45.8±4.4	48.8	45.0	54.7±6.1	61.3

Table 5.5: BLEU scores of human translation against 4 different human translations. Evaluated 5 times, always comparing one translation against the 4 remaining. The minimum, average and maximum scores of the 5-fold estimation are given.

tions to differ more from the original than from each other raising the BLEU variance.

English scores are generally higher and this may indicate that there is less variance in word order, lexical selection or word morphology in English, but it also could be the simple case that the translators to English produced more rigid translations.

BLEU When not Translating at All

Our particular type of text (WSJ) contains a lot of numbers and proper names that are often not altered during translation. Just for curiosity and to check that our datasets are not just numbers, punctuation and company names, we evaluate BLEU for texts not translated at all. I.e. the input text is evaluated against the standard 4 references. As displayed in Table 5.6, the scores are very low but nonzero, as expected.

		To Czech	To English
Evaluation	Lowercase	2.20	2.66
Evaluation	Case Sensitive	2.20	2.65
Tuning	Lowercase	2.93	3.60
Tuning	Case Sensitive	2.93	3.59

Table 5.6: BLEU scores when not translating at all, i.e. only punctuation, numbers and some proper names score.

Previous Research Results

Table 5.7 summarizes previously published results of Czech→English translation. Dependency-based MT (DBMT, cmejrek:curin:havelka:2003) is a system with rule-based transfer from Czech deep syntactic trees (obtained automatically using one of two parsers of Czech) to English syntactic trees. GIZA++ **?** and ReWrite **?** is the standard baseline word-based statistical system. PBT **?** is a phrase-based statistical MT system developed at RWTH Aachen that has been evaluated on English-Czech data by bojar:etal:fintal:2006.

	Average over 5 refs.		4 refs. only	
	Dev	Test	Dev	Test
DBMT with parser I, no LM	18.57	16.34	-	-
DBMT with parser II, no LM	19.16	17.05	-	-
GIZA++ & ReWrite, bigger LM	22.22	20.17	-	-
PBT, no additional LM	38.7±1.5	34.8±1.3	36.3	32.5
PBT, bigger LM	41.3±1.2	36.4±1.3	39.7	34.2
PBT, more parallel texts, bigger LM	42.3±1.1	38.1±0.8	41.0	36.8

Table 5.7: Previously published results of Czech→English MT.

All figures in Table 5.7 are based on the same training dataset as we use: the baseline corpus of PCEDT (20k sentences) and on the same tuning and evaluation sets. However, the tokenization of the data is slightly different the we use and also a different implementation of the BLEU metric was used. Our experience is that a different scoring script can change BLEU results by about 2 points absolute, so these numbers should not be directly compared to our results reported here.

Unlike cmejrek:curin:havelka:2003 who evaluate four-reference BLEU five times using the original English text in addition to the 4 available reference back-translations in a leave-one out procedure, we always report BLEU estimated on the 4 reference translations only.

To the best of our knowledge, we are the first to evaluate English→Czech machine translation quality with automatic measures.

5.3.3 Experiments

Motivation: Margin for Improving Morphology

Czech is a Slavonic language with very rich morphology and relatively free word order. (See e.g. bojar:cslp:2004 for more details.) The Czech morphological system defines 4,000 tags in theory and 2,000 were actually seen in a big tagged corpus. (For comparison, the English Penn Treebank tagset contains just about 50 tags.) When translating to Czech, any MT system has to face the richness and generate output words in appropriate forms.

Table 5.8 displays BLEU scores of single-factored translation English→Czech using the baseline corpus only. The second line in the table gives the scores if morphological information was disregarded in the evaluation: the MT output is lemmatized (word forms replaced with their respective base forms) and evaluated against lemmatized references.

	Dev (std)	Dev (opt)	Test (opt)
Regular BLEU, lowercase	25.68	29.24	25.23
Lemmatized MT output against lemmatized references	34.29	38.01	33.63

Table 5.8: Margin in BLEU for improving morphology.

We see that more than 8 point BLEU absolute could be achieved if output word forms were chosen correctly.[1] This observation gives us a strong motivation for focussing on morphological errors first.

Obtaining Reliable Word Alignment

Given the richness of Czech morphological system and quite limited amount of data in the baseline corpus (20k sentences), our first concern was to obtain reliable word alignments. Like bojar:etal:fintal:2006, we reduce the data sparseness by either lemmatizing or stemming Czech tokens and stemming English tokens. (By stemming we mean truncating each word to at most 4 characters.) The vocabulary size of Czech word forms reduces to a half after stemming or lemmatization and comes thus very close to the vocabulary size of English word forms.

[1] Although not all required word forms may be available in the training data, we could easily generate output word forms from lemmas and morphological tags deterministically using a large target-side-only dictionary.

