Abstract

Automatic translation from one human language to another using computers, better known as machine translation (MT), is a longstanding goal of computer science. In order to be able to perform such a task, the computer must "know" the two languages—synonyms for words and phrases, grammars of the two languages, and semantic or world knowledge. One way to incorporate such knowledge into a computer is to use bilingual experts to hand-craft the necessary information into the computer program. Another is to let the computer learn some of these things automatically by examining large amounts of parallel text: documents which are nearly exact translations of each other. The Canadian government produces one such resource, for example, in the form of parliamentary proceedings which are recorded in both English and French. Similar resources are becoming available for other language pairs, and it is possible to mine the Internet for bilingual text.

Recently, statistical data analysis has been used to gather MT knowledge automatically from parallel bilingual text. These techniques are extremely promising, as they provide a methodology for addressing the knowledge-acquisition bottleneck that plagues all large-scale natural language processing applications. In the early 1990s, a substantial project by IBM achieved high-quality translation results automatically. In the context of the Johns Hopkins workshop, we describe a methodology for addressing the bilingual text. These techniques are extremely promising, as they provide a methodology for addressing the bilingual text. These techniques are extremely promising, as they provide a methodology for addressing the bilingual text. These techniques are extremely promising, as they provide a methodology for addressing the bilingual text.
The statistical machine translation (SMT) techniques have unfortunately not been applied widely yet in the MT community. The statistical approach is still very much a minority approach in this field. We also had little to expand some narrow.

We largely addressed these goals, as described in this report. We also had little to expand some narrow.

1. Large-scale experiments, the mathematics involved were not particularly familiar to computational linguists at the time they were first published [Brown et al., 1993a]. Another reason is that common software tools and data sets are not generally available. It requires a great deal of work to build the necessary software infrastructure for experimentation in this area.

2. Workshop Goals

At the outset of the workshop, we proposed the goals:

1. Build a statistical machine translation toolkit and make it available to researchers. This toolkit would include corpus preparation software, bilingual-text training software, and run-time decoding software. It would be aimed at two types of users:

   a. Users who say, "I have a parallel corpus, what can I do with it?"

   b. Users who say, "I have a parallel corpus, what can I do with it?"

   c. For performing statistical machine translation, it would be aimed at two types of users:

   i. Users who say, "I have a parallel corpus, what can I do with it?"

   ii. Users who say, "I have a parallel corpus, what can I do with it?"

2. Improve baseline results through the use of morphological and syntactic transducers.

3. Perform baseline evaluations. These evaluations would consist of both objective measures (statistical) and subjective measures (human judgments of quality), as well as attempts to correlate the two.

4. Build a Czech-English machine translation system during the workshop, using the toolkit.

5. Do some initial studies (for example, to observe effects of task-independent statistical measures such as language model perplexity).

6. Test the new idea in the context of an end-to-end machine translation task, for example, by measuring word-error rate on a standard data set, such as Switchboard or CallHome.

7. Improve the idea.

8. Use the idea to design an end-to-end mixed-initiative interface, for example, by using natural language processing models (for example, to observe effects of task-independent statistical measures such as language model perplexity).
3 Background on Statistical Machine Translation

The designer of an SMT system constructs a general model of the translation relation, then lets the system acquire specific rules automatically from bilingual and monolingual text corpora (see Figures 1 and 2). These rules are usually coarse and probabilistic—for example, in a certain corpus, *bat* translates as *palo* 71% of the time, and as *murciélago* 29%. The most-established SMT system is based on word-for-word substitution [Berger et al., 1994], although some experimental SMT systems employ syntactic processing [Wu, 1997; Alshawi et al., 1997; Su et al., 1995]. An advantage of the SMT approach is that designers can improve translation accuracy by modestly changing the underlying model rather than overhauling large handcrafted resources.

SMT views any string of words $e$ as a potential translation of any string $f$. We would like to set up a probability distribution $P(e|f)$ over all pairs of strings, so that given $f$, we can output the $e$ which maximizes $P(e|f)$. We break this problem down into two simpler problems by using Bayes' Rule:

$$P(e|f) \sim P(e) \cdot P(f|e)$$

The $P(e)$ factor helps our $e$ output to be natural and grammatical, while the $P(f|e)$ factor ensures that $e$ is normally interpreted as $f$, and not some other thing.

It is interesting that reasonable translations can come from highly flawed $P(e)$ and $P(f|e)$ estimates. For example, below are several English translations of some French sentence. Translations that pass $P(e)$ are marked with $\times$ in the first column; likewise for translations that pass $P(f|e)$.

| $P(e)$ | $P(f|e)$ |
|-------|--------|
| *Jon appeared in TV.* | $\times$ |
| *Appeared on Jon TV.* |       |
| In *Jon appeared TV.* | $\times$ |
| *Jon is happy today.* | $\times$ |
| *Jon appeared on TV.* | $\times$ $\times$ |
| TV *appeared on Jon.* |       |
| TV in *Jon appeared.* |       |
| *Jon was not happy.* | $\times$ |

Both models are flawed. $P(e)$ assigns too much probability to *TV appeared on Jon*, while $P(f|e)$ assigns too much to *Jon appeared in TV*. However, the translation that they both agree on is very reasonable. A goal of SMT is to find probability estimates that are “good enough” to yield acceptable translations. Once the general models are in place, we need a training algorithm to fix their parameters, and a “decoding” algorithm for finding the $e$ that maximizes the value of the formula above. We discuss these problems in the next section.

4 Core Software Modules Built During the Workshop

Here we describe core software modules for bilingual-text training, translation (“decoding”), corpus preparation, and visualization.

4.1 GIZA

This section describes the part of Egypt which extracts linguistic information from a bilingual corpus. This module is called GIZA, and it is based on the algorithms and translation models described in [Brown et al., 1993a]. We briefly review these models here. Longer reviews are available, and readers looking for more background should consult:

- “A Statistical MT Tutorial Workbook” [Knight, 1999]. A user-friendly description of the translation models, with exercises. Readers who go through this workbook will find it easier to understand papers such as [Brown et al., 1993a].
Carlos Garcia has three associates.

Los asociados no son fuertes.

The modern groups sell strong pharmaceuticals.

Los grupos modernos venden medicinas fuertes.

The groups do not sell zanzanina.

Los grupos no venden zanzanina.

Carlos Garcia has a company also.

Garcia también tiene una empresa.

The company has three groups.

La empresa tiene tres grupos.

The clients are angry.

Los clientes estan enfadados.

The clients and the associates are enemies.

Los clientes y los asociados son enemigos.

The small groups are not modern.

Los grupos pequenos no son modernos.

The modern groups sell strong pharmaceuticals.

Los grupos modernos venden medicinas fuertes.

Figure 1: Bilingual text corpora used to train SMT systems (adapted from Knight, 1997).
Carlos has strong clients. Its small groups have strong enemies. Carlos is angry with his enemies in Europe who also sell pharmaceuticals. The pharmaceuticals are with Carlos and his associates.

Carlos Garcia has three associates. His associates are not strong. Garcia has a company also. Its clients are angry. The associates are also angry. The clients and the associates are enemies. The company has three groups. Its groups are in Europe. The modern groups sell strong pharmaceuticals. The groups do not sell zanzanine.
If any target-language position is over subscribed (contains more than one word), then return failure.

An assignment is deemed equally likely as any other, so any assignment can be carried out with
probabilty 1

Finally, read off the French string.

