Statistical Machine Translation

Final Report, JHU Workshop 1999

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight,
John Lafferty, Dan Melamed, Franz-Josef Och,
David Purdy, Noah A. Smith, David Yarowsky

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 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.

Recently, statistical data analysis has been used to gather MT knowledge automatically from parallel bilingual text. Unfortunately, these techniques and tools have not been disseminated to the scientific community in very usable form, and new follow-on ideas have developed sporadically. In a six-week summer workshop at Johns Hopkins University, we constructed a basic statistical MT toolkit (called Egypt) intended for distribution to interested researchers. We also used the toolkit as a platform for experimentation during the workshop. Our experiments included working with distant language pairs (such as Czech/English), rapidly porting to new language pairs, managing with small bilingual data sets, speeding up algorithms for decoding and bilingual and text training, and incorporating morphology, syntax, dictionaries, and cognates. Late in the workshop, we built an MT system for a new language pair (Chinese/English) in a single day. We describe both the toolkit and the experiments in this report.

1 Introduction

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 [Resnik, 1999].

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 (and slightly exceeded) commercial-level translation quality through automatic bilingual-text analysis. Much of this effort was documented in [Brown *et al.*, 1993a].

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 the field of MT. This is partly due to the fact that the mathematics involved were not particularly familiar to computational linguistics researchers 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. Such infrastructure is available, for example, in the speech recognition community. It is possible for an individual researcher to

1. come up with a new idea (for example, to study how topical context can be used to resolve acoustic ambiguities [Florian and Yarowsky, 1999].
2. do some initial studies (for example, to observe effects on task-independent statistical measures such as language model perplexity)
3. test the new idea the context of an end-to-end speech recognition task (for example, by measuring word-error rate on a standard data set such as SWITCHBOARD or CALLHOME)
4. revise the idea
5. and compare the final results to previous work

It is far more difficult to do this sort of thing in machine translation—in fact, it is seldom done. In practice, a huge proportion of the work would be need to be spent on large-scale MT software development, rather than on inventing and revising the new idea itself. Of course, evaluating machine translation is harder than evaluating speech recognition word-error rate, and this complicates the picture substantially. However, the lack of tools and data sets remains the biggest barrier to entry in this field.

2 Workshop Goals

At the outset of the workshop, we proposed five 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 for performing actual translation. It would be aimed at two types of users:
   - users who say to themselves "I have a parallel corpus—what can I do with it?" and
   - users who say to themselves "I have a new idea for statistical machine translation—how can I test it?"
2. Build a Czech-English machine translation system during the workshop, using this toolkit.
3. Perform baseline evaluations. These evaluations would consist of both objective measures (statistical model perplexity) and subjective measures (human judgments of quality), as well as attempts to correlate the two. We would also produce learning curves that show how system performance changes when we vary the amount of bilingual training text.
4. Improve baseline results through the use of morphological and syntactic transducers.
5. Late in the workshop, build a translator for a new language pair in a single day.

We largely achieved these goals, as described in this report. We also had time to perform some unanticipated beyond-the-baseline experiments: speeding up bilingual-text training, using online dictionaries, and using language cognates. Finally, we built additional unanticipated tools to support these goals, including a sophisticated graphical interface for browsing word-by-word alignments, several corpus preparation and analysis tools, and a human-judgment evaluation interface.

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 $\mathrm{P}(e|f)$ over all pairs of strings, so that given f, we can output the e which maximizes $\mathrm{P}(e|f)$. We break this problem down into two simpler problems by using Bayes' Rule:

$$\mathrm{P}(e \mid f) \sim \mathrm{P}(e) \cdot \mathrm{P}(f \mid e)$$

The $\mathrm{P}(e)$ factor helps our e output to be natural and grammatical, while the $\mathrm{P}(f|e)$ 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 $\mathrm{P}(e)$ and $\mathrm{P}(f|e)$ estimates. For example, below are several English translations of some French sentence. Translations that pass $\mathrm{P}(e)$ are marked with `x` in the first column; likewise for translations that pass $\mathrm{P}(f|e)$.

	P(e)	P(f|e)
Jon appeared in TV.		x
Appeared on Jon TV.		
In Jon appeared TV.		x
Jon is happy today.	x	
Jon appeared on TV.	x	x
TV appeared on Jon.	x	
TV in Jon appeared.		
Jon was not happy.	x	

Both models are flawed. $\mathrm{P}(e)$ assigns too much probability to *TV appeared on Jon*, while $\mathrm{P}(f|e)$ 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].

```
Garcia and associates .
Garcia y asociados .

Carlos Garcia has three associates .
Carlos Garcia tiene tres asociados .

his associates are not strong .
sus asociados no son fuertes .

Garcia has a company also .
Garcia tambien tiene una empresa .

its clients are angry .
sus clientes estan enfadados .

the associates are also angry .
los asociados tambien estan enfadados .

the clients and the associates are enemies .
los clientes y los asociados son enemigos .

the company has three groups .
la empresa tiene tres grupos .

its groups are in Europe .
sus grupos estan en Europa .

the modern groups sell strong pharmaceuticals .
los grupos modernos venden medicinas fuertes .

the groups do not sell zanzanine .
los grupos no venden zanzanina .

the small groups are not modern .
los grupos pequenos no son modernos .
```

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 .
Garcia and 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 .
```

Figure 2: Monolingual text corpora used to train SMT systems (adapted from [Knight, 1997]).

- "The Mathematics of Statistical Machine Translation" [Brown *et al.*, 1993a]. A full mathematical description of the IBM translation models. This paper provides 99 percent information needed for implementation. The material is presented in equation format, and there is a fair amount working involved in turning these equations into equivalent program code.
- "Automating Knowledge Acquisition for Machine Translation" [Knight, 1997]. A very high-level description of the ideas involved in statistical machine translation, as well as knowledge acquisition for other types of machine translation. This is a reasonable paper to start with, but it does not give enough information for implementation.
- "Translation with Finite-State Devices" [Knight and Al-Onaizan, 1998]. A view of translation as a cascade a finite-state devices. Readers familiar with such devices may find it convenient to start here.

We chose Model 3 of [Brown *et al.*, 1993a] as our baseline for the workshop, so we describe it first. The purpose of the translation model is to assign a probability P(f | e) to every pair of sentences. Model 3 represents a particular naive theory about how, given an English sentence, one might stochastically produce French equivalents. The theory goes like this:

1. For each English word e in the sentence, we choose a fertility phi with probability $n(\phi \mid e)$.
2. Call the sum of all those fertilities m-prime.
3. Make a new English string by deleting words with fertility zero, copying words with fertility one, duplicating words with fertility two, etc.
4. After each of these m-prime words, make a decision to insert a NULL (with probability $p1$) or not (with probability $p0$).
5. Let ϕ_0 be the number of NULL words inserted.
6. Let m be the total number of words in front of us now, m-prime + ϕ_0.
7. Replace each word e with a foreign-language translation f, according to the probability table t(f | e).
8. Assign target-language positions to those foreign-language words not generated by NULL, according to the probability table d(j | i,l,m). Here, j is the target-language position, i is the position in the English string of the words that generated the French word now being placed, l is the number of words in English string, and m is the number of words in the French string.

9. If any target-language position is over subscribed (contains more than one word), then return failure.
10. Assign target-language positions to the NULL-generated words. These should fall into empty positions. Any assignment is deemed equally likely as any other, so any assignment can be carried out with probability $1/\phi_0$.
11. Finally, read off the French string.

Many sequences can yield the same French string from the same English string. At least some of these sequences can be collapsed if we consider word alignments. Here is a sample alignment from [Brown *et al.*, 1993a]:

```
NULL  And  the  program  has  been  implemented
  |     |    |       |      |     |        |
             |       |      |     |     +-+---+
             |       |      |     |     | |   |
            Le programme    a    ete   mis en application
```

Model 3 only considers alignments in which every French word is connected to exactly one English word. Each English string is assumed to begin with a special word NULL, which French words may connect to. An alignment can therefore be represented as a vector a, assigning to each French position a "source" English position. In Model 3, an English word may connect to zero or more French words, corresponding to its fertility in the sentence.

We can compute the number (say, k) of generative sequences (as above) that correspond to certain word alignment, and it turns out that all those sequences will have the same probability. So we can compute the probabilities employed along one sequence, and multiply the results by k to get P(a,f | e). To get P(f | e), we sum P(a,f | e) over all word alignments a. The formula for P(a,f | e) is given in section 25 of the "Statistical MT Tutorial Workbook" [Knight, 1999], and in equations 32 and 101 of [Brown *et al.*, 1993a]. It is a function of the words in the sentence pair and the probability tables n, p, t, and d:

$$
\begin{aligned}
\mathrm{P}(a, f \mid e) = \; & \binom{m - \phi_0}{\phi_0} {p_0}^{m-2\phi_0} {p_1}^{\phi_0} \cdot \\
& \prod_{i=1}^{l} n(\phi_i \mid e_i) \prod_{i=0}^{l} \phi_i! \cdot \\
& \prod_{j=1}^{m} t(f_j \mid e_{a_j}) \cdot \\
& \frac{1}{\phi_0!} \cdot \prod_{j:a_j \neq 0} d(j \mid a_j, l, m)
\end{aligned}
$$

Training is a matter of inducing those four probability tables from a bilingual corpus. This is what GIZA does. The basic idea is to bootstrap. For given English word e, we initially pretend that all French words are equally likely translations. For a given sentence pair, all alignments will therefore look equally likely as well. Every time we see a certain word pair co-occurring in an alignment, we mark down a "count." After we have traversed the entire corpus, we normalize these counts to create a new word-translation table. (The same goes for the fertility, null, and distortion tables). According to these new tables, some alignments will now be more probable than others. We collect counts again, but now weigh co-occurrences by the probability of the alignments that they occur in. This is the EM algorithm:

```
set t, d, n, and p tables uniformly
for several iterations
  set up count tables tc, dc, nc, and pc with zero entries
  for each sentence pair (e, f) of lengths (l, m)
    for each word alignment a of (e, f)
```

```
target position is j
target string has m words
old alignment is a
old fertilities are g(0..l)
new alignment is a'
old source position for j is i
new source position for j is i'
word fj now points to word ei'

if (i == i')

  P(a' | e,f)
  ----------- = 1.0
  P(a | e,f)

if (i > 0) and (i' > 0)

  P(a' | e,f)   g(i')+1   n(g(i')+1 | ei')   n(g(i)-1 | ei)   t(fj | ei')   d(j | i',l,m)
  ----------- = ------- x ---------------- x -------------- x ----------- x -------------
  P(a | e,f)      g(i)    n(g(i') | ei')     n(g(i) | ei)     t(fj | ei)    d(j | i,l,m)

if (i == 0)

  P(a' | e,f)   t(fj | ei')    n(g(i')+1 | ei')   d(j | i',l,m) x g(i')+1      g(0) x (m-g(0)+1)       p0^2
  ----------- = -----------  x ---------------- x ----------------------- x ---------------------- x ----
  P(a | e,f)    t(fj | NULL)   n(g(i') | ei')                1              (m-2g(0)+2)(m-2g(0)+1)    p1

if (i' == 0)

  P(a' | e,f)   t(fj | NULL)    n(g(i)-1 | ei)              1            (m-2g(0))(m-2g(0)-1)    p1
  ----------- = -----------  x ---------------- x ------------------ x -------------------- x ----
  P(a | e,f)    t(fj | ei)      n(g(i) | ei)      d(j | i,l,m) x g(i)     (m-g(0))(g(0)+1)      p0^2
```

Figure 3: Calculating the relative probability of a word alignment a' obtained by moving a single link in word alignment a.