Table 5.9 displays BLEU scores on Test (opt) English→Czech depending on the preprocessing of corpus for word alignment. The translation process itself was performed on full word forms (single-factored), with a single trigram language model collected from the Czech side of the parallel corpus. In all cases, we employed the grow-diag-final heuristic for symmetrization of two independent GIZA++ runs.

Preprocessing for Alignment		Parallel Corpus Used	
English	Czech	Baseline (20k sents.)	Large (860k sents.)
word forms	word forms	25.17	-
4-char stems	lemmas	25.23	25.40
4-char stems	4-char stems	25.82	24.99

Table 5.9: BLEU in English→Czech translation depending on corpus preprocessing for word alignment.

The results confirm improvement in translation quality if we address the data sparseness problem for alignments either by full lemmatization or by simple stemming. Surprisingly, using full lemmatization of the Czech side scored worse than just stemming Czech. This result was confirmed neither on the large training set, nor by bojar:etal:fintal:2006 for Czech→English direction, so we attribute this effect to random fluctuations in MERT procedure.

We also see nearly no gain or even some loss by increasing the corpus size from 20k to 860k sentences. (See section 5.3.3 below for more details on various ways of using more data.) This observation can be explained by the very specific domain of our test set, see section

Scenarios of Factored Translation English→Czech

Scenarios Used

We experimented with the following factored translation scenarios:

Single-factored scenario (T). [fr]

English			Czech		
lowercase	eAlc		cAlc	lowercase	cAlcou
morphologye	Atag		cAlemmalemma	cAlemmaout	
			cAtag	morphology	cAtagou

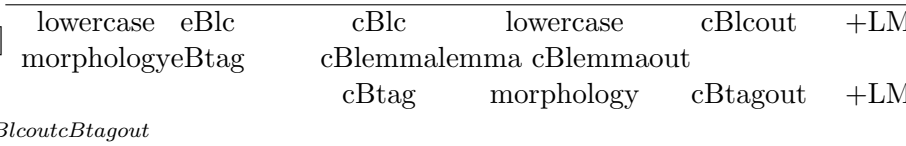
nodesep=1pt,arrows=-¿ *eAlccAlc*

The baseline scenario is single-factored: input (English) lowercase word forms are directly translated to target (Czech) lowercase forms. A 3-gram

language model (or more models based on various corpora) checks the stream of output word forms.

We call this the T (translation) scenario.

Checking morphology (T+C). [fr]



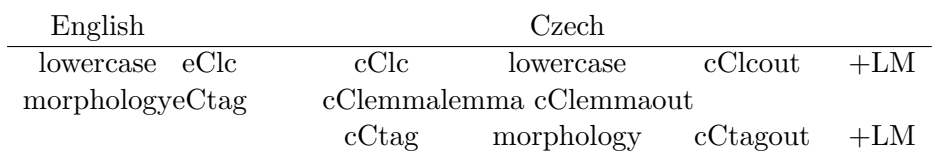
nodesep=1pt,arrows=-¿ *eBlccBlc−>cBlcoutcBtagout*

In order to check the output not only for word-level coherence but also for morphological coherence, we add a single generation step: input word forms are first translated to output word forms and each output word form then generates its morphological tag.

Two types of language models can be used simultaneously: a (3-gram) LM over word forms and a LM over morphological tags. For the morphological tags, a higher-order LM can be used, such as 7 or 9-gram.

We used tags with various levels of detail, see section 5.3.3. We call this the T+C (translate and check) scenario.

Translating and checking morphology (T+T+C). [fr]

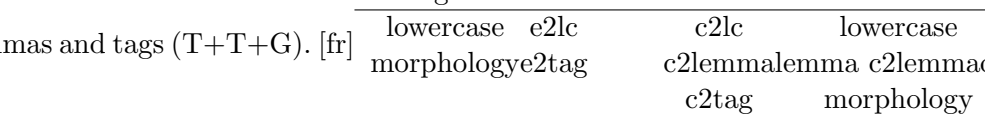


nodesep=1pt,arrows=-¿ *eClccClceCtagcCtag−>cClcoutcCtagout*

As a refinement of T+C, we also used T+T+C scenario, where the morphological output stream is constructed based on both, output word forms and input morphology. This setting should ensure correct translation of morphological features such as number of source noun phrases.

Again, two types of language models can be used in this T+T+C scenario.

Generating forms from lemmas and tags (T+T+G). [fr]



nodesep=1pt,arrows=-¿ *e2tagc2tage2lcc2lemma−>c2lemmaoutc2lcout−>c2tagoutc2lcout*

The most complex scenario we used is linguistically appealing: output lemmas (base forms) and morphological tags are generated from input in two independent translation steps and combined in a single generation step to produce output word forms. The input English text was not lemmatized so we used English word forms as the source for producing Czech lemmas.