Only consider the word alignment in which every French word is assigned to exactly one English word.
This is a page from a document discussing the calculation of relative probabilities of word alignments. It involves moving a single link in the alignment of words between languages. The process includes updating probabilities and normalizing them to create new tables. The method used is called hill climbing, which iteratively improves the alignment through small steps until it cannot be improved further. This alignment is called the Viterbi alignment, even though it may not be the most probable alignment. The Viterbi alignment is used to collect counts over the neighborhood of the alignment, and these counts are used to determine alignment weights for collecting relative probabilities of word alignments.
Figure 4: Calculating the relative probability of a word alignment obtained by swapping two links in word alignment.
Where do we start hill climbing? We start by training a simpler model (Model 2) and then using the best alignment it knows about. Model 2 has the same word-translation (t) probabilities, but it does not know about fertilities. Instead of distortion (d) probabilities, Model 2 uses alignment (a) probabilities. The alignment probabilities point "backwards," having the form a(i,j,l,m) instead of d(j,i,l,m).

Model 2 has an efficient EM training algorithm that avoids the necessity of enumerating all alignments:

1. Set t, d, n, and p tables uniformly for several iterations.
2. Set up count tables tc, dc, nc, and pc with zero entries for each sentence pair (e, f) of lengths (l, m)
3. For j = 1 to m
   a. For i = 0 to l
      i. tc(fj | ei) = tc(fj | ei) * a(i | j,l,m) / total
      ii. ac(i | j,l,m) = tc(fj | ei) * a(i | j,l,m) / total

   b. Normalize tc and ac to create new tables t and a

There is also a fast algorithm for computing the best Model 2 alignment between two sentences, used as the base for hill climbing in Model 3. Essentially, each French word chooses an English "source" independently, finding the best one it can. This is difficult to do in Model 3 because one choice may affect another. For example, if every French word chooses English position 5 to connect to, then this will lead to a very large fertility for the English word positioned there.

It is possible to run Model 2 directly on the corpus, but it can be useful to solidify certain word-pair connections by first running yet a simpler model (Model 1) that only has a single table t. The training algorithm for Model 1 looks similar to the one above. We take the t table learned by Model 1, together with a uniform a table, as the starting point for Model 2.

In this section we briefly describe the implementation of the decoder built for the workshop. The job of the decoder is to search for the most probable target (English) sentence given the input source (French) sentence, weighing together the source-to-target translation model and the language model prior on target sentences. It is similar to the decoder used in the Candide system.

Not long after beginning to implement the decoder, it became clear that it was necessary to have working versions of the translation and language models for testing and debugging. Thus, we extended the implementation of Model 1 that had been developed at CMU for information retrieval applications into a full implementation of IBM Models 1/3. The resulting decoder, translation models and interface to the CMU/Cambridge language modeling toolkit now constitute a complete, standalone statistical translation system that we call Wea ver.

When the prototype decoder was completed during the workshop, we were pleasantly surprised with its speed and performance. From the days of DART under translation evaluation conditions, until now.
Figure 5: Transferring parameter values from Model 2 to Model 3.

```
c = 10.0

phi = 10

for k = 1 to phi
  prod = 1.0
  for k = 1 to phi
    prod = prod * alpha[i, k] * times(partition, p, k)
  sum = sum + prod
  if sum > 10.0 then pr
  if sum < 9.9 then pr
  for each partition p in big/gamma/(phi) always enter this loop
  if pij > 0.9 then pij = 0.9
  if pij < 0.1 then pij = 0.1
  for i = 0 to m
    for j = 0 to m
      pij = pij + (1 - pij)
  for k = 0 to m
    alpha[i, k] = (k + 1) / k^k
  for i = 0 to m
    for j = 0 to m
      pij = pij * alpha[i, j] * times(partition, p, j)
  sum = sum + prod
```

Figure 5: Transferring parameter values from Model 2 to Model 3.

```
c = 10.0

phi = 10

for k = 1 to phi
  prod = 1.0
  for k = 1 to phi
    prod = prod * alpha[i, k] * times(partition, p, k)
  sum = sum + prod
  if sum > 10.0 then pr
  if sum < 9.9 then pr
  for each partition p in big/gamma/(phi) always enter this loop
  if pij > 0.9 then pij = 0.9
  if pij < 0.1 then pij = 0.1
  for i = 0 to m
    for j = 0 to m
      pij = pij + (1 - pij)
  for k = 0 to m
    alpha[i, k] = (k + 1) / k^k
  for i = 0 to m
    for j = 0 to m
      pij = pij * alpha[i, j] * times(partition, p, j)
  sum = sum + prod
```

Figure 5: Transferring parameter values from Model 2 to Model 3.

```
c = 10.0

phi = 10

for k = 1 to phi
  prod = 1.0
  for k = 1 to phi
    prod = prod * alpha[i, k] * times(partition, p, k)
  sum = sum + prod
  if sum > 10.0 then pr
  if sum < 9.9 then pr
  for each partition p in big/gamma/(phi) always enter this loop
  if pij > 0.9 then pij = 0.9
  if pij < 0.1 then pij = 0.1
  for i = 0 to m
    for j = 0 to m
      pij = pij + (1 - pij)
  for k = 0 to m
    alpha[i, k] = (k + 1) / k^k
  for i = 0 to m
    for j = 0 to m
      pij = pij * alpha[i, j] * times(partition, p, j)
  sum = sum + prod
```
During the early 1990s, one had come to expect that only short sentences could be decoded in the statistical approach, and that even then the decoding time might be hours per sentence. In the nearly 10 years since the early IBM work, Moore's law, better compilers, and more plentiful memory and disk space have helped build a system that now allows us to decode 25-word sentences in seconds. To be sure, we use fairly tight thresholds and constrains on the search, as described below. But the relative efficiency of the decoder bodes well for future work in this direction, and the original spirit of the IBM work, which emphasized probabilistic modeling over efficiency, arms the original spirit of the IBM work, which emphasizes probabilistic modeling over efficiency.

Our long-term goal in building a decoder is to design a flexible research tool that can be used for exploring the implementation of models 1-3. The implementation of models 1-3 was originally made in the context of applications to information retrieval, and was intended to support large vocabularies and large numbers of parameters. The basic data structures and was intended to support large vocabularies and large numbers of parameters. The basic data structures

The distortion parameters for Model 3 are modified from those used in the Candide system for bootstrapping Models 4 and 5. The original parameters take the form $d_{jijijim}$, where $j$ is an index in the source (French) sentence, $i$ is an index in the target (English) sentence, and $l$ and $m$ are the lengths of the target and source sentences, respectively. The length $m$ of the source is determined once all of the fertilities have been chosen. When decoding Model 3, however, the length is unknown. As a result, our implementation uses the parameters $d_{jijijim}$: Training of Model 3 can be carried out in three different modes: collecting counts in the E-step by summing over the neighbors of the best alignment reachable (by moves and swaps) from the Model 2 Viterbi alignment, or by summing over all "pegged" alignments. In the later case, the alignments can be hashed to avoid collecting counts over the same alignment more than once. Experiments were not completed to compare the results of training in these different ways, but these comparisons should be made in the future.