```
      compute P(a | e, f) = P(a,f | e) / sum over all a' of P(a',f | e)
      for j = 1 to m
        tc(fj | ei) += P(a | e,f)
      update dc, nc, and pc similarly
  normalize tc, dc, nc, and pc to create new tables t, d, n, and p
```

Unfortunately, there are too many alignments for us to enumerate, so this method is impractical. Instead, we find a reasonable-looking alignment (using some other method), score it with the Model 3 formula above, and then iteratively improve it through small steps until it can be improved no further. This is called hill climbing. We call the resulting alignment the Model 3 Viterbi alignment, even though it is an approximation—there may be a more probable alignment somewhere else. We can collect counts over this Viterbi alignment and its neighborhood, defined as alignments that can be reached by the same "small steps." A small step consists of either (1) moving a French-word connection from one English position to another, or (2) swapping connection positions between two French words. When taking a small step from an alignment a to another alignment a', we can avoid the wholesale recomputation of P(a',f | e) by computing it in terms of P(a,f | e). Figure 3 illustrates case (1) above, in which we move a single connection. Figure 4 illustrates case (2).

These tricks make hill climbing fast, and they also allow us to determine alignment weights for collecting counts in the neighborhood of the pseudo-Viterbi alignment.

```
target positions are j1 and j2
target string has m words
old alignment is a
old fertilities are g(0..l)
new alignment is a'
old source position for j1 is i1
new source position for j1 is i2
old source position for j2 is i2
new source position for j2 is i1
so word fj1 now points to word ei2
so word fj2 now points to word ei1

if (i1 == i2)

  P(a' | e,f)
  ----------- = 1.0
  P(a | e,f)

if (i1 > 0) and (i2 > 0)

  P(a' | e,f)   t(fj1 | ei2)   t(fj2 | ei1)   d(j1 | i2,l,m)   d(j2 | i1,l,m)
  ----------- = ------------ x ------------ x -------------- x --------------
  P(a | e,f)    t(fj1 | ei1)   t(fj2 | ei2)   d(j1 | i1,l,m)   d(j2 | i2,l,m)

if (i1 == 0)

  P(a' | e,f)   t(fj1 | ei2)   t(fj2 | ei1)   d(j1 | i2,l,m)
  ----------- = ------------ x ------------ x --------------
  P(a | e,f)    t(fj1 | ei1)   t(fj2 | ei2)   d(j2 | i2,l,m)

if (i2 == 0)

  P(a' | e,f)   t(fj1 | ei2)   t(fj2 | ei1)   d(j2 | i1,l,m)
  ----------- = ------------ x ------------ x --------------
  P(a | e,f)    t(fj1 | ei1)   t(fj2 | ei2)   d(j1 | i1,l,m)
```

Figure 4: Calculating the relative probability of a word alignment a' obtained by swapping two links in word alignment a.

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:

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

It is possible to use uniform fertilities when starting Model 3, but [Brown *et al.*, 1993a] give equations suggesting an initialization of fertilities that takes Model 2 knowledge into account. GIZA's implementation is shown in Figure 5.

[Brown *et al.*, 1993a] advise replacing words that occur only once (*singletons*) with a special token UNK. Partially due to cheaper/bigger computer memories, and partially due to careful implementation, we were able to induce translation models for the entire vocabulary of the Canadian Hansard Set A, without even discarding singletons. This is a qualitative advance over what Brown et al. were able to accomplish given the resources available to them. The more detailed training data resulted in improved model perplexity, a statistical measure we discuss in Section 5).

4.2 Decoder

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. Our implementation is very similar to the decoder used in the *Candide* system.[1]

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 WEAVER.

When the prototype decoder was completed during the workshop, we were pleasantly surprised with its speed and performance. From the days of DARPA machine translation evaluations carried out at IBM

[1] Since the *Candide* decoder was not documented beyond the fairly high-level description given in the 1992 IBM patent, it appears to be impossible to resolve detailed differences between the two implementations without recovering the original *Candide* source code. During the workshop we spoke several times with Peter Brown, Stephen Della Pietra, and Vincent Della Pietra, but many details could not be recalled.

```
phi-max = maximum fertility of any word
big-gamma(n) = partition of n into a sum of integers all >= 1
  (e.g., 1+2+4 is a partition of 7, and so is 1+1+1+1+3)
times(partition, k) = number of times integer k appears in partition

for j = 1 to m
  total = 0
  for i = 0 to l
    total += t(fj | ei) * a(i | j,l,m)
  for i = 0 to l
    pij =  t(fj | ei) * a(i | j,l,m) / total
    if (pij > 0.99) then pij = 0.99        ;; for under- and
    if (pij < 0.01) then pij = 0.01        ;; overflow later
for i = 1 to l
  for k = 1 to min(m, phi-max)
    beta = 0.0
    for j = 1 to m
      beta += [ pij / (1 - pij) ]^k
    alpha[i,k] = (-1)^(k+1) * beta / k
for i = 1 to l
  r = 1.0
  for j = 1 to m
    r = r * (1 - pij)
  for phi = 0 to min(m, phi-max)
    sum = 0.0
    for each partition p in big-gamma(phi)  ;; always enter this loop
      prod = 1.0                            ;; even if phi is 0!
      for k = 1 to phi
        prod = prod * alpha[i,k]^times(p,k) / times(p,k)!
      sum += prod
    c(phi | ei) += r * sum
```

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

during the early 1990s, one had come to expect that only short (~ 10 word) 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 constraints on the search, as described below. But the relative efficiency of the decoder bodes well for future work in this direction, and affirms the original spirit of the IBM work, which emphasized 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 new terrain in statistical translation, with context-sensitive or structural models that go beyond the trail blazed by the rudimentary Models 1–5.

4.2.1 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 structure for Models 1–3 in the WEAVER system is based on dense arrays and binary search rather than hash tables for lookup of the translation parameters.

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

$$d\,(j \mid i, l, m)$$

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 l is unknown. As a result, our implementation uses the parameters

$$d\,(j \mid i, m)\,.$$

Training of Model 3 can be carried out in three different modes: collecting counts in the E-step by summing only over the neighbors of the best alignment reachable (by moves and swaps) from the Model 2 Viterbi alignment, or by summing over the neighbors of 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.

The basic structure of the decoder is very simple. The interface to the `StackDecoder<State>` template is shown below:

```
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);
}
```

The basic operations that are performed during search are adding states and updating the thresholds of the frontier stacks, propagating these thresholds back to "ancestor" stacks. The interface to the decoder is also very basic:

```
class Decoder {
public:
  enum {ALIGN=0, DECODE};
  Decoder(TranslationModel &tm_, LanguageModel &lm_, int debug=0);
  ~Decoder();
  int  translate(Sentence &e_, Sentence &f_, int m=Decoder::DECODE);
  int  align(Sentence &e_, Sentence &f_);
  void writeAlignmentData(const String &fn, int id1, int id2) const;
  ...
```

One difference between the current implementation and the *Candide* decoder is that we do not "normalize" the stacks based upon the total unigram probability of the set of source words corresponding to that stack. As a result, other things being equal, there is a bias against hypotheses that contain one or more rare words.

4.2.3 Intermediate Calculations

Several intermediate calculations are required to carry out the search: inverse translations, language model "holes," and fertility marginals.

It is impractical to consider all possible translations of all words. Instead, we restrict to the most promising candidates by calculating "inverse translations." The inverse translation probability $t^{-1}(e \mid f)$ of a source word f is calculated as

$$t^{-1}(e \mid f) = \frac{t\,(f \mid e)\; p\,(e)}{\sum_{e'} t\,(f \mid e')\; p\,(e')}$$

where we use a unigram model $p\,(e)$ to estimate the prior probability of a target word being used. This prior term was not used in the IBM system, but we found that it gave better results to include it, and also allowed the candidate translation lists to be smaller.

In order to hypothesize a word that generates *no* source words–that is, a word with fertility zero–it is useful to have a modification of the trigram language model to estimate the probability

$$p\,(w_2 \mid w_1, w_3)$$

that a word falls in the between two words. Noah Smith modified the language modeling code to compute these values efficiently, using the approximation

$$p\,(w_2 \mid w_1, w_3) \;\propto\; p\,(w_1)\, p\,(w_2 \mid w_1)\, p\,(w_3 \mid w_2)\,.$$

When building up a hypothesis in which a single English word e aligns with several French words, we need to factor in the probability that e has fertility *at least* a given value. Thus, the sums

$$m(k \mid e) = \sum_{\phi=k}^{\phi_{\max}} n(\phi \mid e)$$

are precomputed and stored together with the fertility probabilities.

4.2.4 Conversion from GIZA to Weaver format

A program was written to convert the GIZA format for the translation models into Weaver format. The main task of this program is to sort the probabilities into a form suitable for the binary search data structures.

4.2.5 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 17 seconds per sentence. In this run, the top 12 inverse translations were considered for each source word, and the top 8 words according to the language model probabilities $p(w_2 \mid w_1, w_3)$ were considered as fertility zero extensions. The stack thresholding parameter was set to 0.001, meaning that those hypotheses with a score smaller than 0.001 times the best scoring hypothesis in a stack were not considered for extension. These represent fairly tight thresholds, typically resulting in several hundreds of thousands of hypotheses scored, rather than millions.

4.2.6 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 implement stack normalization, β-probabilities and other improvements in pruning. Further work also needs to be done to analyze search errors, and to support an "anytime" translation mode.

4.3 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). A translation model is built using iterative training on a sentence-aligned bitext. Translations are produced through a process known as "decoding."

Statistical translation systems are difficult to understand, even for their designers, because of the vast number of probabilities learned automatically. Candide Model 3, for which Cairo was designed, consists of four types of parameters to model unidirectional language translation. Using a French-to-English example, these are:

- 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 n French tokens.
- Distortion. The probability that any English token in indexed position i of an English sentence will align with an French token in position j of an 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 encoding (a NULL token is empty in English but is aligned to one or more French tokens). This is a single value for the model.

Cairo was implemented to allow inspection of the bilingual text word alignment process and the decoding process in statistical machine translation models.

4.3.1 Capabilities

Cairo takes as its input an SGML-style file that specifies the two sentences, their alignment, and all relevant model parameters.[2] 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 accommodates sentences of arbitrary length.

[2] Relevant parameters may include the probabilities associated with a word translation, fertility, etc. that actually occur in the alignment as well as other probabilities to be used as a basis for comparison.

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 a 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, then these will be displayed as well. For example, if the user clicks on an English word ε, a list of French words to which ε is likely to translate appear in the translation table, sorted by probability. The French words $\iota_1, \iota_2, \ldots, \iota_n$ which are aligned to ε for each alignment are displayed in colors matching their respective alignments. Likewise, sorted lists of fertilities, distortions, and likely English words (*i.e.*, expected by the language model) are displayed. One, two, or four of the tables may be shown at a time.

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.

4.3.2 Implementation and Use

Cairo is implemented in Java, using the Swing interface library; this makes it portable to any architecture and useful in Web-based applications. 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" gives additional insight. This tool has allowed us and others to track the progress of our translation models and discover models' biases that affect performance.

4.4 Whittle

Whittle is a corpus preparation tool. It splits the corpus into training and test corpora, generates vocabulary files, replaces words with integers assigned by frequency, discards sentences over a certain length, etc. Its options are as follows:

```
whittle v1.0, by mike jahr, 8/13/99

usage: whittle.pl [options] <input-directory | input-stem>
   -a       all: do not split into training and test sets
  *-b nn    specify baseline test set size; nn can be either an integer
              (the desired number of sentences) or a decimal (fraction of
              the total corpus size)
   -c       only generate a counts/vocab file; usually done automatically
   -d dd    specify output file directory; overriden by -o
   -E xx    specify suffix for English files
   -f i     specify minimum word frequency; less-frequent words will be
              replaced with the UNKNOWN token. i<=1 retains all words.
   -F xx    specify suffix for foreign-language files
   -h       print this help information
  *-i nn    interleaving ratio: fraction of test set to be spread evenly
              throughout the corpus. the rest is taken from the end.
   -l i     specify maximum sentence length; pairs where one or both are
              longer will be discarded. use i=0 for no max length.
```

```
 -n i     assign word identifiers starting from i
 -o d/p   specify output file directory and prefix; overrides -d
 -s nn    specify total output corpus size; nn as above
 -t nn    specify training set size; nn as above
 -T nn    specify test set size; nn as above. note that the test set
            cannot be larger than specified by the -b value.
 -u i     specify identifier of unknown word
 -v       print version  information
 -w       word format, no integerizing; output in human-readable form
--all         do not split corpus; identical to -a
--bst         output in binary (.bst) format
--snt         output in .snt format, one sentence per line
--sfreq       sort output by sentence frequency
--split       only split the corpora: output parallel files in the
                same format as the input
--test nn     specify test set size; identical to -T
--test-only   only generate a test set; equivalent to --train 0
--train nn    specify training set size; identical to -t
--train-only  only generate a training set; equivalent to --test 0
```

5 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 had about statistical machine translation was, how much data do you need? Are 50,000 sentence pairs enough? If you go from one million sentence pairs to two million, how much improvement will you see? We imagined that a lot of data is useful—if you've never seen the phrase "real estate" before in your parallel corpus, then you probably aren't going to translate it correctly. It is virtually mandatory in the machine-learning community to plot data size on the x-axis vs. performance on the y-axis (a learning curve). This is particularly important in natural language processing, where we can frequently "purchase" more data. For some reason, this type of study has not often been done (one exception is [Melamed, 1998a]). The learning curve leads naturally to a second question: what to plot on the y-axis? Machine translation is notoriously difficult to evaluate. It shares this notoriety with other tasks that involve generating human language, as opposed to interpreting it. It is possible to evaluate a speech recognizer by counting word errors; not so with a speech synthesizer. Likewise, it is easier to say whether a language interpretation system "got the right syntactic structure" than to say whether a generation system "produced a good syntactic structure." Machine translation involves both interpretation and generation.

Following many of the evaluation regimes that have been proposed for MT (e.g., [White and O'Connell, 1994]), we decided to go with either a simple 1-5 scoring mechanism, or a simple relative ranking of translations from different systems/configurations.

Most MT evaluation regimes involve collecting human judgments over many sentences, and these regimes are expensive for individual researchers. In the commercial world, the market provides a constant evaluation of translation software, albeit one that seems currently driven by factors other than output-text quality. In the final analysis, there is no measure that can substitute for user satisfaction. But this leaves the individual researcher in a bind.

The speech recognition community has largely focused on word-error recognition rate, in the belief that this figure will ultimately correlate with user satisfaction. And it is easy to measure. Researchers can try, discard, and adopt many new ideas without involving human subjects. They can also compare results on common test data sets. As mentioned above, even this is difficult to do in translation.

Interestingly, many speech researchers find it convenient to evaluate their ideas with respect to perplexity, a measure of how well a statistical model fits (or explains) the data. For example, in language modeling, the goal is to produce a program that assigns a probability P(e) to every string of words e. (All these probabilities sum to 1, so there is only so much to go around). It is possible to ask how good 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 S sentences is given by:

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

(N is the number of words in the corpus). If we want a high probability, then we want a low perplexity. Probabilities are typically in the unintelligible range, such as 10^{-43}, while perplexities look more like 50.6.

It is typical in language modeling to bet at least something on every conceivable string e'. If we accidentally bet nothing on e', then our P(e') would be zero, and our perplexity would be infinite. So if a language model uses previously-observed word-pair frequencies in constructing a probability score P(e) for a new string, it will typically smooth these frequencies so as to accept a string that contains novel word pairs.

In statistical MT, a key component is the translation model. We can do the same thing. An instantiated translation 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'. 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.

5.2 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 take t(f | e) to be uniform across all French words observed in both training and test corpora. For previously seen e, novel translations f are assigned a small t(f | e) constant floor value.

How is perplexity related to translation quality? This question has no clear answer at present, and it is a question that we wanted to investigate in the workshop. Test-set perplexity does not measure how well a system translates, 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—it could simply enumerate all possible translations and select the one that it recognizes as being the best. This does not work the other way around. A commercial MT system may produce good translations without being able to recognize one. Recognition has other applications as well, e.g., identifying whether or not to web documents constitute parallel text, or taking an initial automatic pass at scoring student translations.

But the main thing is that if practice follows theory, and improved test-set perplexity indeed leads to better translations, then an individual researcher can try, discard, and adopt new ideas daily using test-set perplexity as a guide, pending subjective human evaluations held less frequently. There is one additional interesting twist. Many of the parameter-learning algorithms used in statistical speech and MT (such as the EM algorithm) are designed to optimize training-set perplexity. Mathematically, that's all they do. If better perplexity means better translation, then maybe these algorithms are doing pretty much the right thing.

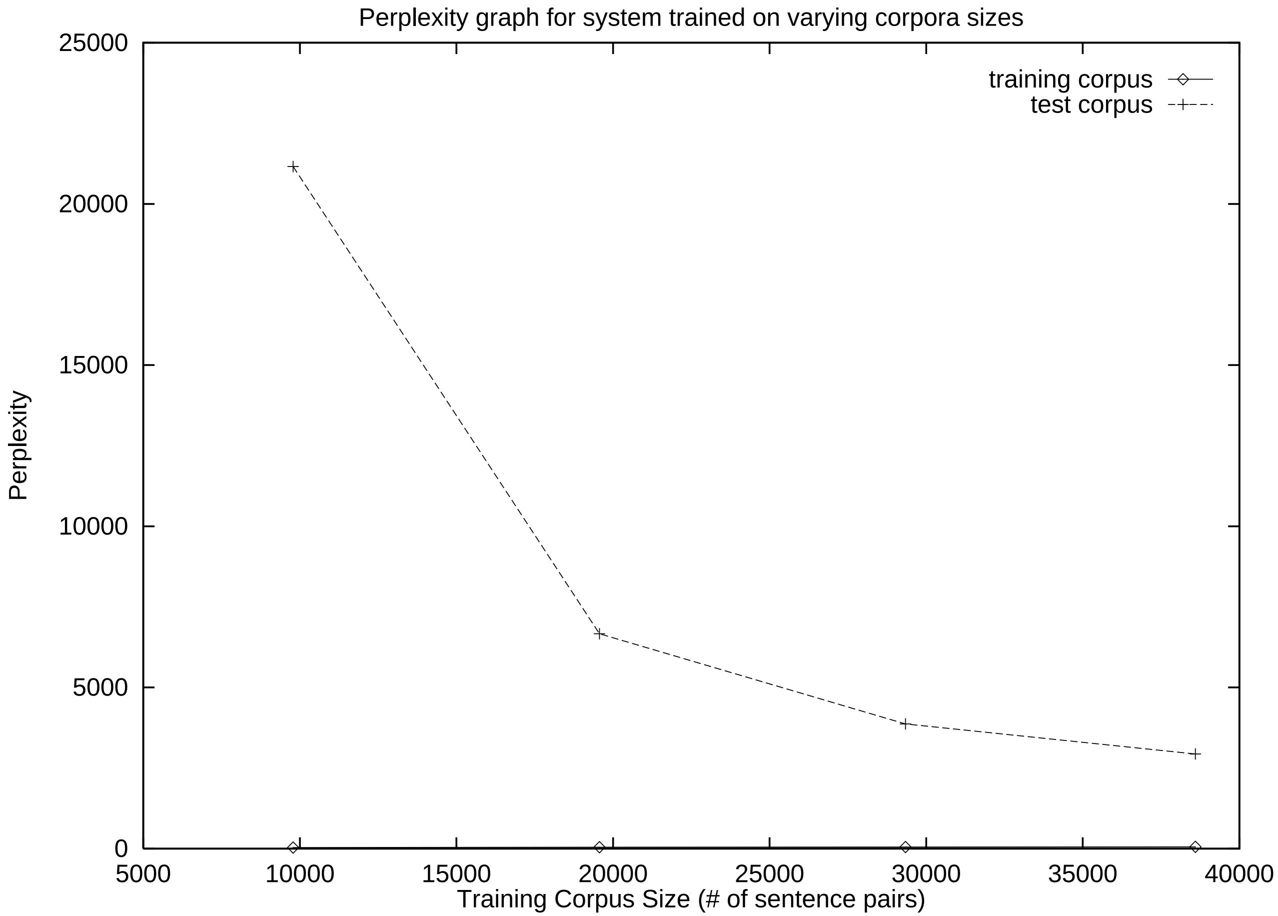

Figure 6: Perplexity as a function of bilingual training set size (Czech/English).

5.3 Training Iterations versus Test-set Perplexity

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. [Brown *et al.*, 1993a] reported such a curve for training-set perplexity, but not test-set perplexity. While training-set perplexity is guaranteed to decrease, this is not the case with 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.