The T+T+G setting allows us to use up to three types of language

models. Trigram models are used for word forms and lemmas and 7 or 9-gram language models are used over tags.

Experimental Results: T+C Works Best

Table 5.10 summarizes estimated translation quality of the various scenarios. In all experiments, only the baseline corpus of 20k sentences was used with word alignment obtained using grow-diag-final heuristic on stemmed English and lemmatized Czech input streams. Language models are also based on the 20k sentences only, 3-grams are used for word forms and lemmas, 7-grams for morphological tags.

	Dev (std)	Dev (opt)	Test (opt)
Baseline: T	25.68	29.24	25.23
T+T+G	23.93	30.34	25.94
T+T+C	25.12	30.73	26.43
T+C	23.51	**30.88**	**27.23**

Table 5.10: BLEU scores of various translation scenarios.

The good news is that multi-factored models always outperform the baseline T (except for Dev (std), but this is not surprising, as the default weights can be quite bad for multi-factored translation).

Unfortunately, the more complex a multi-factored scenario is, the worse the results are. Our belief is that this effect is caused by search errors: with multi-factored models, more hypotheses get similar scores and future costs of partial hypotheses might be estimated less reliably. With the limited stack size (not more than 200 hypotheses of the same number of covered input words), the decoder may more often find sub-optimal solutions.

To conclude, the scenario for just checking output morphology (T+C) gives us the best results, 27.23 BLEU, 2 points absolute improvement over the single-factored baseline.

Granularity of Czech Part-of-Speech

As stated above, Czech morphological tag system is very complex, in theory up to 4,000 different tags are possible. In our T+C scenario, we experiment with various simplifications of the system to find the best balance between expresivity and richness of the statistics available in our corpus. (The more information is retained in the tags, the more severe data sparseness is.)

Full tags (1098 unique seen in baseline corpus): Full Czech positional

tags are used. A tag consists of 15 positions, each holding the value of a morphological property (e.g. number, case or gender).

POS (173 unique seen): We simplify the tag to include only part and subpart of speech (distinguishes also partially e.g. verb tenses). For nouns, pronouns, adjectives and prepositions[2], also the case is included.

CNG01 (571 unique seen): CNG01 refines POS. For nouns, pronouns and adjectives we include not only the case but also number and gender.

CNG02 (707 unique seen): Tag for punctuation is refined: lemma of the punctuation symbol is taken into account; previous models disregarded e.g. the distributional differences between a comma and a question mark. Case, number and gender added to nouns, pronouns, adjectives, prepositions, but also to verbs and numerals (where applicable).

CNG03 (899 unique seen): Highly optimized tagset:

- Tags for nouns, adjectives, pronouns and numerals describe the case, number and gender; the Czech reflexive pronoun se or si is highlighted by a special flag.
- Tag for verbs describes subpart of speech, number, gender, tense and aspect; the tag includes a special flag if the verb was the auxiliary verb být (to be) in any of its forms.
- Tag for prepositions includes the case and also the lemma of the preposition.
- Lemma included for punctuation, particles and interjections.
- Tag for numbers describes the shape of the number (all digits are replaced by the digit 5 but number-internal punctuation is kept intact). The tag thus distinguishes between 4- or 5-digit numbers or the precision of floating point numbers.
- Part of speech and subpart of speech for all other words.

Experimental Results: CNG03 Best

Table 5.11 summarizes the results of T+C scenario with varying detail in morphological tag. All the results were obtained using only the baseline

[2]Some Czech prepositions select for a particular case, some are ambiguous. Although the case is never shown on surface of the preposition, the tagset includes this information and Czech taggers are able to infer the case.

corpus of 20k sentences, word-alignment symmetrized with grow-diag-final heuristic and based on stemmed Czech and English input. Also the language models are based solely on the 20k sentences. Trigrams are used for word forms and 7-grams for tags.

	Dev (std)	Dev (opt)	Test (opt)
Baseline: T (single-factor)	26.52	28.77	25.82
T+C, CNG01	22.30	29.86	26.14
T+C, POS	21.77	30.27	26.57
T+C, full tags	22.56	29.65	27.04
T+C, CNG02	23.17	**30.77**	27.45
T+C, CNG03	23.27	30.75	**27.62**

Table 5.11: BLEU scores of various granularities of morphological tags in T+C scenario.

Our results confirm significant improvement over single-factored baseline. Detailed knowledge of the morphological systems also proves its utility: by choosing the most relevant features of tags and lemmas but avoiding sparseness, we can improve about 0.5 BLEU absolute over T+C with full tags. Too strict reduction of features used causes a loss.

More Out-of-Domain Data in T and T+C Scenarios

Figure 5.1 gives a chart and full details on our experiments with adding more data into the T and T+C scenarios. We varied the scenario (T or T+C), the level of detail in the T+C scenario (full tags vs. CNG03), the size of the parallel corpus used to extract phrases (Baseline vs. Large, as described in section 5.3.1) and the size or combination of target side language models (a single LM based on the Baseline or Large corpus, or both of them with separate weights set in the MERT training).