4.2.2 Organization of the Decoder

The decoder is designed according to the "stack decoding" paradigm. This scheme is similar to a search, but the stack decoding algorithm enforces a discipline and organization on the search that is very useful. The stacks (priority queues) in our implementation are indexed by subsets of the source (French) sentence, corresponding to the source words that have been generated so far in a given hypothesis. Each hypothesis is extended by exactly one source word in each step, and extended by exactly one source word in each step, so each hypothesis is extended by exactly one source word in each step. The stacks (priority queues) in our implementation are indexed by subsets of the source (French) sentence, corresponding to the source words that have been generated so far in a given hypothesis. Each hypothesis is extended by exactly one source word in each step, and extended by exactly one source word in each step, so each hypothesis is extended by exactly one source word in each step. The search (priority queue) is a description and organization on the search that is very useful. The stacks (priority queues) in our implementation are indexed by subsets of the source (French) sentence, corresponding to the source words that have been generated so far in a given hypothesis. Each hypothesis is extended by exactly one source word in each step, and extended by exactly one source word in each step, so each hypothesis is extended by exactly one source word in each step, and extended by exactly one source word in each step, so each hypothesis is extended by exactly one source word in each step.

The basic structure of the decoder is very simple. The interface to the StackDecoder class is shown below:

```cpp
template <class State>
class StackDecoder {
public:
    StackDecoder(int numStacks_, int maxStackSize_ = 10000);
    ~StackDecoder();
    Stack<State> &StackOfState(State &state);
    void RecomputeThresholds();
    int Frontier(List<State> &list);
    int AddState(State /*pState*/);
    int RemoveState(State /*pState*/);
    int UpdateThresholds(List<State> &list);
};
```
4.2.4 Conversion from GIZA to Weaver Format

are precomputed and stored together with the fertility probabilities.

\[
\sum_{n=0}^{\infty} \left( \sum_{\phi} \left( n \phi \right) \right) = \sum_{\phi} \left( \phi \right)
\]

When building up a hypothesis in which a single English word aligns with several French words, we need to factor in the probability that this English word is frequent, and that French words have been assigned to a 

The inverse translation probability of a source word is calculated as

\[
\left( \frac{\rho \rho f}{\rho} \right) \left( \frac{f}{\rho} \right) \left( \frac{\rho}{\rho} \right) = \left( f \rho \right)_{1-1}
\]

When building up a hypothesis in which a single English word aligns with several French words, we need to factor in the probability that this English word is frequent, and that French words have been assigned to a

The inverse translation probability of a source word is calculated as

\[
\left( \frac{\rho \rho f}{\rho} \right) \left( \frac{f}{\rho} \right) \left( \frac{\rho}{\rho} \right) = \left( f \rho \right)_{1-1}
\]

When building up a hypothesis in which a single English word aligns with several French words, we need to factor in the probability that this English word is frequent, and that French words have been assigned to a

The inverse translation probability of a source word is calculated as

\[
\left( \frac{\rho \rho f}{\rho} \right) \left( \frac{f}{\rho} \right) \left( \frac{\rho}{\rho} \right) = \left( f \rho \right)_{1-1}
\]

When building up a hypothesis in which a single English word aligns with several French words, we need to factor in the probability that this English word is frequent, and that French words have been assigned to a

The inverse translation probability of a source word is calculated as

\[
\left( \frac{\rho \rho f}{\rho} \right) \left( \frac{f}{\rho} \right) \left( \frac{\rho}{\rho} \right) = \left( f \rho \right)_{1-1}
\]

When building up a hypothesis in which a single English word aligns with several French words, we need to factor in the probability that this English word is frequent, and that French words have been assigned to a

The inverse translation probability of a source word is calculated as

\[
\left( \frac{\rho \rho f}{\rho} \right) \left( \frac{f}{\rho} \right) \left( \frac{\rho}{\rho} \right) = \left( f \rho \right)_{1-1}
\]
Performance

On one test run of the decoder on Hansard data, sentences of lengths between 5 and 25 words were decoded, and the average decoding time was 1.7 seconds per sentence. In this run, the top 1/2 inverse translations were considered for each source word, and only 8 words according to the language model probabilities \( p(w_2|w_1,w_3) \) were considered as fertility zero extensions. The stack thresholding parameter was set to 0.0:0.01, meaning that those hypotheses with a score smaller than 0.001 times the best scoring hypothesis in a stack were discarded. The complete decoding process was to:

1. Compute the complete set of per-word translations.
2. Use a stack-based approach to select the most probable translations.
3. Expand the stack by considering fertility zero extensions.
4. Stop decoding when the stack is empty or the time limit is reached.

Future Development of the Decoder

Our ultimate goal in building a decoder is to design a flexible research tool for the next generation of statistical machine translation models. Thus, the most immediate development of the decoder will be to abstract away any of the details of Model 3 that remain in the interface, so that general translation models and language models can be "plugged in." Further work needs to be done to improve stack normalization, pruning, and other improvements to existing statistical machine translation models. The next immediate development of the decoder will be to...

Cairo

Cairo is a visualization tool developed for statistical translation models of the sort developed at IBM in the early 1990s. These models, referred to as the "Candide" models, are based on the source-channel paradigm, where one language (the "source" language) is interpreted as an encoded form of another (the "target" language). Translations are produced through a process known as "decooding." A translation model is built using iterative training on sentence-aligned bitext. The translation process is known as "decooding." There are four types of parameters to model unidirectional language translation:

- **Translation**: The probability that an English vocabulary item (type) will translate to a given French type.
- **Fertility**: The probability that a given English type will translate into a given French token.
- **Distortion**: The probability that a given English token in indexed position \( i \) of an English sentence will align with a French token in position \( j \) of a French sentence, given the lengths of the sentences.
- **NULL-insertion**: The probability of insertion of a NULL token at any position in the English sentence during the decoding process (a NULL token is empty in English but is aligned to one or more French tokens).

Cairo was implemented to allow inspection of the bilingual text and different process and the decoding process in statistical machine translation models. It can be implemented to allow inspection of the bilingual text and different models and their parameters, and to support an "in-line" process.

4.3 Cairo

Cairo takes as its input an SGML-style file that specifies the two sentences, their alignment, and all relevant model parameters. The more information given as input, the more powerful the visualization will be; however, no information is required except for the sentences themselves.

In a graphical user interface (GUI), Cairo displays the given sentence pair (assumed to be a translation pair) with lines drawn between aligned words. This representation can be displayed vertically or horizontally, and different colors can be used to highlight different parts of the sentence. Relevant parameters may include the probabilities associated with a word translation, fertility, etc.; these parameters can be modified in real-time as the user experiment.
Each token in a sentence can optionally include multiple, parallel streams of data. One common use by our collaborators is to include part-of-speech tags, lemmas, and/or most-probable translations for each token. The number and names of all streams are provided in the input file by the user. The user has full control over which streams are displayed.

Cairo allows for up to three alignments to be displayed at once for a sentence pair. Visual rendering of multiple alignments is accomplished with different colors, using subtractive color mixing to indicate shared word alignments. When evaluating a machine translation, it is useful to refer to a gold standard, or reference translation and alignment. Cairo provides the simultaneous display of a reference translation alongside the given translation, and allows rapid switching of the display between the machine alignment and the reference alignment.

A Cairo user can mouse-click on an English token and see its model parameters displayed in tables alongside the alignment. In addition, if relevant language model parameters (used in tandem with the translation model in the decoding process) are specified, these will be displayed as well. For example, if the user clicks on an English word “house,” a list of French words to which “house” is likely to translate appears in the translation table, sorted by probability. The French words are also displayed in the translation table. When dealing with models consisting of millions of parameters, actually examining specific alignments can be enlightening. Being able to see potential neighborhood alignments by looking at other model parameters that might have been “used” gives additional insight. This tool allows us and others to track the progress of our translation models and discover models that may perform better.