Among the workshop participants, there was much disagreement over the value of test-set perplexity, and so we wanted to do experiments that would quantitatively relate objective perplexity with subjective human judgments of translation quality. We planned to generate models with various perplexities by training the baseline Model 3 system on variously sized parallel corpora. During the workshop, we decided that it would be good to generate additional data points by keeping corpus size constant while comparing the baseline system with our improvements. There was a fair amount of consternation over smoothing, which is a necessity (with tangible benefits) in language modeling, but something IBM never did in their translation modeling. It was interesting that before the Egypt toolkit was fully integrated (in the penultimate week of the workshop), several participants used perplexity to guide their research—despite reservations about its utility!

Moreover, the plots we generated were often counterintuitive. For example, some of us expected that overtraining in Model 2 would be revealed. That is, training-set perplexity would continue to decrease as test-set perplexity would begin to rise. However, on some runs with particularly small training corpora, we observe the second iteration of Model 1 to have the lowest test-set perplexity—all subsequent iterations of Models 1, 2, and 3 made things worse (see Figure 8)! Yet we felt that the alignments were getting better. Our theory was that the machine was indeed learning useful things about the training corpus, but that as the word-translation (t) table became sharper, many of the test sentence pairs were left without "explanation." Of course, the early-stage Model 1 explanations were not particularly good—they were just the result of a fuzzier word-translation (t) table. This pattern was not observed when we trained on larger parallel corpora. In that case, better alignments led to better explanations of the previously unseen test set.

5.4 Evaluation of Translation Quality

We believe more work on evaluation is called for. There was insufficient time during the workshop to fully evaluate the quality of output from Egypt, but we record some of our thoughts here.

It is ideal that human subjects fluent in both the source language and the target language evaluate the material. Monolingual users of the target language alone are poorly prepared to evaluate the content of the translation, despite reference translations. This point is proven with the DARPA evaluations of 1994, when the quality of human translations, of variable quality, were also used for evaluation in an exam that asked evaluators to answer questions on the content of a translated passage. A monolingual evaluator is best suited for evaluating the fluency of the output, and not the content, unless the content can really be assured, such as in treaties and other legal documents where the translator has a very high incentive for ensuring that precise translations are produced.

Another concern that arises relates to the use of native speakers of the source language, and their capacity to determine the fluency of output in the target language, as well as possible biases which allow them to find meaning in "word hash" output from MT that a monolingual speaker might miss. In defining the scale for evaluation, we aim to address this concern.

Since the system is trained on texts with unique properties in terms of lexicons used, translation styles involved, and so on, we concluded that it was reasonable to draw evaluation texts from the pre-training corpus. However, we strongly emphasize that in no way is the system to be trained on texts that will later be used for evaluation. This was described in a previous section on the corpus-preparation tool Whittle.

In finding text for evaluation, there were several concerns: length, context, and novelty. Because of limits on the computational complexity of decoding and training, the corpora was stripped of sentences of length above a certain bound, originally 30 tokens. This processed corpora was then used for extracting sentences for evaluation. In addition, for the length of the entire evaluation, we suggest a testbed of 300 sentences. As 100 sentences seems like a reasonably small amount of work for evaluators, per evaluation, we decided upon 300 test sentences from the outset.

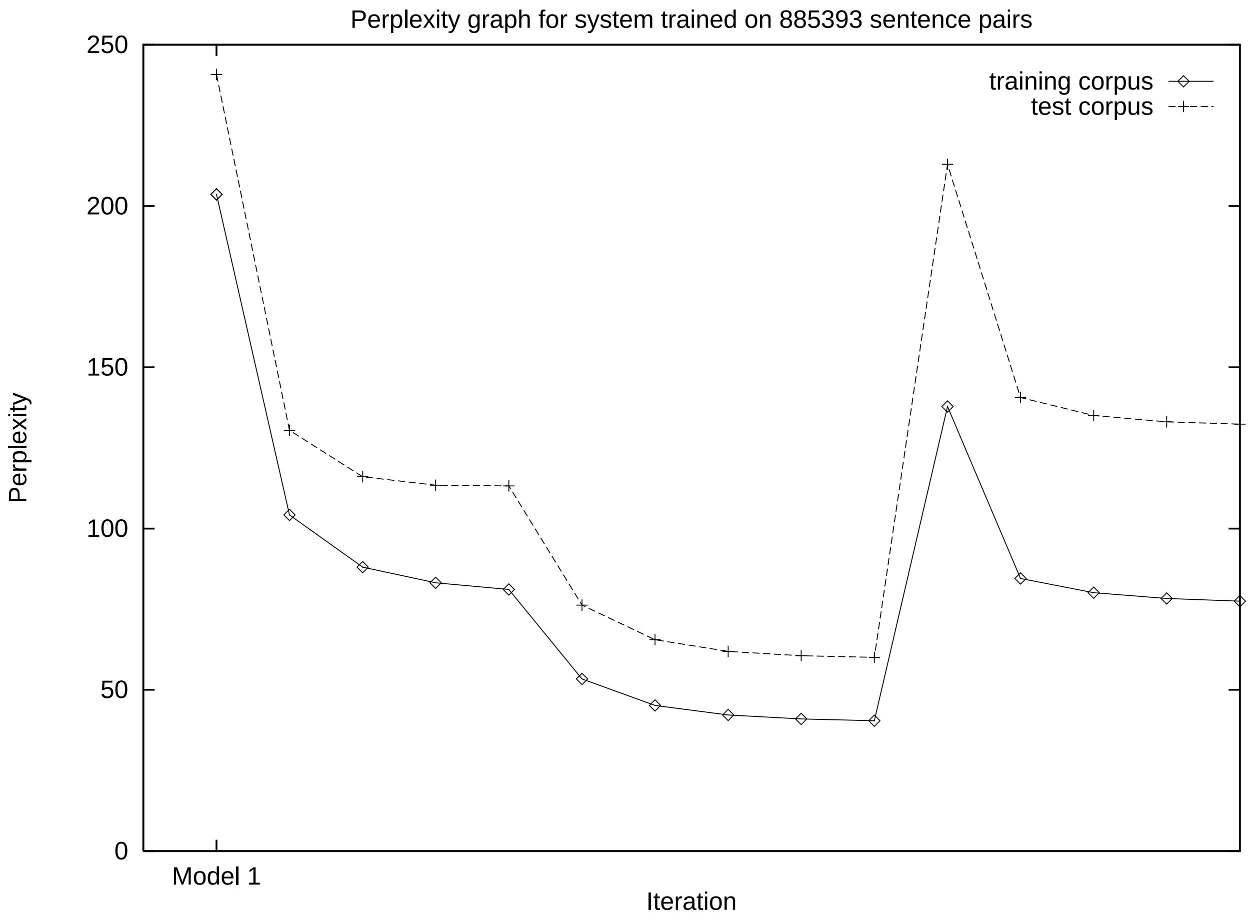

Figure 7: Perplexity as a function of EM iterations (French/English, 885,393 sentence pairs of training data).

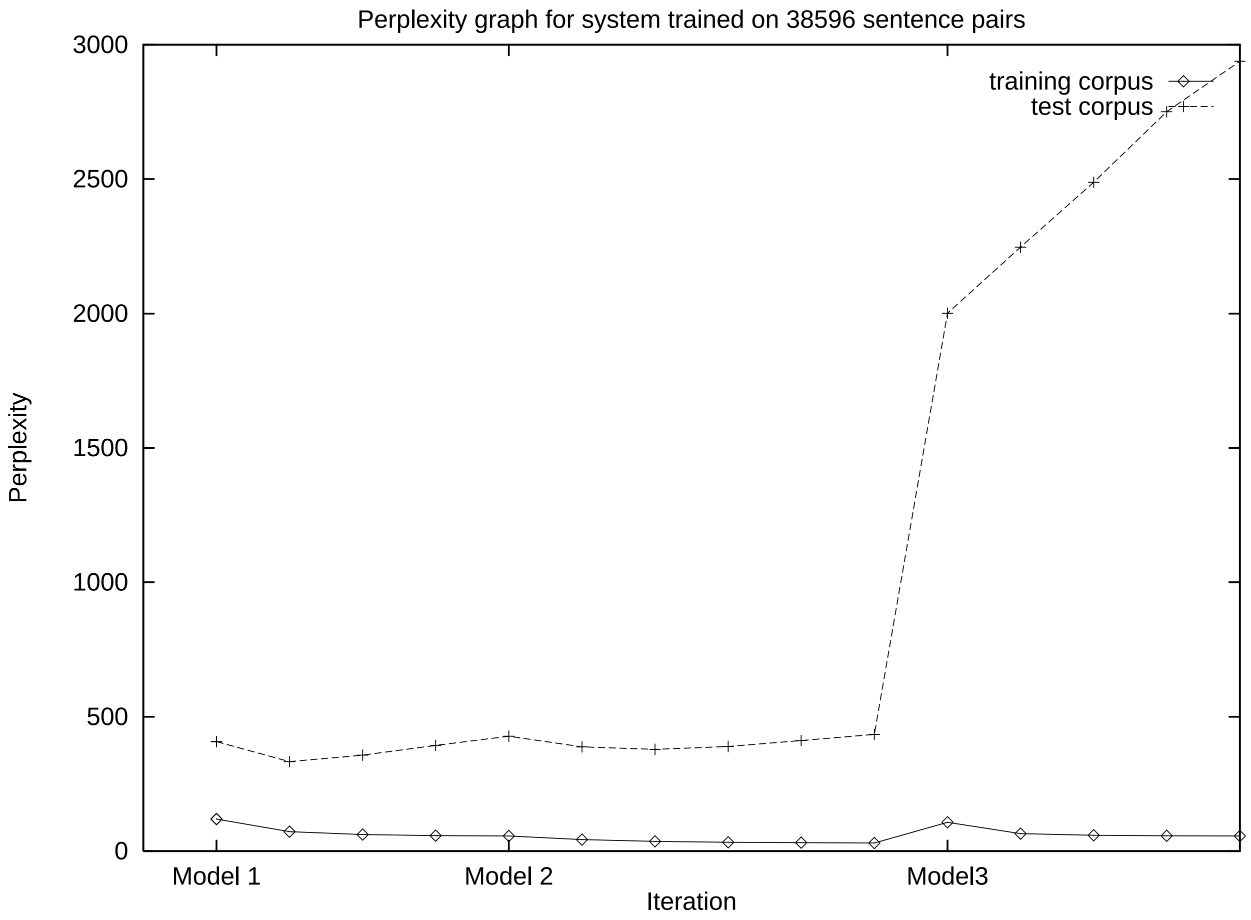

Figure 8: Perplexity as a function of EM iterations (Czech/English, 38,596 sentence pairs of training data).

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 150 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 150 sentences from an entire story removed from the corpus. Then, for the remainder of the test text, we chose 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 which operate like a venetian blind—as one allows more and more sentences into the training text, 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 perplexity of these sentences, but expect that this supposition will be validated.

For purposes of ensuring test integrity, we recommend that some subset of the test set be used for researching how to improve the MT system and the remainder of the test set be used for objective analysis of how the improvements are doing. Again, time limitations prevented us from deciding if this was a matter of concern. However, a good division of the test-set may be to use the in-context sentences for research purposes, so that one may seek to teach the MT how to learn more from the corpus, and use the out of context sentences as a function of how much real achievement has been made in MT performance.

6 Experiments with Model 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/distortion 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 Speeding up Model 3 Training

Simple algorithm

A straightforward count collection procedure for a sentence pair ($\mathbf{f}$,$\mathbf{e}$) following the description in [Brown *et al.*, 1993a] would be:[3]

1. Calculate the Viterbi alignment of Model 2: $\mathbf{a}_0 := V(\mathbf{f}|\mathbf{e};2)$, $i := 0$
2. While in the neighborhood $\mathcal{N}(\mathbf{a}_i)$ an alignment $\mathbf{a}'$ exists with $\Pr(\mathbf{a}'|\mathbf{f},\mathbf{e};3) > \Pr(\mathbf{a}_i|\mathbf{f},\mathbf{e};3)$:
   (a) Set $\mathbf{a_{i+1}}$ to the best alignment in the neighborhood.
   (b) $i := i+1$.
3. For every alignment $\mathbf{a}$ in the neighborhood $\mathcal{N}(\mathbf{a}_i)$
   (a) Calculate $p := Pr(\mathbf{a}|\mathbf{f},\mathbf{e})$.

[3] To simplify the description it is ignored the process called "pegging" which generates a bigger number of alignments considered in training. In the training program for Model 3 pegging is implemented with minor simplifications.

(b) For every $j := 1$ to m: Increase distortion counts

$$c(j|a_j, m, l; \mathbf{f}, \mathbf{e}) := c(j|a_j, m, l; \mathbf{f}, \mathbf{e}) + p$$

(c) For every $i := 1$ to l: Increase the fertility counts with p:

$$c(\phi_i|e_i; \mathbf{f}, \mathbf{e}) := c(\phi_i|e_i; \mathbf{f}, \mathbf{e}) + p$$

(d) Increase the counts for p_0 and p_1:

$$\begin{aligned} c(0; \mathbf{f}, \mathbf{e}) &:= c(0; \mathbf{f}, \mathbf{e}) + p \cdot (m - 2 \cdot \phi_0) \\ c(1; \mathbf{f}, \mathbf{e}) &:= c(1; \mathbf{f}, \mathbf{e}) + p \cdot \phi_0 \end{aligned}$$

A major part of the training time in this procedure is consumed by calculating the probability $Pr(\mathbf{a}'|\mathbf{f}, \mathbf{e})$ of an alignment $\mathbf{a}'$. This takes in general about $l + m$ operations. In [Brown *et al.*, 1993a] a simple method is mentioned to obtain $Pr(\mathbf{a}'|\mathbf{f}, \mathbf{e})$ incrementally from $Pr(\mathbf{a}|\mathbf{f}, \mathbf{e})$ if $\mathbf{a}$ differs only by moves or swaps from $\mathbf{a}'$. This trick results in a constant number of operations for calculating the score of a move or the score of a swap (see also section 4.1).

Faster hill-climbing

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

$$M_{i,j} = \frac{Pr(m_{[i,j]}(\mathbf{a}), \mathbf{f}|\mathbf{e})}{Pr(\mathbf{a}, \mathbf{f}|\mathbf{e})} \cdot [a_j \neq i]$$

for the scores of a move $a_j := i$ and in a matrix

$$S_{j,j'} = \frac{Pr(s_{[j,j']}(\mathbf{a}), \mathbf{f}|\mathbf{e})}{Pr(\mathbf{a}, \mathbf{f}|\mathbf{e})} \cdot [a_j \neq a_{j'}] \cdot [j < j']$$

for the scores of a swap of a_j and $a_{j'}$. During the hill-climbing it is necessary after doing a move or a swap 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 $a_j = i$ it is necessary to perform the following updates:

- update in M the columns j' with $a_{j'} = a_j$ or $a_{j'} = i$
- update in M the rows a_j, i
- update in S the rows and columns j' with $a_{j'} = a_j$ or $a_{j'} = i$

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

Faster count-collection

The above described simple algorithm for performing the count-collection has the disadvantage that all alignments in the neighborhood of $\mathbf{a}$ have to be enumerated explicitly. In addition for every of these alignments it is necessary to perform a loop over all English and a loop over all French positions to increase the lexicon/distortion and the fertility counts. To increase efficiency we make use of the fact that the alignments in $\mathcal{N}(\mathbf{a})$ are very similar. This allows us to share a lot of operations over these alignments.

To get efficiently the distortion and lexicon probability counts we introduce the following auxiliary quantities which make use of the move/swap matrices which are available after performing the above described hill-climbing:

- probability of all alignments in $\mathcal{N}(\mathbf{a})$

$$\begin{aligned} Pr(\mathcal{N}(\mathbf{a})|\mathbf{f},\mathbf{e}) &= \sum_{\mathbf{a}'\in\mathcal{N}(\mathbf{a})} Pr(\mathbf{a}'|\mathbf{f},\mathbf{e}) \\ &= Pr(\mathbf{a}|\mathbf{f},\mathbf{e})\cdot\left(1+\sum_{i,j} M_{i,j}+\sum_{j,j'} S_{j,j'}\right) \end{aligned}$$

- probability of all alignments in $\mathcal{N}(\mathbf{a})$ which differ in position j from $\mathbf{a}$ is:

$$\begin{aligned} Pr(\mathcal{N}_j(\mathbf{a})|\mathbf{f},\mathbf{e}) &= \sum_{\mathbf{a}'\in\mathcal{N}(\mathbf{a})} Pr(\mathbf{a}'|\mathbf{f},\mathbf{e})[a_j \neq a'_j] \\ &= Pr(\mathbf{a}|\mathbf{f},\mathbf{e})\left(\sum_i M_{i,j}+\sum_{j'}(S_{j,j'}+S_{j',j})\right) \end{aligned}$$

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

$$\begin{aligned} c(j|i;\mathbf{f},\mathbf{e}) &= \begin{cases} Pr(\mathcal{N}(\mathbf{a})|\mathbf{f},\mathbf{e}) - Pr(\mathcal{N}_j(\mathbf{a})|\mathbf{f},\mathbf{e}) & \text{if } i = a_j \\ Pr(\mathbf{a}|\mathbf{f},\mathbf{e})\left(M_{i,j}+\sum_{j'}[a_{j'}=i]\cdot(S_{j,j'}+S_{j',j})\right) & \text{if } i \neq a_j \end{cases} \\ c(f|e;\mathbf{f},\mathbf{e}) &= \sum_i \sum_j c(j|i;\mathbf{f},\mathbf{e})\cdot[f=f_j]\cdot[e=e_i] \end{aligned}$$

To get efficiently the fertility probability counts and the counts for p_0 and p_1 we introduce the following auxiliary quantities:

- probability of all alignments which have an increased fertility for position i:

$$Pr(\mathcal{N}_i^{+1}(\mathbf{a})|\mathbf{f},\mathbf{e}) = Pr(\mathbf{a}|\mathbf{f},\mathbf{e})\left(\sum_j [a_j \neq i]\cdot M_{i,j}\right)$$

- probability of all alignments which have a decreased fertility for position i:

$$Pr(\mathcal{N}_i^{-1}(\mathbf{a})|\mathbf{f},\mathbf{e}) = Pr(\mathbf{a}|\mathbf{f},\mathbf{e})\left(\sum_j [a_j = i]\sum_{i'} M_{i',j}\right)$$

- probability of all alignments which have an unchanged fertility for position i:

$$Pr(\mathcal{N}_i^{+0}(\mathbf{a})|\mathbf{f},\mathbf{e}) = Pr(\mathcal{N}(\mathbf{a})|\mathbf{f},\mathbf{e}) - Pr(\mathcal{N}_i^{+1}(\mathbf{a})|\mathbf{f},\mathbf{e}) - Pr(\mathcal{N}_i^{-1}(\mathbf{a})|\mathbf{f},\mathbf{e})$$

These quantities do not depend on swaps as a swap does not change the fertilities of an alignment. For the fertility counts holds:

$$c(\phi|e;\mathbf{e},\mathbf{f}) = \sum_i [e=e_i]\sum_k Pr(\mathcal{N}_i^{+k}(\mathbf{a})|\mathbf{f},\mathbf{e})\cdot[\phi_i + k = \phi]$$

For the fertility counts for the empty word holds:

$$\begin{aligned} c(0;\mathbf{e},\mathbf{f}) &= \sum_k Pr(\mathcal{N}_0^{+k}(\mathbf{a})|\mathbf{f},\mathbf{e})(J - 2\cdot(\phi_0+k)) \\ c(1;\mathbf{e},\mathbf{f}) &= \sum_k Pr(\mathcal{N}_0^{+k}(\mathbf{a})|\mathbf{f},\mathbf{e})(\phi_0+k) \end{aligned}$$

Using these auxiliary quantities we can formulate a count-collection algorithm which takes about $O(M^2)$ operations ($M = \max(l,m)$) which is about one order of magnitude faster than in the above described straightforward algorithm. In practice, we observed that the resulting training is about 10-20 times faster.

6.2 Working with Relative Distortion Probabilities

Motivation

A general deficiency of the models 2 and 3 is the use of absolute alignment/distortion probabilities $a(a_j|j,l,m)$ and $d(j|a_j,l,m)$ which do not take into account the fact that alignments have a strong tendency to preserve the local neighborhood. In the mathematical model of the alignment probabilities this can be captured by introducing a dependence of the previous alignment a_{j-1}:

$$\mathrm{Pr}(a_j|a_1^{j-1},f_1^{j-1},m,\mathbf{e}) \quad = \quad a(a_j|a_{j-1},l,m)$$

which results then in the so-called Hidden Markov alignment model, see [Vogel *et al.*, 1996; Dagan *et al.*, 1993]. The disadvantage of this model is that it does not model fertilities.

Simplified Model 4: Model 4'

The corresponding extension of the distortion probabilities of Model 3 to relative distortion probabilities leads to a model similar to Model 4 [Brown *et al.*, 1993a] which therefore will be called Model 4'.[4]

Relative distortion probabilities extending Model 3 are more complicated as there is no unique previous position j' in the French sentence which is aligned with position a_{j-1}. Therefore we use the difference of position j to the ceiling of the average value of the positions in the French string of the words which are aligned to the English word at position $i-1$. In Model 4', we replace $d(j|i,l,m)$ by two sets of parameters: one for placing the head of each cept, and one for placing any remaining words of the cept (for notation see [Brown *et al.*, 1993a]):

$$\begin{aligned}
\mathrm{Pr}(\Pi_{[i]1}=j|\pi_1^{[i]-1},\tau_0^l,\phi_0^l,\mathbf{e}) &= d_1(j-\odot_{i-1}) \\
\mathrm{Pr}(\Pi_{[i]k}=j|\pi_{[i]1}^{k-1},\pi_1^{[i]-1},\tau_0^l,\phi_0^l,\mathbf{e}) &= d_{>1}(j-\pi_{[i]k-1})
\end{aligned}$$

The resulting model has only $3\cdot M-2$ distortion parameters if the sentence length is restricted to M words. In Figure 9 are illustrated the dependencies of the alignment positions in Model 4.

To calculate the probability $Pr(\mathbf{f},\mathbf{a}|\mathbf{e};4)$ we have to perform the summation over all (τ,π) which produce one $(\mathbf{f},\mathbf{a})$:

$$\begin{aligned}
Pr(\mathbf{f},\mathbf{a}|\mathbf{e};4) &= \sum_{(\tau,\pi)\in(\mathbf{f},\mathbf{a})} Pr(\tau,\pi|\mathbf{e};4) \\
&= \binom{m-\phi_0}{\phi_0} {p_0}^{m-2\phi_0}{p_1}^{\phi_0}\prod_{i=1}^{l} n(\phi_i|e_i)\cdot\prod_{j=1}^{m} t(f_j|e_{a_j})\cdot \\
&\quad \prod_{j=1,a_j\neq 0}^{m}\Big([j=\mathrm{h}(a_j)]\cdot d_1(j-c_{\rho_{a_j}})+[j\neq \mathrm{h}(a_j)]\cdot d_{>1}(j-p(j))\Big)
\end{aligned}$$

with

- first French word in cept i (head of cept i):

$$h(i)=\min_k\{k:i=a_k\}$$

- previous French word in same cept as a_j:

$$p(j)=\max_{k<j}\{k:a_j=a_k\}$$

[4] In Model 4 of [Brown *et al.*, 1993a] the relative distortion probabilities depend also on word classes $\mathcal{A}$ and $\mathcal{B}$ which divide the English and French vocabulary into fifty classes. Using this refined model it is possible to model for example the effect that adjectives and nouns have different word order in French and English, but it also results in a huge number of distortion parameters which have to be estimated.

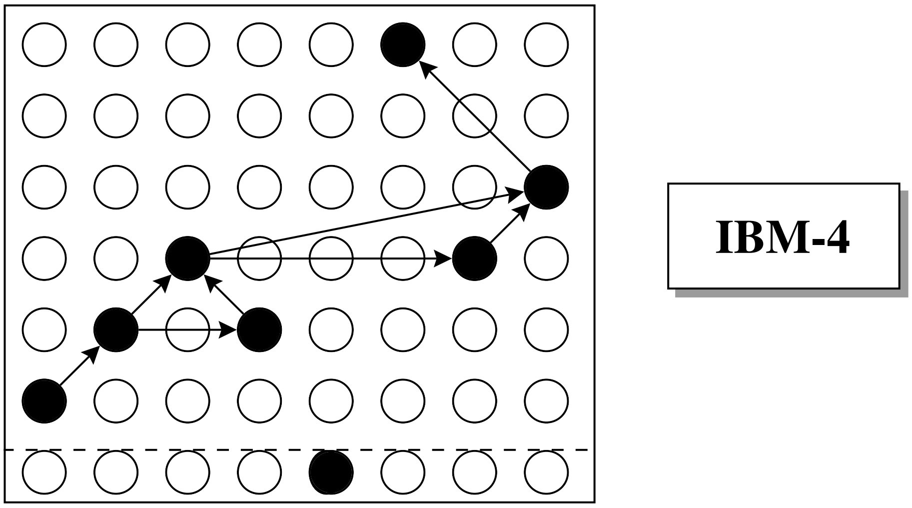

Figure 9: Dependencies of the distortion probabilities in Model 4.

- first word before e_i with non-zero fertility:

$$\rho_i := \begin{cases} \max\{i' : \phi_{i'} > 0 \land 0 < i' < i\} & \text{if } |\{i' : \phi_{i'} > 0 \land 0 < i' < i\}| > 0 \\ 0 & \text{else} \end{cases}$$

- center of cept i:

$$c_i := \frac{\sum_j [a_j = i] \cdot j}{\phi_i}$$

Note that the combinatorial factor $\prod_{i=1}^{l} \phi_i$ occurring in Model 3 is not necessary, as fixing τ and π fixes also the order of these words in the resulting French string. In [Brown *et al.*, 1993a] the value of ρ_i is undefined if there is no word before e_i with non-zero fertility. We defined $\rho_i = 0$ in this case.

Training

As in Model 3, it is too inefficient in Model 4' to consider all possible alignments. The considered alignments correspond to the alignments considered in Model 3 training. The hill-climbing procedure for calculating good alignments is significantly more time-consuming as there is now a dependence between different alignment positions.