Several observations can be made:

- Ignoring the domain difference and using only the single Large language model (denoted mix in the chart) hurts. Only the T+C CNG03 scenario does not confirm this observation and we believe this can be attributed to some randomness in MERT training of T+C CNG03 s^+.
- CNG03 outperforms full tags only in small data setting, with large data (treating the domain difference properly), full tags are significantly better.

xunit=25mm,yunit=10mm (23.5,-0.7)(28.5,3)
(24.99,0.5)2pt [u](24.99,0.5)mix (25.82,0.5)2pt [u](25.82,0.5)s (26.54,1.5)2pt
[u](26.54,1.5)mix (27.04,1.5)2pt [d](27.04,1.5)s (27.15,1.5)2pt
[ur](27.15,1.5)s^+ (27.15,2.5)2pt [u](27.15,2.5)s^+ (27.29,2.5)2pt
[d](27.29,2.5)mix (27.41,0.5)2pt [u](27.41,0.5)L (27.48,2.5)2pt
[u](27.48,2.5)L (27.62,2.5)2pt [d](27.62,2.5)s (28.12,1.5)2pt [u](28.12,1.5)L
(23,1)(29,1) (23,2)(29,2)
[l](23.5,0.5)T [l](23.5,1.5)T+C full tags [l](23.5,2.5)T+C CNG03
[Ox=23,Dx=1,Dy=5](23,0)(29,3)

s	small data, 20k sentences in the domain
s^+	small data, 20k sentences in the domain with an additional separate LM (860k sents out of the domain)
L	large data, 860k sentences, separate in-domain and out-of-domain LMs
mix	large data, 860k sentences, a single LM mixes the domains

Scenario	Acronym	Parallel Corpus	Language Models	Dev (std)	Dev (opt)	Test (opt)
T	mix	Large (860k)	Large (860k)	23.47	28.74	24.99
T	s	Baseline (20k)	Baseline (20k)	26.52	28.77	25.82
T+C full tags	mix	Large (860k)	Large (860k)	15.66	29.50	26.54
T+C full tags	s	Baseline (20k)	Baseline (20k)	22.56	29.65	27.04
T+C full tags	s^+	Baseline (20k)	20k+860k	19.97	30.33	27.15
T+C CNG03	s^+	Baseline (20k)	20k+860k	19.95	30.48	27.15
T+C CNG03	mix	Large (860k)	Large (860k)	15.77	30.71	27.29
T	L	Large (860k)	20k+860k	19.45	29.42	27.41
T+C CNG03	L	Large (860k)	20k+860k	14.22	30.33	27.48
T+C CNG03	s	Baseline (20k)	Baseline (20k)	23.27	30.75	27.62
T+C full tags	L	Large (860k)	20k+860k	14.06	30.64	**28.12**

Figure 5.1: The effect of additional data in T and T+C scenarios.

- The improvement of T+C over T decreases if we use more data.

First Experiments with Verb Frames

Microstudy: Current MT Errors

The previous sections described significant improvements gained on small data sets when checking morphological agreement or adding more data (BLEU raised from 25.82% to 27.62% or up to 28.12% with additional out-of-domain parallel data). However, the best result achieved is still far below the margin of lemmatized BLEU, as estimated in section 5.3.3. In fact, lemmatized BLEU of our best result is yet a bit higher (35%), indicating that T+C improve not only morphology, but also word order or lexical selection

An adjective in MT output...	**Portion**
agrees with the governing noun	**74%**
depends on a verb (cannot check the agreement)	7%
misses the governing noun (cannot check the agreement)	7%
should not occur in MT output	5%
has the governing noun not translated (cannot check the agreement)	5%
mismatches with the governing noun	**2%**

Table 5.12: Microstudy: adjectives in English→Czech MT output.

Translation of	**Verb**	**Modifier**
... preserves meaning	56%	79%
... is disrupted	14%	12%
... is missing	27%	1%
... is unknown (not translated)	0%	5%

Table 5.13: Analysis of 77 Verb-Modifier pairs in 15 sample sentences.

issues.

We performed a microstudy on local agreement between adjectives and their governing nouns. Altogether 74% of adjectives agreed with the governing noun and only 2% of adjectives did not agree; the full listing is given in Table 5.12.

Local agreement thus seems to be relatively correct. In order to find the source of the remaining morphological errors, we performed another microstudy of current best MT output (BLEU 28.12%) using an intuitive metric. We checked whether Verb-Modifier relations are properly preserved during the translation of 15 sample sentences.