A window displays information such as the names of the languages the sentences are in, sentence identification numbers, the language model used, etc. Any text given in the input file is displayed in this window.

When evaluating a machine translation, it is possible to modify and extend Cairo for purposes other than statistical machine translation. Cairo has proven extremely useful in statistical machine translation research. When dealing with models consisting of millions of parameters, actually examining specific alignments can be enlightening. Being able to see potential neighborhood alignments by looking at other model parameters that might have been “used” gives additional insight. This tool has allowed us and others to track the progress of our translation models and discover models that may perform better.

4.4. Whittle

Whittle is a corpus preparation tool. It splits the corpus into training and test corpora, discards sentences over a certain length, etc. This tool is a command-line tool. If you specify the command-line arguments and test corpora, you can perform the desired actions.

4.4.2. Implementation and Use

A window displays information such as the names of the languages the sentences are in, sentence identification numbers, the language model used, etc. Any text given in the input file is displayed in this window.

When evaluating a machine translation, it is possible to modify and extend Cairo for purposes other than statistical machine translation. Cairo has proven extremely useful in statistical machine translation research. When dealing with models consisting of millions of parameters, actually examining specific alignments can be enlightening. Being able to see potential neighborhood alignments by looking at other model parameters that might have been “used” gives additional insight. This tool has allowed us and others to track the progress of our translation models and discover models that may perform better.
Learning Curves

During the workshop, we aimed at producing several kinds of informative learning curves.

5.1. Training-Set Size vs. Translation Quality

One question we faced at the workshop was how much data do you need to achieve a good learning curve. If you provide a large training set, then you can expect to achieve a good performance. However, the performance of a machine translation system is not always directly proportional to the size of the training set. Sometimes, there is a point of diminishing returns, where increasing the size of the training set does not lead to a significant improvement in performance.

In the computer science community, it is often assumed that a larger training set will lead to better performance. However, this is not always the case. In some cases, increasing the size of the training set can actually lead to worse performance. This is because the training set may contain noise or errors, which can lead to a model that is not generalizable.

In speech recognition, the size of the training set is often used as a measure of performance. However, in machine translation, this is not always the case. The performance of a machine translation system is not always directly proportional to the size of the training set. Sometimes, there is a point of diminishing returns, where increasing the size of the training set does not lead to a significant improvement in performance.
A language model is without making reference to word-error rate or any other task-level measure. One simply asks for the particular number $P(e)$ that a particular (instantiated) model assigns to a particular text. If the text is good English, we expect $P(e)$ to be high; if the text is bad English, we expect $P(e)$ to be low. If we observe a language model assigning probabilities the other way around, then it probably isn’t a very good language model.

It is reasonable to ask for the $P(e)$ that a model assigns to the text it was trained on. In this case, a memorizing program would do very well, by assigning $P(e) = 1$. However, this program would by definition assign $P(e') = 0$ to every other text $e'$, and this will lead to a very poor word-error rate. Therefore a more reasonable evaluation is test-set perplexity, which is related to the probability that the model assigns to a previously unseen (test) English text. This becomes a gambling game. The language model must lay bets on all kinds of strings. If it concentrates its bets on certain subset of strings, then it must hope that when the previously unseen text is revealed, it is to be found in that subset. Because all probabilities sum to 1, increasing our bet on one string necessarily means decreasing our bet on some other string.

Perplexity is a function of probability. A translation model assigns a value $P(F|E)$ to any pair of sentences $F$ and $E$, and the perplexity over a set of sentences is given by:

$$ 2 \log \left( \frac{1}{N} \sum_{i=1}^{N} P(F_i|E_i) \right) $$

where $N$ is the number of words in the corpus. If we want a high probability, then we want a low perplexity.

Suppose we have a language model (such as learned by GIZA) assigns a probability $P(F|E)$ to every pair of sentences. In this case, our previously unseen test data consists of sentence pairs. Given a certain sentence $E$, a model will lay bets on various sentences $F$'s. When the actual translation is revealed in the test data, we can see whether the model bet a lot or a little. We hope it bet a lot.

Training-set Size vs. Test-set Perplexity

Using perplexity as a substitute for translation quality allows us to produce learning curves such as the one shown in Figure 6, for Czech/English MT. In computing test perplexity, we had to take account of previously unseen words. For a novel English word $e'$, we took $P(e'|F)$ to be uniform across all French words observed in training. For previously seen words, we took $P(e'|F)$ to be equal to the product of conditional probabilities $P(e'|F) = P(e'|F) \times P(e'|F)$, as in one hypothesis.

How is perplexity related to translation quality? This question has no clear answer at present, and is a question that we wanted to investigate in the workshop. Test/set perplexity does not measure how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language. Test-set perplexity does not measure how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previously unseen text to appear during training, but that does not mean how well a system translates into the target language, but rather how well it recognizes whether two sentences are (or are not) translations of one another. If a system had infinite computing power, it is easy to see how it could use good recognition to do good translation. This does not work the other way around. A commercial MT system may produce good translations without good recognition, but it cannot use its recognition to do good translation—its recognition engine has developed its own model of what makes an English sentence grammatical. What we need is a mechanism that allows a previous...
Figure 6: Perplexity as a function of bilingual training set size (Czech/English).
This leads to a second kind of learning curve (see Figure 7), one that plots performance as a function of the number of EM iterations. While training-set perplexity is guaranteed to decrease, test-set perplexity, as would be demonstrated by a program that is able to successively memorize more and more of the training data on each iteration, was much less clear.

Among the workshop participants, there was much disagreement over the value of test-set perplexity as a variable to work with, and so we wanted to do experiments that would quantitatively relate objective perplexity with subjective quality. Our theory was that the baseline Model 1 would begin to fail, unless we trained on larger parallel corpora. Models 1 and 2 made things worse (see Figure 8). Yet we felt that the alignments were getting better.

Against this backdrop, our next step was to oversample training material in Model 2. We generated an additional data point by keeping corpus size constant while comparing the baseline system with our improvements. There was a fair amount of confusion over smoothing, which is a necessary evil in language modeling, but not something IBM ever did in their translation modeling. (It was interesting that before the Egypt toolkit was fully integrated, we felt that the perplexity was getting better.)

Moreover, the perplexities would be good to go with our improvements. There was a fair amount of confusion over smoothing, which is a necessary evil in language modeling, but not something IBM ever did in their translation modeling. (It was interesting that before the Egypt toolkit was fully integrated, we felt that the perplexity was getting better.)

In this case, better alignments led to better explanations of the previous unseen test set.
Figure 7: Perplexity as a function of EM iterations (French/English, 885,393 sentence pairs of training data).
Figure 8: Perplexity as a function of EM iterations (Czech/English, 38,596 sentence pairs of training data).
For the precise sentences to select, two text-dependent measures are significant: how the MT performs on text that is in the context of text on which it has been trained, and how it performs on relatively novel text. For the latter, text out of context, we decided to strip the last 1/50 sentences from the processed corpora, and call this "novel" because the system has not seen these sentences in the neighborhood of other text which is more likely to use the same names, the same events, and so on. One can go further with this concept. For the processing of a number of news stories, one may extract an entire story and define it as novel in relation to the others, or with a number of Reader's Digest stories, one may again use 1/50 sentences from an entire story removed from the corpus. Then, for the remainder of the test text, we choose sentences evenly spaced in the pre-training corpus. So, if there are about 1.5 million sentences in the corpus, then every 100,000th sentence would be chosen for testing; this can be accomplished very easily using Whittle. A nice aspect of this method is that one can use Whittle to create a series of training texts where the in-context factor should be more relevant to the in-context sentences chosen for evaluation. We did not have a chance to evaluate the performance of these sentences, but expect that this approach will be validated.