As in Model 3, it turned out to be better to fix p_0 during the EM-training to avoid that the empty word gets aligned with too many French words.

An important point for making Model 4' training more efficient would be to implement a better incremental computation for the costs of a move/swap operation. THe simplification of the hill-climbing algorithm mentioned in [Brown *et al.*, 1993a], where the scoring of Model 3 is used to reduce the effort in scoring according to Model 4, is also not implemented.

Results

The training of Model 4' is in the current implementation about 10 times slower than the training of Model 3. Because of this only results on 50 thousand training corpus sentences (of the Hansards corpus) are reported here.

Figure 10 shows the distortion probabilities d_1 and $d_{>1}$. Because of the construction of the probabilities $d_{>1}(\Delta j) = 0$ for $\Delta j < 1$. These probabilities are much more focussed than the distortion probabilities of Model 3, which are also shown in Figure 10 for $d(j|15, 30, 30)$.

Figure 11 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.

7 Czech-English MT

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.

7.1 Resources

We had available a Czech/English corpus which is a parallel text of articles from the Reader's Digest, years 1993-1996. The Czech part is a translation of the English one. The Reader's Digest corpus consists of 53,000 sentence pairs from 450 articles. Sentence pairs were aligned automatically by [Gale and Church, 1993] algorithm, but this alignment was not sufficient for good quality alignment. Dan Melamed realigned the corpus using SIMR/GSA [Melamed, 1996] during this workshop. With language-pair-specific parameter settings learned from a small amount of word-aligned data, SIMR performance can be substantially improved; however, this experiment simply adopted French/English settings.

There was also a lot of manual work to do on this corpus before the workshop. Every issue of this magazine contains only 30-60% of articles translated from English to the local language. We had to search in the English version to find the corresponding articles that are in the Czech version. The translations in Reader's Digest are mostly very liberal. They include many constructions with direct speech. Articles with culture-specific facts have been excluded.

The tools available for Czech were: a morphological analyzer, POS tagger, and lemmatizer 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 morphologically analyzed, tagged, lemmatized, and parsed by these tools. Description of these tools are in [Hajič and Hladká, 1998; Hajič *et al.*, 1998].

There is also a Czech/English online dictionary available. This dictionary consists of 88,000 entries and covers 89% of tokens in the Czech part of 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 experiments at IFAL.

Two Czech commercial translation systems were available: PC Translator 98 and SKIK v. 4.0. We used translations by commercial systems for evaluation purposes.

7.2 About the Czech Language

Czech, as a Slavic language, is a highly inflectional and almost free word-order language. Most of the functions expressed in Czech by endings (inflection) are rendered by English word order and some function words.

For example, most Czech nouns or personal pronouns can form singular and plural forms in 7 cases. Most adjectives can form 4 genders, both numbers, 7 cases, 3 degrees of comparison, and can be either of positive or negative polarity (giving 336 possibilities for each adjective). In the corpus there are 72,000 word forms in Czech part against 31,000 forms in English.

Czech is a pro-drop language. This means that the subject pronoun (I, he, they) has usually a zero form. There are no definite and indefinite articles in Czech. English preposition equivalents can be also the part of a Czech noun or pronoun inflection. For demonstration there are 15% more tokens in English than in Czech in the corpus.

All these features create problems in translation. Our implemented translation models (IBM3, IBM4) allow only one-to-many alignments form English to Czech. Therefore it is useful to have more words in Czech

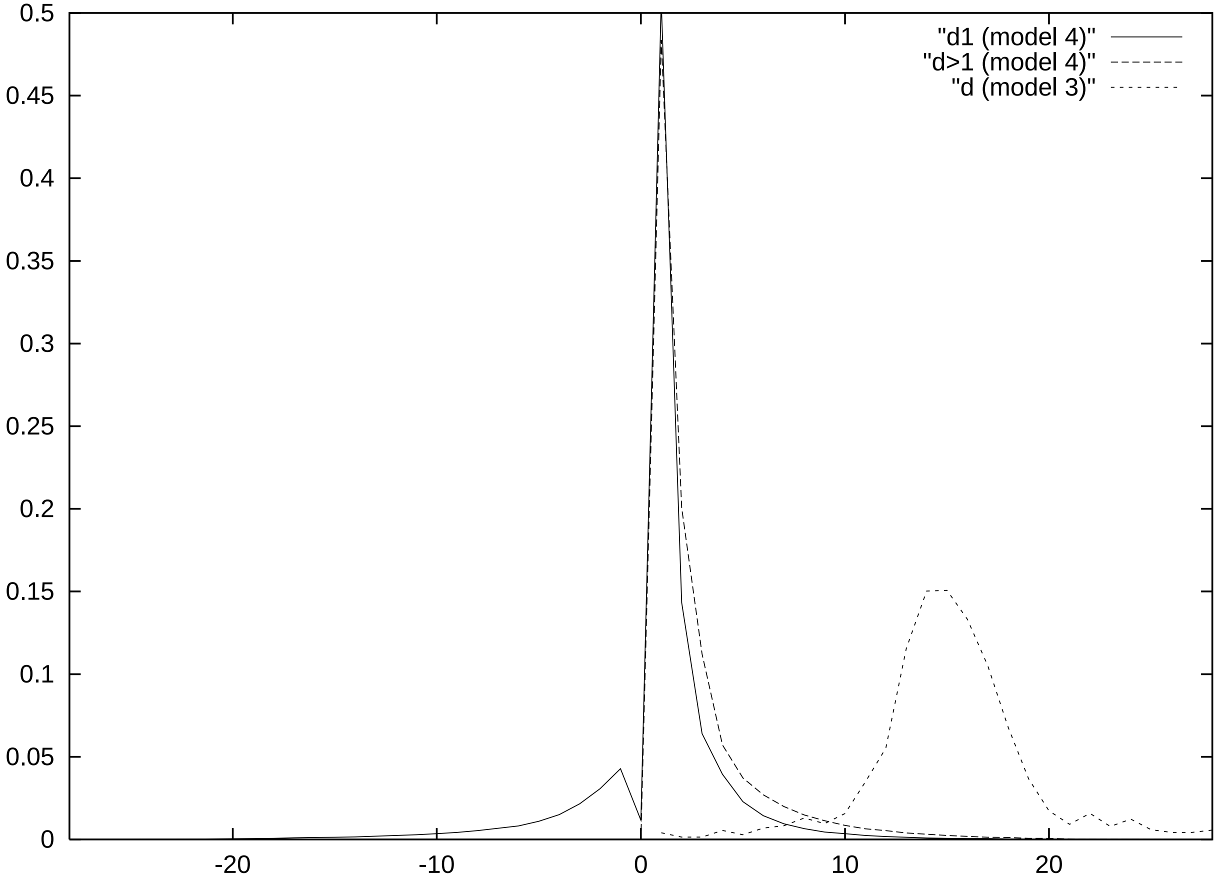

Figure 10: Distortion probabilities for Model 3 and Model 4.

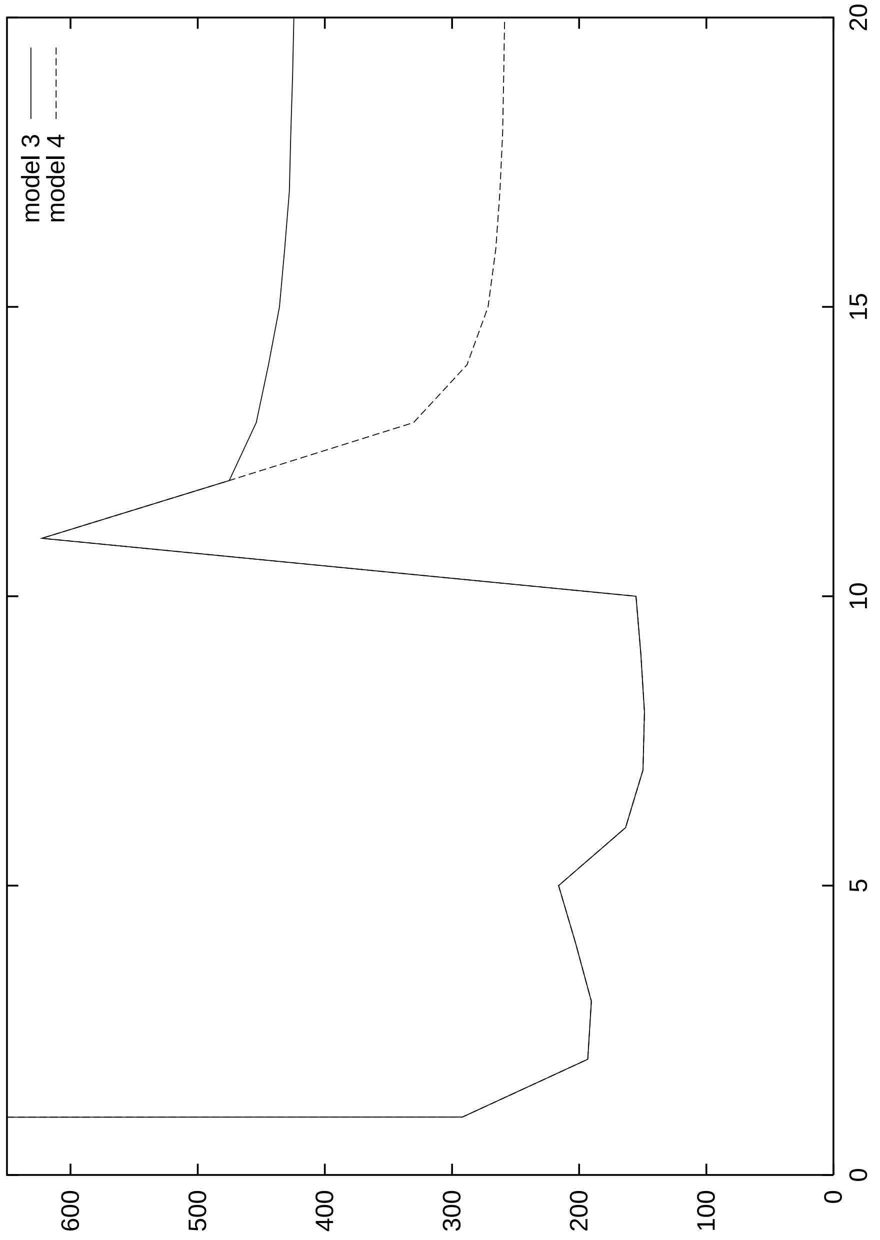

Figure 11: Test corpus perplexity during the EM-algorithm (Hansards corpus; transfer from Model 2 to Model 3 in iteration 11).

I_1 am_2 $convinced_3$ $that_4$ $[]_5$ $team_6$ $work_7$ is_8 the_9 key_{10}
for_{11} the_{12} $realization_{13}$ of_{14} $ones_{15}$ $dreams_{16}$

[I]$_1$ jsem$_2$ přesvědčen$_3$, že$_4$ *[the]*$_5$ týmová$_6$ práce$_7$ je$_8$ *[the]*$_9$ klíčem$_{10}$
ke$_{11}$ *[the]*$_{12}$ splnění$_{13}$ *[of]*$_{14}$ *[the]*$_{15}$ snů$_{16}$

Figure 12: Addition of artificial words into Czech sentence

than in English. We therefore decided to help the translation model by preprocessing Czech into English-like form—"Czech-prime," as we call it.

7.3 Tuning Czech-prime

As the training program for translation model parameters (GIZA) and decoder (from WEAVER) were in development during the workshop, we did many experiments on Czech/English translation using the Alignment Templates system developed at the University of Aachen [Och *et al.*, 1999]. This system considers whole phrases rather than single words as the basis for the alignment models. The basic idea is that a whole group of adjacent words in the source sentence may be aligned with a whole group of adjacent words in the target language. As a result the context of words has a greater influence and the changes in word order from source to target language can be learned explicitly. The Alignment Template approach was applied to some of the tasks considered during the Workshop. The aim was to provide an additional baseline for the IBM3 system and to analyze how important the modelling of word groups are for translation quality. For more details, see [Och *et al.*, 1999].

The normal Czech input containing all word forms is the baseline corpus. The next step was lemmatized Czech input. In this version of the input, we discard information about number, tense, gender and other features which are necessary to produce a useful translation. In the full Czech-prime, there is information such as number or tense attached to each lemma, which is expected to be relevant for English translation. Furthermore, artificial words are added to the Czech corpus in positions where they should appear in English.

An example of a Czech sentence with artificial words (in brackets) is given in Figure 12. Corresponding words in both languages are coindexed. There is an artificial word *[I]* for first person, singular subject, large numbers of artificial articles and an artificial preposition *[of]* corresponding to the Czech genitive. There is a potential over-generation of artificial words as you can see in position 5. This over-generation can be compensated for in the translation or language models.

Here is a description of major changes for individual parts of speech in Czech:

- nouns
  - different lemma for singular and plural
  - if the noun is not governing pronoun, the artificial article is added before the noun group (group of nouns and adjectives)
  - if the noun is in genitive, dative, locative or instrumental case, and it is not governed by a preposition in parse tree, the artificial preposition is added before the noun group
- verbs
  - different lemma for different tenses
  - if the verb is not governing a nominative noun, the artificial subject is added (artificial subjects differ for person, gender and number depending on the form of the verb)
  - special solution for auxiliary verb *to be*
  - artificial word for negative verbs
- personal pronouns
  - different lemma for singular and plural

art. word	translation	%	art. word	translation	%
NtheS	the	41.20	Vsubj1S	I	84.73
NtheS	a	22.31	Vsubj1S	my	6.59
NtheS	,	5.19	Vsubj1S	NULL	4.61
NtheS	's	4.06	Vsubj1S	me	1.28
NtheP	the	26.61	Vsubj3S	he	12.61
NtheP	of	11.68	Vsubj3S	is	11.22
NtheP	,	8.02	Vsubj3S	it	8.02
NtheP	-	6.83	Vsubj3S	NULL	7.39
Nprep2	of	36.85	Vnot	n't	27.09
Nprep2	,	16.89	Vnot	not	14.49
Nprep2	-	7.83	Vnot	NULL	12.11
Nprep2	in	5.53	Vnot	no	8.45

Table 1: Examples of alignment of artificial words in the training corpus