The *source* text of the sample sentences contained 77 Verb-Modifier pairs. Table 5.13 lists our observations on the two members in each Verb-Modifier pair. We see that only 43% of verbs are translated correctly and 79% of nouns are translated correctly. The system tends to skip verbs quite often (21% of cases).

More importantly, our analysis has shown that even in cases where both the Verb and the Modifier are correct, the relation between them in Czech is either non-gramatical or meaning-disturbing in 56% of these cases. Commented samples of such errors are given in Figure 5.2. The first sample shows that a strong language model can lead to the choice

Input:	Keep on investing.
MT output:	Pokračovalo investování. (grammar correct here!)
Gloss:	Continued investing. (Meaning: The investing continued.)
Correct:	Pokračujte v investování.

Input:	brokerage firms rushed out ads ...			
MT Output:	brokerské	firmy	vyběhl	reklamy
Gloss:	brokerage	$\text{firms}_{pl.fem}$	$\text{ran}_{sg.masc}$	$\text{ads}^{pl.voc,sg.gen}_{pl.nom,pl.acc}$
Correct option 1:	brokerské	firmy	vyběhly	s $\text{reklamami}_{pl.instr}$
Correct option 2:	brokerské	firmy	vydaly	$\text{reklamy}_{pl.acc}$

Figure 5.2: Two sample errors in translating Verb-Modifier relation from English to Czech.

of a grammatical relation that nevertheless does not convey the original meaning. The second sample illustrates a situation where two correct options are available but the system chooses an inappropriate relation, most probably because of backing off to a generic pattern verb-$\text{noun}^{accusative}_{plural}$. This pattern is quite common for for expressing the object role of many verbs (such as vydat, see Correct option 2 in Figure 5.2), but does not fit well with the verb vyběhnout. While the target-side data may be rich enough to learn the generalization vyběhnout–s–*instr*, no such generalization is possible with language models over word forms or morphological tags only. The target side data will be hardly ever rich enough to learn this particular structure in all correct morphological and lexical variants: vyběhl–s–reklamou, vyběhla–s–reklamami, vyběhl–s–prohlášením, vyběhli–s–oznámením, ... We would need a mixed model that combines verb lemmas, prepositions and case information to properly capture the relations.

To sum up, the analysis has revealed that in our best MT output:

- noun-adjective agreement is already quite fine,
- verbs are often missing,
- verb-modifier relations are often malformed.

Design of Verb Frame Factor

In this section we describe a model that combines verb lemmas, prepositions and noun cases to improve the verb-modifier relations on the target side and possibly to favour keeping verbs in MT output. The model

is incorporated to our MT system in the most simple fashion: we simply create an additional output factor to explicitly model target verb valency/subcategorization, i.e. to mark verbs and their modifiers in the output. An independent language model is used to ensure coherence in the verb frame factor.

Figure 5.3 illustrates the process of converting a Czech sentence to the corresponding verb frame factor. We make use of the dependency analysis of the sentence and associate each input word with a token:

- tokens for verbs have the form H:Vsubpart of speech:verb lemma indicating that the verb is the head of the frame,
- tokens for words depending on a verb have the form M:slot description to denote verb frame members. Slot description is based on the respective modifier:
  - slot description for nouns, pronouns and (nominalized) adjectives contains only the case information (e.g. subst_{nom} for nouns or pronouns in nominative),
  - slot description for prepositions contains the preposition lemma and the case (e.g. prep:na_{acc} for the preposition na in accusative),
  - sub-ordinating conjunctions are represented by their lemma (e.g. subord:zda for the conjunction zda),
  - co-ordinating conjunctions are treated in an oversimplified manner, the slot description just notes that there was a co-ordinating conjunction. No information is propagated from the co-ordinated elements.
  - adverbs are completely ignored, i.e. get a dummy token —
- punctuation symbols have the form PUNCT:punctuation symbol and conjunctions have the form DELIM:subpart of speech:conjunction lemma to keep track of structural delimiters in the verb frame factor,
- all other words get a dummy token —.

Thus for the beginning of the sample sentence in Figure 5.3 Poptávka trvale stoupá za podpory we create the following stream:

$$\text{M:subst}_{nom} \text{ — H:VB:stoupat M:prep:za}_{gen} \text{ —}$$

The stream indicates that the verb stoupat tends to be modified by a (preceding) subject and (following) argument or adjunct governed by the preposition za in genitive.

Keeping in mind the valency theory for Czech (e.g. panevova:94), there are several limitations in our model:

- We do not make any distinctions between argument and adjuncts (except for the above mentioned deletion of adverbs). Ideally, all adjuncts would get the dummy token —.
- In theory, the order of verb frame members is not grammatically significant for some languages, so we should allow independent reordering of the verb frame factor.
- If a verb depends on a verb, their modifiers can be nearly arbitrarily mixed within the clause (in Czech). Our model does not distinguish which modifiers belong to which of the verbs.