6 Experiments with Models 3 and 4 Training

In the IBM Models beyond Model 2, efficient training algorithms are difficult yet crucial. During the workshop, we spent quite a bit of effort in this direction:

- The training program was profiled and all time-consuming functions were rewritten and if possible less often used.
- The data structures for lexicon, alignment/dissimilarity and fertility parameters were rewritten for higher efficiency.
- The training of Model 3 was completely reorganized. As Model 3 training is the most time-consuming part of the training procedure, the last point resulted in the biggest improvement.

6.1 Spedding up Model 3 Training

A straightforward common collection procedure for a sentence pair \((f, e)\) following the description in [Brown et al., 1993a] would be:

1. Calculate the Viterbi alignment of Model 2 and \[\pi(f; e; 1) = \alpha(f; 0) = \alpha(i; 1)\] for every alignment in the neighborhood \(\mathcal{N}(\alpha)\).
2. Set \(\pi(f; e; 1) = \alpha(f; 0)\) to the best alignment in the neighborhood.
3. While in the neighborhood \(\mathcal{N}(\pi)\) an alignment \(\alpha\) exists with \(\pi(f; e; 1) < \pi(f; e; 2) < \pi(f; e; 3)\):
   - Set \(\pi(f; e; 1) = \pi(f; e; 2)\).
4. For every alignment \(\alpha\) in the neighborhood \(\mathcal{N}(\alpha)\): Calculate the Viterbi alignment of Model 2 and \(\pi(f; e; 3) = \alpha(f; 0)\).

To simplify the description it is ignored the process called "pegging" which generates a bigger number of alignments considered in learning. In the training program for Model 2, pegging is implemented with minor simplifications.
For the scores of a move/swap of to recalculate only those rows or columns in this matrix which are affected by the move/swap. Applying this caching procedure it is possible to reduce the number of operations in hill-climbing by about one order of magnitude. For example when performing a move men t it is necessary to perform a loop over all English and a loop over all French to increase the alignments in the neighborhood of the last hill-climbing step. Similar rules hold after a swap. In the count-collection (step 3) it is possible to use the matrices obtained in the above mentioned to obtain of an alignment lexicon/distortion and the fertility counts. To increase efficiency we make use of the fact that the alignments make use of the move/swap matrices which are available after performing the above described hill-climbing steps.

The last hill-climbing step.

Similar rules hold after a swap. In the count-collection (step 3) it is possible to use the matrices obtained in

\[ f \neq f' \] if \( f \neq f' \) or \( f' = f \) in \( i \) and in a matrix

\[ f \neq f' \] for the score of a move or a swap.

For reducing the number of calls to these functions the values are cached in a matrix. Profiling the training program reveals that still most of the time is used in scoring a move and scoring a matrix.

Produce hill-climbing.

Swap (see also section 4.1).

This method in a count-collection for calculating the score of a move of a score of a matrix.

This leads to the following formula for the calculation of the probability of an alignment:

\[ \Pr(f \mid i) = (\Pr(T \mid i) \cdot \Pr(f \mid T \mid i)) \]

A major part of the training time in this procedure is consumed by calculating the probability of an alignment:

\[ \Pr(f \mid i) = \Pr(T \mid i) \cdot \Pr(f \mid T \mid i) \]

For every \( f \) do:

\[ d + (\alpha \cdot f) \]

For every \( f \) do:

\[ d + (\alpha \cdot f) \]

Increase the fertility counts with:

\[ d + (\alpha \cdot f) \]

Increase the fertility counts with:

\[ d + (\alpha \cdot f) \]

Increase the fertility counts with:

\[ d \]

Increase the counts for \( d \) and \( \phi \)
For the distortion counts and the translation counts it holds:

\[
(y + \psi \phi) (\omega' j (u)_y N)_d \sum \gamma = (j' \psi \phi) \\
((y + \psi \phi) \cdot \gamma - f) (\omega' j (u)_y N)_d \sum \gamma = (j' \psi \phi) 
\]

For the fertility counts for the empty word holds:

\[
[\phi = y + \psi \phi] \cdot (\omega' j (u)_y N)_d \sum \gamma [\gamma = \gamma] \sum = (j' \psi \phi) 
\]

Fertility counts hold:

These quantities do not depend on swaps so a swap does not change the fertility of an alignment. For the fertility of all alignments which have an undecreased fertility for position i:

\[
\left( \gamma \sum_{\gamma = \gamma} \omega' j u_d = \omega' j (u)_i N_d \right) 
\]

Fertility counts hold:

To get efficiently the fertility probability counts and the counts for \( \mu_{\gamma} \) we introduce the following:

\[
[\gamma = \gamma] \cdot [f = f] \cdot (\omega' j \cdot f) \sum \gamma \sum = (\omega' j \cdot f) 
\]

\[
\left( \gamma \sum_{\gamma = \gamma} \omega' j u_d = \omega' j (u)_i N_d \right) 
\]

Fertility counts hold:

For the distortion probability counts and the translation probability counts holds:

\[
\left( \sum_{\gamma = \gamma} \omega' j u_d \right) \sum \gamma = (\omega' j (u)_i N_d) 
\]

Fertility counts hold:

\[
\left( \sum_{\gamma = \gamma} \omega' j u_d \right) \sum \gamma = (\omega' j (u)_i N_d) 
\]

Fertility counts hold:

\[
\left( \sum_{\gamma = \gamma} \omega' j u_d \right) \sum \gamma = (\omega' j (u)_i N_d) 
\]
In the model of Brown et al., it is assumed that the previous word is chosen from a given set of parameters, and the current word is chosen from a given set of parameters. This model is known as the Hidden Markov Model (HMM). An extension of this model is the Hidden Markov Model with relative distortion probabilities, which introduces a dependence on the previous word order.

The corresponding extension of the distortion probabilities in Model 4 is given by:

$$\Pr(\mathbf{X}_i = \mathbf{X}, \mathbf{Y}_j = \mathbf{Y}, k) = \prod_{m=1}^{l} \Pr(\mathbf{X}_i = \mathbf{X}, \mathbf{Y}_j = \mathbf{Y}, k, i)$$

where \(\mathbf{X}_i, \mathbf{Y}_j\) are the previous and current word, respectively, and \(k\) is the distance between them.

**Simplified Model: Model 4**

The simplified model is obtained by setting the previous word probability to 1 for all words, i.e.,

$$\Pr(\mathbf{X}_i = \mathbf{X}, \mathbf{Y}_j = \mathbf{Y}, k) = \prod_{m=1}^{l} \Pr(\mathbf{X}_i = \mathbf{X}, \mathbf{Y}_j = \mathbf{Y}, k)$$

Motivation

6.2 Working with Relative Distortion Probabilities
According to Model 4, it is not meaningful to restrict the order of the words that are allowed to follow the word "be" with non-zero fertility. The value of \( i = 0 \) is undefined if there is no word before the current word. The value of \( d = 0 \) is also undefined if there is no word before the current word. The value of \( d = 0 \) is undefined if there is no word before the current word. The value of \( d = 0 \) is also undefined if there is no word before the current word. The value of \( d = 0 \) is also undefined if there is no word before the current word.