  - for third person, singular, there is a different lemma for masculine animate, feminine and others (*he, she, it*)
  - if the pronoun is in genitive, dative, locative or instrumental case, and it is not governed by a preposition in the parse tree, an artificial preposition is added

- other pronouns
  - different lemma for singular and plural
- adjectives and adverbs
  - artificial word for second (*more*) and third grade (*the most*)
  - artificial word for negative adjective or adverb

In Table 1 see an example of artificial words alignment in the training corpus (first 4 cases in order). We demonstrate in how many cases the artificial word is aligned to the certain word in English. The table contains artificial words for the singular article (*NtheS*), plural article (*NtheP*), genitive preposition (*Nprep2*), first person, singular subject (*Vsubj1S*), third person, singular subject (*Vsubj3S*), and negative verb (*Vnot*).

Translation models for the Alignment Templates system and the GIZA parameter estimation tool were built on the training corpus with the Czech part preprocessed as above. The test Czech sentences were preprocessed in the same way.

7.4 Evaluation

We carried out a human evaluation of translations to observe progress obtained by each level of preprocessing the Czech input. The tool for the human evaluation, which allows us to make an evaluation via Internet, was developed during the workshop. It displays the original sentence (in Czech) and translations from different translation systems. Translations are shuffled for each original sentence. Evaluators assign marks from 1 to 5 to each translation. Mark 1 is the best, mark 5 is the worst translation.

In our particular case the evaluation was done by two evaluators on 66 randomly chosen sentences from the test data. Results are in the Table 2. Average counts of assigned marks are in columns. Rows correspond to translation systems. The average value of marks assigned to each translation system is in the last column in the table.

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-prime) in comparison with the two commercial systems and the Alignment Templates system. Results on Czech/English translation using the Alignment Templates system (AlTemp) are better then one of commercial systems and almost the same as the second one.

System	1(+)	2	3	4	5(−)	**average**
Commercial 1	11	19.5	18	14.5	3	**2.68182**
Czech'+dict – AlTemp	13	18	14	17	3.5	**2.69466**
Commercial 2	7	18.5	21	16.5	3	**2.84848**
Czech' – Egypt	6	16	16.5	18	9.5	**3.13636**
Simple lemmatized – Egypt	3	12.5	23.5	23	4	**3.18939**
baseline – Egypt	3.5	6.5	13.5	27.5	15	**3.66667**

Table 2: Human evaluation of Czech/English Translation

7.5 Experiment with the Computer Oriented Corpus

The parallel corpus from Reader's Digest is relatively small. Experience from different sizes of training sets of the Canadian Hansard corpus indicates that 50,000 sentence pairs is really the basic amount of data. The results are significantly better for corpus ten times larger. Therefore, we have done an experiment on the strictly domain-specific data from IBM as well. The training set contained 1 million short sentence pairs and 10 million words in each language. The Alignment Template system was used to train a translation model and to translate the test part of corpus.

Almost 34% of the sentences from the testing data were translated exactly the same as in the reference set. According to human evaluation on 56 randomly chosen sentences from testing corpus, another 30% of sentences were excellent translations, 11% were good or acceptable, 8% of translations had bad word order, and 17% of translations were bad.

By comparison, in the Reader's Digest test corpus, only 1.42% of translated sentences are exactly the same as their reference translations.

7.6 Conclusion

We carried out the first experiments on statistical machine translation from Czech to English. We can observe how the progress of translation quality depends on the preprocessing of the Czech input. The Reader's Digest corpus output from the Alignment Template system is comparable with translations by commercial systems. The results of the Alignment Template system and the Egypt system are not directly comparable as in the Egypt system the dictionary was not used as a knowledge source. In addition, in the development of the procedure of transforming Czech to Czech-prime we used the Alignment Template system to gather knowledge about problematic constructions. This may have led to a bias in favor of the Alignment Template system. Nevertheless it seems to be possible to conclude from these results that modelling word-groups in source and target language (as done in Alignment Templates) is important.

The results reached for the technical computer oriented corpus are very good and promising. Larger amount of data can significantly help the system. As the general translation tool has been just developed, it is now possible to experiment with different system parameters, such as the number of iterations of particular models, and to adjust the translation models to better suit the Czech/English language pair.

8 Using Online Bilingual Dictionaries and Cognates in Training

Intuition tells us that a bilingual dictionary could be used to improve the Model 1 translation model. This model consists essentially of a function t such that $t(e, f) = Pr(e \mid f)$ for all words e in the output language and all words f in the input language. A dictionary provides precisely these mappings. Utilization of dictionary information is further motivated by the fact that the word alignments chosen by our baseline Model 3 training display a tendency to use rare words as "garbage collectors." Such words tend to align to many common function words. Dictionary entries for these words could be used to curb this tendency. It is possible that dictionary information could also slow or prevent overtraining. We experimented with three methods of adding a bilingual dictionary to the training of the Model 3 translation system. The first was to seed the t function before training, giving more weight to dictionary pairs. The second was simply to add the dictionary to the training corpus. Lastly, we modified the Model 1 algorithm, redefining "co-occurrence" [Melamed, 1998b] to make use of dictionary information.

For all of our experiments, we used a corpus of 50,000 total Czech-English aligned sentence pairs. The Czech was lemmatized and included morphological markers. The dictionary entries were lemmatized using the same lemmatization scheme as the corpus. The training corpus contained 38842 sentences, 555,714 English tokens (27,393 unique), 665,268 Czech tokens (28,429 unique), and 2,768,252 unique word co-occurrence pairs. The Czech-English dictionary consisted of 204,110 entries of one Czech word to one or more English words.

While these experiments were aimed at improving Model 1's performance, the improvement was most noticeable after Model 2 and 3 training. This suggests that the global optimization of parameters in Model 1 training has been modified.

8.1 Experiment 1: Seeding the Translation Parameters

GIZA's Model 1 training begins with a uniformly distributed t function:

$$\forall w_{Cz} \in Vocabulary_{Cz}, w_{En} \in Vocabulary_{En}, t(w_{Cz} \mid w_{En}) = \frac{1}{\mid Vocabulary_{Cz} \mid}$$

We experimented with seeding the t function so that word pairs (w_{Cz}, w_{En}) receive a factor of C greater probability than other pairs. The value of C was set at 2, 4, 8, 16, 32, 64, and ∞ (*i.e.*, non-dictionary pairs received an initial t value of 0).

Because the t 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 resulting dictionary contained 22,915 one-to-one entries.

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

Results of Seeding.

Value of C	Final test perplexity
1 (baseline)	1986.52
2	1974.49
4	1949.16
8	1905.78
16	1879.21
32	1848.84
64	1833.52
∞	1649.00

It is clear that, while seeding the t function using the dictionary improves training (assuming the validity of the perplexity measure), there is a limit to this improvement. This is explained easily: the information that is given to the algorithm in the seeding step is "washed out" with successive iterations of training.