Another problem with the verb frame factor is the explicit representation of the number of intervening words (tokens —). A skipping language model would be necessary to describe the linguistic reality more adequately.

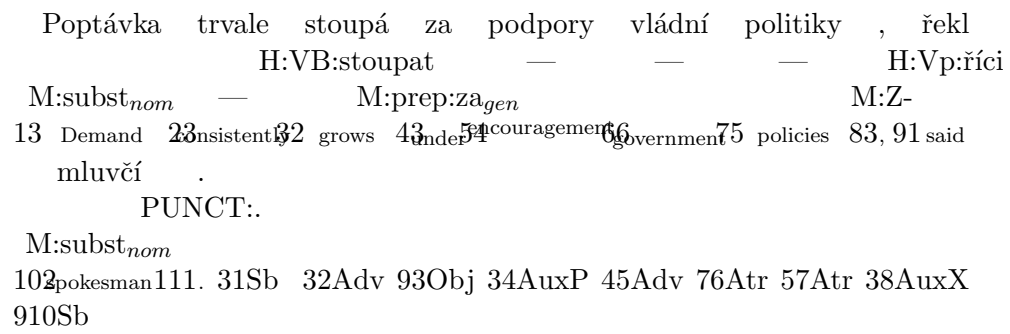


Figure 5.3: Verb frame factor based on dependency syntax tree of a sample sentence: Demand has been growing consistently under the encouragement of government policies, a spokesman said

Preliminary Results with Verb Frame Factor

Table 5.14 displays BLEU scores of the scenario translate-and-check verb factor (T+Cvf) compared to the single-factored baseline (T). Word alignment for these experiments was obtained using grow-diag-final heuristic on stemmed English and lemmatized Czech texts. Only the baseline corpus

(20k sentences) was used to extract phrase tables. The verb frame factor language model is a simple n-gram LM with n of 7, 9 or 11 and it is based either on the baseline corpus (PCEDT) or the Czech side of the Large corpus. In all cases, a simple trigram model checks the fluency of word form stream.

	BLEU.dev.opt	BLEU.dev.std	BLEU.opt
T+Cvf LM-11gr-Large	28.68	19.51	24.23
T+Cvf LM-7gr-Baseline	28.54	19.75	25.05
T+Cvf LM-7gr-Large	28.32	19.69	25.07
T+Cvf LM-9gr-Large	27.98	19.55	25.09
Baseline: T	29.24	25.68	**25.23**

Table 5.14: Preliminary results with checking of verb frame factor.

Unfortunately, all T+Cvf results fall below the single-factored baseline. A more thorough analysis of the data available and the hypothesis space would be necessary to explain why our verb frame factor tends to mislead the decoder.

XXX why did it fail? I'm going to check if at least the verb-mod-relations get improved.

Single-factored Results Czech→English

Our primary interest was in English→Czech translation but we also experimented with Czech→English direction, mainly to allow for comparison with previous reported results.

It should be noted that translating to English in our setting is easier. In general, there are fewer word forms in English so language models face milder data spareness and there are fewer chances to make an error (BLEU would notice). Moreover, the particular test set we use contains input Czech text that came from an English original and was translated sentence by sentence. The Czech thus probably does not exhibit full richness and complexity of word order and language constructions and is easier to translate back to English than a generic Czech text would be.

Table 5.15 lists Moses results of Czech→English translation. We observe a minor improvement when checking the part-of-speech factor (T+C). A larger improvement is obtained by adding more data and quite differently from English→Czech results (see section 5.3.3), mixing in-domain and out-of-domain LM data does not hurt the performance.

Scenario	Parallel Corpus	Language Models	Dev (std)	Dev (opt)	Test (opt)
T	Baseline (20k)	Baseline (20k)	28.97	35.39	28.50
T+C	Baseline (20k)	Baseline (20k)	23.07	36.13	28.66
T	Large (860k)	20k+860k	19.31	39.60	33.37
T	Large (860k)	Large (860k, i.e. mix)	28.94	40.15	**34.12**

Table 5.15: Sample Czech→English BLEU scores.

Summary and Conclusion

We experimented with factored English→Czech translation. The majority of our experiments were carried out in a small data setting and we translated to a morphologically rich language. In this setting, lemmatization or stemming of training data is vital for obtaining reliable alignments. Multi-factored translation for ensuring coherence of morphological properties of output words significantly increases BLEU performance, although the effect is reduced with additional training data. Experiments also indicate that more complex translation scenarios lower the scores, probably due to more severe search errors.

Our English→Czech experiments confirm that in minimum-error-rate training, it is helpful to keep language models based on in- and out-of-domain data separate. We did not observe this domain sensitivity in Czech→English direction.

Based on manual analysis of sample output sentences, we also conducted some preliminary experiments on using target-side syntactic information in order to improve grammaticality of verb-modifier relations. The results are rather inconclusive and further refinement of the model would be necessary.