The training of Model 4 is in the current implementation about 10 times slower than the training of Model 3.

Results

According to Model 4, it is also not meaningful to restrict the order of the words that are allowed to follow the word "be" with non-zero fertility. The value of \( i = 0 \) is undefined if there is no word before the current word. The value of \( d = 0 \) is also undefined if there is no word before the current word. The value of \( d = 0 \) is also undefined if there is no word before the current word. The value of \( d = 0 \) is also undefined if there is no word before the current word. The value of \( d = 0 \) is also undefined if there is no word before the current word.
Figure 1 shows the behavior of the test perplexity. It can be seen that using Model 4 the test perplexity can be reduced significantly. Because of the different kind of deficiencies of Model 4 and Model 3 it is not possible to conclude only from these learning curves the superiority of Model 4. However, by analyzing the resulting Viterbi alignments it seems that Model 4 gives better results.

One of the goals of this workshop was to develop a Czech/English translation system. We present the first experimental results of the Czech/English machine translation based on the probabilistic approach. Our implementation translates models (IBM 3, IBM 4) into the Czech language. The translation system is able to handle one-to-many alignments from English to Czech. Therefore it is useful to have more words in Czech.

The corpus consists of 88,000 English sentences aligned against 85,000 Czech sentences. The corpus was developed in a project coordinated by Charles University in Prague. The corpus consists of articles from Reader's Digest, years 1993-1996. The English part is a translation of the original articles. The corpus has been morphologically analyzed, tagged, lemmatized, and parsed by the tools available for Czech.

The tools available for morphological analysis, POS tagging, and lemmaization are provided by IFAL (Charles University, Prague) and a statistical parser for Czech developed at a previous NLP summer workshop at Johns Hopkins University. The corpus has been processed and converted into a format readable by the tools used in the workshop.

There was also a lot of useful work done on the corpus before the workshop. There were several full-text corpora which are a partial or full text of articles from the readers Digest years 1993-1996. There was also a Czech/English online dictionary available. This dictionary consists of 23,000 entries and covers 90% of the Czech part of the corpus. We also experimented with a technically-oriented Czech/English corpus from IBM. This is a huge and very good source of Czech/English parallel data, but for a very specific domain. This corpus consists of operating system messages and operating system guides. These are products of localization and translation of software from English to Czech. The translations are very literal and precise. In most cases sentences are translated sentence by sentence. This source is not publicly available and can be used only for internal evaluation.

However, the laboratory environment provides a good basis for the experiments. The corpus contains a small amount of word-related data. The performance of the system can be substantially improved, and further work is needed to improve the accuracy of the system. We have also considered the possibility of using other resources in the translation process. These include the use of statistical methods and the incorporation of additional information from the English and Czech corpora. The system is able to handle one-to-many alignments from English to Czech. Therefore it is useful to have more words in Czech.

The Czech/English system is able to handle one-to-many alignments from English to Czech. Therefore it is useful to have more words in Czech.
Figure 10: Distortion probabilities for Model 3 and Model 4.
Figure 11: Test corpus perplexity during the EM-algorithm (Hansards corpus; transfer from Model 2 to Model 3 in iteration 1).
I am convinced that team work is the key for the realization of one's dreams.
Table 1: Examples of alignment of artificial words in the training corpus

<table>
<thead>
<tr>
<th>English</th>
<th>Czech</th>
<th>Artif. Word</th>
<th>English Word</th>
<th>Part of Speech</th>
</tr>
</thead>
</table>
| We can observe the progress of quality of translation obtained by the English tool from the baseline to the existing.

In Table 1, we see examples of artificial words alignment in the training corpus (first 4 cases in order).

We carried out a human evaluation of translations to observe progress obtained by each level of preprocessing.

<table>
<thead>
<tr>
<th>Translation tool</th>
<th>Average assigned marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Egypt toolkit</td>
<td>2.5</td>
</tr>
<tr>
<td>Simple lemmatized</td>
<td>3.0</td>
</tr>
<tr>
<td>English-like</td>
<td>3.5</td>
</tr>
<tr>
<td>AlignTemplates</td>
<td>4.0</td>
</tr>
<tr>
<td>Commercial 1</td>
<td>4.0</td>
</tr>
<tr>
<td>Commercial 2</td>
<td>4.0</td>
</tr>
</tbody>
</table>

In our particular case, the evaluation was done by two evaluators on 66 randomly chosen sentences from the test data. The tool for the human evaluation, which allows us to make an evaluation via the Internet, was developed during the workshop. It displays the original sentence (in Czech) and translations from different translation systems. Evaluators assign marks from 1 to 5 to each translation. Mark 1 is the best, mark 5 is the worst translation.

Table 1 shows examples of artificial words alignment in the training corpus (first 4 cases in order).

We can observe the progress of quality of translation obtained by the Egypt toolkit from the baseline to the simple lemmatized version and to the English-like version of Czech input (Czech/English) in comparison with the two commercial systems and the AlignTemplates system. Results on the English-like version are better than one of commercial systems and almost the same as the second one.
The function before training gives more weight to decoding parameters. The second was simply to add
sentences of the Alignment Template system to the training of the model. The function
parameters of the Alignment Template system is further motivated by the fact that the word alignment process
is closely related to the training of the model. This motivated the use of bilingual dictionaries.

### Table 2: Human Evaluation of Czech/English Translation

<table>
<thead>
<tr>
<th>System</th>
<th>Average</th>
<th>&lt; 4</th>
<th>4</th>
<th>5</th>
<th>6–7</th>
<th>8 at least</th>
<th>Exact</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 ×</td>
<td>3.6</td>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3.0</td>
<td>3.4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2.9</td>
<td>3.1</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3.0</td>
<td>3.4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2.9</td>
<td>3.1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3.0</td>
<td>3.4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2.9</td>
<td>3.1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3.0</td>
<td>3.4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Conclusion

We carried out the first experiments on statistical machine translation from Czech to English. We can observe
some of the results of the system, but the others are exactly the same as in the Reader's Digest test corpus.

By comparison, in the Reader's Digest test corpus, only 1.4% of translations were exact. The average
proportion of exact translations was 1%, which is significantly lower. This suggests that the model
parameters of the Alignment Template system are important. The results were measured on the basis of
human evaluation performed by independent evaluators. The results are significant after the first experiment. The
tests were performed on the basis of different sizes of training sets. We can observe

Two approaches to counter this are to apply a stronger filter when seeding and to make the dictionary that is given to the algorithm in the seeding step "washed out" with successive iterations of training.

<table>
<thead>
<tr>
<th>Value of $C$</th>
<th>Final test perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19.6</td>
</tr>
<tr>
<td>2</td>
<td>19.7</td>
</tr>
<tr>
<td>4</td>
<td>19.4</td>
</tr>
<tr>
<td>8</td>
<td>19.9</td>
</tr>
<tr>
<td>16</td>
<td>20.1</td>
</tr>
<tr>
<td>32</td>
<td>20.4</td>
</tr>
<tr>
<td>64</td>
<td>20.6</td>
</tr>
<tr>
<td>128</td>
<td>20.8</td>
</tr>
<tr>
<td>Baseline</td>
<td>21.0</td>
</tr>
</tbody>
</table>

Note that a lower test perplexity score corresponds to lower surprise at the test set sentences.

The results of this experiment are shown in the table above. Note that a lower test perplexity score corresponds to lower surprise at the test set sentences.