5.3.4 Acknowledgement

The work on this project was partially supported by the grants Collegium Informaticum GAČR 201/05/H014, Grant No. 0530118 of the National Science Foundation of the USA, and grants No. ME838 and GA405/06/0589. The translation of reference sentences was supported by the grants NSF 0122466 and MŠMT ČR 1P05ME786. The collection of additional training data was supported by the grant GAUK 351/2005. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the respective grant agencies.

I would like to express my gratitude to all members in our team for

excellent atmosphere and very pleasant collaboration. I also wish to thank specifically to the leader of our team, Philipp Koehn, and the teachers Wade Shen and Marcello Federico for full support and many stimulating comments and suggestions. Finally, I cannot forget to mention the Center for Language and Speech Processing at Johns Hopkins University for making this workshop possible.

5.4 Chinese-English

Wade Shen

5.5 Confusion Network Decoding

Wade Shen and Richard Zens

5.6 Tuning

Nicola Bertoldi

5.7 Linguistic Information for Word Alignment

5.7.1 Word Alignment

If we open a common bilingual dictionary, we may find an entry like

Haus = house, building, home, household

Many words have multiple translations, some of which are more likely than others.

If we had a large collection of German text, paired with its translation into English, we could count how often *Haus* is translated into each of the given choices. We can use the counts to estimate a lexical translation probability distribution

$$t : e|f \rightarrow t(e|f)$$

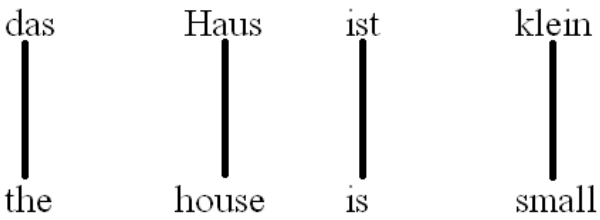
that, given a foreign word, f, returns a probability for each choice for an English translation e, that indicates how likely that translation is.

We can derive an estimate of the translation probability distribution from the data by using the ratio of the counts. For example, if we have 10000 occurrences of $Haus$ and 8000 translate to $house$, then $t(house|Haus) = 0.8$.

For some words that are infrequent in the corpus, the estimates of the probability distribution are not very accurate. Using other linguistic information, such as observing that in a specific language pair verbs usually get translated as verbs, could help in building a more accurate translation.

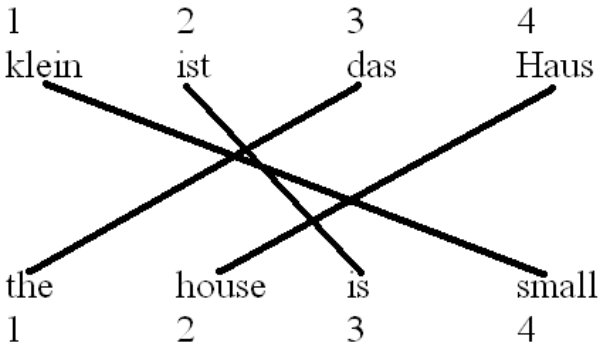
Let's look at an example. Imagine we wanted to translate the German sentence ***das Haus ist klein***. The sentence can be translated word by word into English. One possible translation is ***the house is small***.

Implicit in these translations is an alignment, a mapping from German words to English words:



An alignment can be formalized with an alignment function $a : i \rightarrow j$. This function maps each English target word at position i to a German source word at position j.

For example, if we are given the following pair of sentences:



the alignment function will be

$a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\}$.

5.7.2 IBM Model 1

Lexical translations and the notion of alignment allow us to define a model capable of generating a number of different translations for a sentence, each with a different probability. One such model is IBM Model 1, which will be described below.

For each target word e that is produced by the model from a source word f, we want to factor in the translation probability $t(e|f)$.

The translation probability of a foreign sentence $\mathbf{f} = (f_1, \ldots, f_{l_f})$ of length l_f into an English sentence $\mathbf{e} = (e_1, \ldots, e_{l_e})$ of length l_e with an alignment of each English word e_j to a foreign word f_i according to alignment $a : j \rightarrow i$ is:

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

5.7.3 Learning the Lexical Translation Model

A method for estimating these translation probability distributions from sentence-aligned parallel text is now needed.

The previous section describes a strategy for estimating the lexical translation probability distributions from a word-aligned parallel corpus. However, while large amounts of sentence-aligned parallel texts can be easily collected, word-aligned data cannot. We would thus like to estimate these lexical translation distributions without knowing the actual word alignment, which we consider a hidden variable. To do this, we use the Expectation-Maximization algorithm:

EM algorithm

- Initialize model (typically with uniform distribution)
- Apply the model to the data (expectation step)
- Learn the model from the data (maximization step)
- Iterate steps 2-3 until convergence

First, we initialize the model. Without prior knowledge, uniform probability distributions are a good starting point. In the expectation step, we apply the model to the data and estimate the most likely alignments. In

the maximization step, we learn the model from the data and augment the data with guesses for the gaps.