The results of this experiment are shown in the table above. Note that a lower test perplexity score corresponds to lower surprise at the test set sentences.

Because the function is based on one-to-one mappings, we modified the dictionary so that it contained only one-to-one entries. Multiword entries were converted to multiple entries (i.e., the cross-product of the words in the original entry). Additionally, we limited the entries to those where both words appeared in the corpus. The value of $C$ was set at 2, 4, 8, 16, 32, 64, and 128, non-dictionary pairs received a factor of 1.

We experimented with seeding the function so that word pairs $(w_{CZ},w_{EN})$ received a factor of $C$.

$$\frac{\text{word frequency}_{CZ} \cdot \text{word frequency}_{EN}}{1}$$

GIZA++ model I training begins with a uniformly distributed function.

### 8.1 Experiment I: Seeding The Translation Parameters

I training has been modeled. We have two models: I and I. Training the model that has the global optimization of parameters in a model is not the best approach, but it is the model that has the best approach. The model is not the best approach, but it is the model that has the best approach. The model is not the best approach, but it is the model that has the best approach.
The longest common subsequence ratio \((LCSR)\) measure of string similarity was proposed by Melamed [1995]. They consist of a set of weights for each mapping of a character in the first language to a character in the second language. These weights are learned from a set of known cognate pairs. We used the simplest of his methods, which maps single characters to single characters only, and within certain predetermined equivalence classes (in our case, vowels, consonants, numbers, and punctuation). A word pair can only be scored (using Tiedemann's methodology) if the ratio of the words' lengths is between \(3/10\) and \(1\) and both words are at least 4 characters long.

Because we did not have a set of known cognate pairs, we instead trained the weighting function on the entire dictionary. Needless to say, most of the entries in the dictionary are not cognate pairs. But the hope was that the noise from non-cognate pairs would cancel itself out. We then took the top scoring pairs from the dictionary as a set of known cognates. We had to arbitrarily choose a threshold value that would separate true cognates from garbage. It must be noted that there were no faux amis in this set, because all of the pairs came from a dictionary and could be assumed to be translation pairs.

The following table shows the number of co-occurring word pairs that scored above a certain percentage of their score range.

<table>
<thead>
<tr>
<th>Score Range</th>
<th>Number of Pairs</th>
<th>Cumulative Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.60-0.75</td>
<td>66666660</td>
<td>0.0%</td>
</tr>
<tr>
<td>0.75-0.90</td>
<td>66666600</td>
<td>15.5%</td>
</tr>
<tr>
<td>0.90-1.00</td>
<td>66666000</td>
<td>30.0%</td>
</tr>
<tr>
<td>1.00-1.15</td>
<td>66660000</td>
<td>45.5%</td>
</tr>
<tr>
<td>1.15-1.30</td>
<td>66600000</td>
<td>60.0%</td>
</tr>
<tr>
<td>1.30-1.45</td>
<td>66000000</td>
<td>75.5%</td>
</tr>
<tr>
<td>1.45-1.60</td>
<td>60000000</td>
<td>90.0%</td>
</tr>
<tr>
<td>1.60-1.75</td>
<td>60000000</td>
<td>95.5%</td>
</tr>
<tr>
<td>1.75-1.90</td>
<td>60000000</td>
<td>99.0%</td>
</tr>
</tbody>
</table>

Using the hypothesized cognates, we can now augment the dictionary. The hope is that alignments between these cognate pairs will improve the performance of a model that uses (in one way or another) this dictionary.
In Candide (GIZA) Model training, the function is based on co-occurrence data. A pair of tokens is said to co-occur if they appear in parallel sentences. Melamed describes an "oracle filter" which uses a bilingual lexicon to limit events which can be considered co-occurrences [Melamed, 1995].

Essentially, co-occurrence is redefined as follows: Given a bilingual lexicon and parallel sentences, each token is said to co-occur if its frequency drops when co-occurrence is filtered out. This approach is different from traditional methods which rely on statistical measures. The results are shown in the table below.

### Results of Constraining Co-occurrence

<table>
<thead>
<tr>
<th>Lexicon</th>
<th>Final test perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>None (baseline)</td>
<td>986.58</td>
</tr>
<tr>
<td>Dictionary</td>
<td>986.17</td>
</tr>
<tr>
<td>Dictionary + Cognates</td>
<td>980.33</td>
</tr>
<tr>
<td>Dictionary + Cognates</td>
<td>978.43</td>
</tr>
<tr>
<td>Dictionary + Cognates</td>
<td>976.60</td>
</tr>
<tr>
<td>Dictionary + Cognates</td>
<td>974.63</td>
</tr>
<tr>
<td>Dictionary + Cognates</td>
<td>972.01</td>
</tr>
<tr>
<td>Dictionary + Cognates</td>
<td>969.83</td>
</tr>
<tr>
<td>Dictionary + Cognates</td>
<td>967.35</td>
</tr>
<tr>
<td>Dictionary + Cognates</td>
<td>965.63</td>
</tr>
</tbody>
</table>

It is interesting to note that the greatest benefit seems to come when a merged lexicon of the original dictionary and cognate pairs scoring above 0.2 is used. It was noted previously that the intuitive threshold is at about 0.25; this indicates that even boosting pairs which are not cognates may aid in the alignment process. Also, note that a significant improvement occurs even when cognates are not used, especially when compared to the baseline.

In addition to test perplexity, we examined some Viterbi word alignments of test corpus sentences which improve word alignment in this type of model using the co-occurrence constraint.
The baseline model built by GIZA made the following alignments (a star indicates an incorrect alignment):

<table>
<thead>
<tr>
<th>English</th>
<th>Czech</th>
</tr>
</thead>
<tbody>
<tr>
<td>also</td>
<td>zpusobit</td>
</tr>
<tr>
<td>gigantic</td>
<td>obrovský</td>
</tr>
<tr>
<td>seismic</td>
<td>seismický</td>
</tr>
<tr>
<td>wave</td>
<td>vlna</td>
</tr>
<tr>
<td>tsunami</td>
<td>tichý</td>
</tr>
<tr>
<td>that</td>
<td>který</td>
</tr>
<tr>
<td>far away</td>
<td>moustek</td>
</tr>
<tr>
<td>as</td>
<td>jako</td>
</tr>
<tr>
<td>California</td>
<td>Kalifornie</td>
</tr>
</tbody>
</table>

Note that there are several pairs of cognates that were correctly aligned (although "California" should have aligned to the morphological marker "-a"), indicating perhaps that the constraint is too strong in some instances. Also, there is less garbage collection by unknown words (e.g., "tsunami" has no parallel word in the Czech sentence and it is left unaligned). The word "seismic" no longer collects three Czech words but is aligned to its cognate. The dictionary referred to in this experiment is the original dictionary described above.

Our third experiment is superficially similar to work done by Brown et al. in which a bilingual lexicon was appended to the corpus for training ([Brown et al.], 1993). Each lexicon entry is added to the training corpus. The advantages of this approach are numerous: multiple-word entries can be handled easily (e.g., each entry is the whole word; the word "tsunami" has no parallel word in the Czech sentence and it is left unaligned). Also it is possible to weight the entries differently. In a normal situation, such a weighting of sentence pairs indicates the number of times that each sentence is a weight of one in the training corpus. The dictionary referred to in this experiment is the original dictionary described above.

There are two types of lexical addition that we used. The first was to add every lexicon entry that we used. The next was to add only those entries which consisted of items that appeared as co-occurrences in the corpus; the second was to add only those entries which consisted of items that appeared as co-occurrences in the corpus when the constraint described above is used (with the dictionary and cognates used in the baseline model built by GIZA).
Surprisingly, after initial trials, the first weighting function with frequency/entry = 1 gave the greatest improvement. A smooth value of 0.15 was selected as the minimum weight and entry threshold was set low enough to improve perplexity. The results of this experiment are the most promising. The test sentence Viterbi alignments show similar to those in Experiment 2. The alignments appeared to be more intuitive when the lexicon used was the original dictionary merged with cognates thresholded at 0.3. The reduction in test perplexity was greater than in the previous experiments. The table below shows the final test perplexity values for a sampling of the experiments.

<table>
<thead>
<tr>
<th>Lexicon</th>
<th>Final test perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary (all)</td>
<td>14.448</td>
</tr>
<tr>
<td>Cognates</td>
<td>15.001</td>
</tr>
<tr>
<td>Dictionary (co-occurring pairs)</td>
<td>15.025</td>
</tr>
<tr>
<td>Cognates</td>
<td>15.050</td>
</tr>
<tr>
<td>None (baseline)</td>
<td>15.070</td>
</tr>
</tbody>
</table>

It is interesting to note that the greatest improvement from cognates came with extremely low thresholds (0.3). Also, when only cognates were used and the threshold was set low enough, the improvement was actually better than the results when only the dictionary was used. The relative length from smooth values with a Czech word in the test corpus but not the training corpus.

The results of this experiment demonstrate the value of dictionaries and cognates in building translation models, and they indicate that treating such information as data or using it as a constraint in training the model are two computationally inexpensive and effective ways of improving translation models. In addition, they indicate that meaning shift information is valuable as a constraint in training the model.
A late entry to our list of workshop goals (suggested by David Yarowsky) was to set aside a 24-hour period, late in the workshop, to be devoted to building an MT system for a completely new language pair. This would test all of our components and allow us to see how rapidly a prototype could be built.

Here are the activities that were within the planned scope of this experiment:

- Choose a new language pair with a bilingual corpus
- Install the corpus
- Sentence-align the corpus
- Split the corpus into training and testing portions
- Train a translation model on the full corpus
- Train a language model on the English side of the corpus
- Compute perplexity-based learning curves for these models
- Obtain English translations (decodings) of proposed unseen source-language sentences
- Choose a new language pair with a bilingual corpus

Likewise, here are activities that were outside the planned scope, but would have been useful:

- Implement significant linguistic transductions to improve statistical models
- Use dictionaries and cognates to further improve models
- Evaluate translation with human judges
- Use characteristics of our new models to further improve models
- Evaluate significant linguistic transductions to improve statistical models

Here are the activities that were outside the planned scope of this experiment:

1. Obtain Chinese/English as our new language pair.
2. We considered other language pairs, and
3. We decided on Chinese/English as our test language pair.
4. We considered using the STRAND [Resnik, 1999] system from the University of Maryland to collect bilingual data from the web.

The corpus we used was "Hong Kong Laws," a collection of promulgated laws in both Chinese and English. The corpus has 7 million words on the English side of the corpus. We replaced any Chinese sequence by "@@/@/@/@," and did likewise on the Chinese side.

The most serious transduction work involved mixed-language strings. These occurred fairly often, as a term of law written in English is sometimes followed by its Chinese equivalent, in parentheses. On the English side of the corpus, we replaced any Chinese sequence by "@@/@/@/@/, and did likewise on the Chinese side.
The perplexity value measures the ability of the system to recognize whether or not test sentence pairs are indeed translations of each other. It is clear that additional corpora would significantly improve this recognition capability.

Finally, we used the CMU-Cambridge Language Modeling Toolkit to build an English trigram language model for Hong Kong legal text.

We used the Waver decoder to translate 100 Chinese test sentences. The final translations were:
The Egypt toolkit is intended to be a base for future experimentation. We expect that many things will be tried, and that some will succeed. These things may involve exploiting new resources, inventing better translation models, and possibly traveling further into the realms of syntax and semantics. We will not anticipate them in this report. However, we know of certain enhancements that would improve the immediate use of the toolkit, and so we list these.

**Context-dependent translation probabilities.**
Brooke et al., 1993; Berger et al., 1996; Melamed, 1998a. We would like to employ translations that are sensitive to context, as it can ease the pressure on the language model to disambiguate words in the input text.

**Faster training.**
We have found that training a new model can be done more efficiently and effectively.

**English morphology.**
Given the number of word forms in English is fairly small compared to many other languages, some English morphology would serve to reduce the vocabulary size and improve statistics.

**Phrases.**
It is often possible to locate word sequences that translate as a whole. It would be useful for translation models to uncover these and for decoding to take them into account.

**Better initial alignments.**
Models 3, 4, and 5 rely on small subsets of their training data, so it is critical that the simple models used to bootstrap them are accurate, particularly with small data. Better initial alignments would improve the vocabulary size and improve statistics.

**About the Participants**

Yaser Al-Onaizan is a Ph.D. student at the Computer Science Department at the University of Southern California (USC) and a research assistant at USC’s Information Sciences Institute (USC/ISI). He received his MS from USC in 1995 and his BS from King Saud University in Riyadh, Saudi Arabia in 1992 both in Computer Science (FCS) and a research assistant at USC, Information Sciences Institute (USC/ISI). He received

Jan Curin graduated from the Faculty of Mathematics and Physics, Charles University, Prague in 1998 with a MSc thesis "Automatic Extraction of Translation Lexicons." He continues his studies at the Institute of Formal and Applied Linguistics, Charles University, Prague, as a full-time postgraduate student. He works on formal and applied linguistics, machine learning, and statistical natural language processing. He received his Master's degree in 1998.

Acknowledgements

Thanks to Yaser, Angela, and Victoria, and to everyone who programmed everywhere.

Thanks to everyone involved in the MWF Awards for supporting this workshop.

Thanks to our friends for supporting our workshop in Prague. Also, thanks to

Hedvig's Law and the Baltimore group at the University of Maryland for permission to test the models on their data.
We would like to thank Adwait Ratnaparkhi for allowing us to use the source code of his maximum entropy tagger, and the RALI group at the University of Montreal for permission to experiment with hand-tagged data.

Martin Cmejrek for collaboration on parallel Czech/English data.
Also, thanks to Reader’s Digest Vybér (Prague, Czech Republic) for granting the license for using their textual material and to IBM Czech Republic for the chance to run test translations on their data.

We would like to thank Adwait Ratnaparkhi for allowing us to use the source code of his maximum entropy tagger, and the RALI group at the University of Montreal for permission to experiment with hand-tagged data.

Martin Cmejrek for collaboration on parallel Czech/English data.
Also, thanks to Reader’s Digest Vybér (Prague, Czech Republic) for granting the license for using their textual material and to IBM Czech Republic for the chance to run test translations on their data.

Mr. C. Shih for providing the tools for Czech morphological analysis, tagging, and lemmatization, and to Michael Collins for providing the possibility.

Thanks to West Group for supporting Dan Melamed’s participation in the workshop. Also, thanks to
References


Dau Amick is an associate professor in the Computer Science Department at the University of Utah.

Dana Fong is a graduate student in the Computer Science Department at Stanford University.


Hinrich Schütze is a professor of computer science at the Technical University of Munich.


Mark Jordan is a professor of computer science at Stanford University.