Expectation step

When we apply the model to the data, we need to compute the probability of different alignments given a sentence pair in the data:

$$p(a|\mathbf{e},\mathbf{f}) = \frac{p(\mathbf{e},a|\mathbf{f})}{p(\mathbf{e}|\mathbf{f})}$$

$p(\mathbf{e}|\mathbf{f})$, the probability of translating sentence $\mathbf{f}$ into sentence $\mathbf{e}$ is derived as:

$$p(\mathbf{e}|\mathbf{f}) = \sum_a p(\mathbf{e},a|\mathbf{f}) = \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)$$

Putting the previous two equations together,

$$p(a|\mathbf{e},\mathbf{f}) = \frac{p(\mathbf{e},a|\mathbf{f})}{p(\mathbf{e}|\mathbf{f})} = \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)}.$$

Maximization Step

For the maximization step, we need to collect counts over all possible alignments, weighted by their probabilities. For this purpose, we define a count function c that collects evidence from a sentence pair $(\mathbf{e},\mathbf{f})$ that a particular source word f translates into a particular target word e.

$$c(e|f;\mathbf{e},\mathbf{f}) = \sum_a p(a|\mathbf{e},\mathbf{f}) = \frac{t(e|f)}{\sum_{j=1}^{l_e} t(e|f_{a(j)})} \sum_{j=1}^{l_e} \delta(e,e_j) \sum_{i=0}^{l_f} \delta(f,f_i)$$

where $\delta(x,y)$ is 1 is $x = y$ and 0 otherwise.

Given the count function, we can estimate the new translation probability distribution by:

$$t(e|f;\mathbf{e},\mathbf{f}) = \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e|f;\mathbf{e},\mathbf{f})}{\sum_f \sum_{(\mathbf{e},\mathbf{f})} c(e|f;\mathbf{e},\mathbf{f})}.$$

5.7.4 Introducing Part of Speech Information to the Model

In order to introduce part of speech information to the model, we need to consider the probability of translating a foreign word f_{word} with part of speech f_{POS} into English word e_{word} with part of speech e_{POS}. In order words, we need to consider the translation probability distribution $t(e|f)$, where e and f are vectors, $e = (e_{word}, e_{POS})$, $f = (f_{word}, f_{POS})$. In order to estimate this density function, we need to make some independence assumptions. Depending on the independence assumption, several models can be formed:

POS Model 1 Assuming that words are independent from their parts of speech, we can estimate the translation density as:

$$t(e|f) = t(e_{word}|f_{word}) * t(e_{POS}|f_{POS})$$

POS Model 2 Making weaker independence assumption, the translation density can be estimated as:

$$t(e|f) = \lambda p(e_{POS}|e_{word})t(e_{word}|f_{word}) + (1-\lambda)p(e_{word}|e_{POS})t(e_{POS}|f_{POS})$$

This model has the advantage that it can weigh the importance given to part-of-speech information.

5.7.5 Experiment

To test whether part-of-speech information improves alignment quality, we compared alignments generated using IBM Model 1, alignments generated using only part-of-speech information, and alignments generated using POS Model 1 against manual alignments. The metric used to compare the alignments was AER (alignment error rate). The data consisted of European Parliament German and English parallel corpora. Experiments were done using different sizes of corpora. The scores are presented in the following table:

AER	*10k*	*20k*	*40k*	*60k*	*80k*	*100k*
IBM Model 1	54.7	51.8	49.3	48.6	47.5	47.1
POS Only	76.0	75.4	75.5	75.1	75.3	75.1
POS Model 1	53.6	51.5	49.6	48.4	47.7	47.3

The first row indicates the number of sentences used for training and the first column indicates the model used to generate alignments.

As expected, the AER of the alignments generated using only part of speech information are very high, indicating that part-of-speech information is not sufficient to generate good alignments. However, an AER of around .75 indicates that there is some information provided by part-of-speech information that could be useful.

The AER of alignments generated with IBM Model 1 doesn't statistically differ from the AER of alignments generated with the additional part of speech information. One reason for this might be that the part-of-speech probability was given equal weight to the word probability, even though the latter is more important. POS Model 2 might thus generate an improvement in AER.

Chapter 6

Conclusions

Philipp Koehn: Accomplishments

Appendix A

Follow-Up Research Proposals

A.1 Translation with syntax and factors: Handling global and local dependencies in SMT

Brooke Cowan: just cut and paste your proposal here

A.2 Exploiting Ambiguous Input in Statistical Machine Translation

Richard Zens: just cut and paste your proposal here