Two approaches to counter this are to apply a stronger filter when seeding and to make the dictionary part of the training data. Before we discuss these experiments, we will explain related work that we did to enlarge the dictionary using cognates.

8.2 Finding Cognates

Tiedemann describes three methods of extracting a string similarity measure that is suitable for the recognition of cognates in two specific languages [Tiedemann, 1999]. These methods are based on the longest common subsequence ratio (LCSR) measure of string similarity [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. The 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 7/10 and 10/7 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.

We then scored all word co-occurrence pairs from the training corpus. The result was a list of pairs of words (one Czech and one English) that "looked alike" based on transformations that tend to occur when a word is borrowed across the two languages. An example of this is that English words containing the character 'c' often correspond to a Czech translation that substitutes 'k'. These language-specific tendencies are picked up by the weighting function for the string similarity measure.

The following table shows the number of co-occurring word pairs that scored above each of the thresholds. The intuitive threshold that separates word pairs that actually do look alike from ones that don't appears to be at a score of about 0.3. Note that the pairs which were "not scored" failed to pass through the length ratio or minimum length filters described above. The total number of co-occurrence pairs was 2,768,252.

Distribution of Similarity Scores.

Score range	Number of pairs	Cumulative percentage
0.550000–0.623911	37	0.0013
0.500000–0.549999	193	0.0083
0.450000–0.499999	733	0.0348
0.400000–0.449999	1,764	0.0985
0.350000–0.399999	1,959	0.1693
0.300000–0.349999	2,840	0.2719
0.250000–0.299999	7,775	0.5527
0.200000–0.249999	35,565	1.8375
0.150000–0.199999	114,114	5.9597
0.100000–0.149999	262,515	15.4428
0.050000–0.099999	277,103	25.4528
0–0.049999	176,901	31.8432
Not scored	1,886,753	100.0000

Using the hypothesized cognates, we can now augment the dictionary. The hope is that alignments between rare words that "look alike" will improve the performance of a model which uses (in one way or another) this dictionary.

From this point on, we will use the term *dictionary* to refer to the online dictionary that we used in Experiment 1. A *lexicon* will refer to any listing of word pairs that are known or believed *a priori* to be translations of each other. The cognate lexicons described here were used in the next two experiments, both individually and merged with the dictionary.

8.3 Experiment 2: Constraining Co-occurrence

In Candide (GIZA) Model 1 training, the t function is based on co-occurrence data. A pair of tokens is said to co-occur once for every time 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 lexicon and parallel sentences $E = \{e_1, e_2, ..., e_n\}$ and $C = \{c_1, c_2, ..., c_m\}$, e_i and c_j co-occur if *either* (e_i, c_j) is in the lexicon *or* (e_i, c_k) is not in the lexicon $\forall k$ and (e_p, c_j) is not in the lexicon $\forall p$."

Using this co-occurrence constraint in the first iteration of Model 1 training is akin to seeding in a greedier fashion. We used GIZA to create translation models, with this constraint and numerous lexicons. The lexicons used included the original dictionary (as prepared for Experiment 1), the various thresholded cognate pairs (ranging from 0.05 to 0.55), and merged lists of cognate pairs and the dictionary. The results are shown below.

Results of Constraining Co-occurrence.

Lexicon	Final test perplexity
Dictionary + Cognates $\geq$ 0.05	1596.89
Dictionary + Cognates $\geq$ 0.10	1533.51
Dictionary + Cognates $\geq$ 0.15	1478.24
Dictionary + Cognates $\geq$ 0.20	1464.62
Dictionary + Cognates $\geq$ 0.25	1491.43
Dictionary + Cognates $\geq$ 0.30	1510.44
Dictionary + Cognates $\geq$ 0.35	1510.24
Dictionary + Cognates $\geq$ 0.40	1535.83
Dictionary + Cognates $\geq$ 0.45	1560.68
Dictionary + Cognates $\geq$ 0.50	1559.60
Dictionary + Cognates $\geq$ 0.55	1562.23
Dictionary	1563.32
Cognates $\geq$ 0.05	1767.05
Cognates $\geq$ 0.10	1740.99
Cognates $\geq$ 0.15	1797.04
Cognates $\geq$ 0.20	1848.84
Cognates $\geq$ 0.25	1847.05
Cognates $\geq$ 0.30	1880.77
Cognates $\geq$ 0.35	1885.75
Cognates $\geq$ 0.40	1909.45
Cognates $\geq$ 0.45	1953.93
Cognates $\geq$ 0.50	1960.98
Cognates $\geq$ 0.55	1985.35
None (baseline)	1986.52

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 0.20 is used. It was noted previously that the intuitive threshold is at about 0.30; this indicates that even boosting pairs which are not cognates may aid in the alignment process. Also, note that a significant improvement comes even when only cognates are used, especially when the threshold is as low as 0.10. This suggests that even flawed, conjectural information about cognates can improve word alignment in this type of model using the co-occurrence constraint.

In addition to test perplexity, we examined some Viterbi word alignments of test corpus sentences which these models posited. An example sentence pair follows. Note that the Czech sentence has been lemmatized and certain morphological markers have been inserted.

> "It also unleashed a gigantic seismic sea wave, or tsunami, that killed people as far away as California."

"VR zpusobit též NS1 vsnik obrovský mořský NS2 vlna, který VR; zbíjet až v NS3 Kalifornie."

The baseline model built by GIZA made the following alignments (a star indicates an incorrect alignment):

English	*Czech*
∗also	∅
∗gigantic	zpusobit
∗seismic	vznik obrovský seismický
sea	mořský
wave	vlna
∗tsunami	který
∗that	∅
far	až
as	v
California	NS3 Kalifornie

In contrast, when the constraint described above is used (with the dictionary and cognates $\geq .15$ as the lexicon), the model produces these alignments (a star indicates an incorrect alignment):

English	*Czech*
∗also	zpusobit
gigantic	obrovský
seismic	seismický
sea	mořský
wave	vlna
tsunami	∅
that	který
far	až
∗away	NS3
as	v
∗California	Kalifornie

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

8.4 Experiment 3: Appending a Lexicon to the Corpus

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.*, 1993b]. Each lexicon entry is added to the training corpus. The advantages of this approach are numerous: multiple-word entries can be handled easily (*i.e.*, each entry is like a sentence pair—there are no one-to-one requirements as in seeding); the information remains present throughout all iterations of training. Also it is possible to weight the entries strategically. In a normal situation, such a weighting of sentence pairs in the corpus indicates the number of times that each sentence pair occurs. 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 to the corpus; the second was to add only those entries which consisted of items that appeared as co-occurrences in the corpus. The latter of these two condenses the original dictionary to only 15,490 entries. Note that this option applies only when the original dictionary is used; the cognates were gleaned from the corpus and as such are known to be co-occurrences in the corpus. We also varied the weighting function. The three we tested initially were: uniform (w_1), inverse linear (w_2), and inverse exponential (w_3), all defined below:

$$w_1(entry) = C, C \in \{0.1, 1, 2, 4, 8, 16, 32, 64, 128\}$$

$$w_2(entry) = \frac{1}{frequency(entry)} \cdot \frac{nTokens}{nUnique}$$

$$w_3(entry) = e^{-frequency(entry) \cdot \frac{nUnique}{nTokens}}$$

Surprisingly, after initial trials, the first weighting function with C set to 1 gave the greatest improvement. A smooth value of 0.001 was selected as the minimum weight any entry would receive (several others were tried, but this value resulted in the greatest improvement in test perplexity). Because of the huge number of tests we did using this technique, only trials using the first weighting function (uniform) and the smooth value 0.001 will be discussed.

The results of this experiment are the most promising. The test sentence Viterbi alignments show improvements similar to those in Experiment 2. The alignments appeared to be most 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.

Results of Lexical Appension ($C = 1$).

Lexicon	Final test perplexity
Dictionary (all) + Cognates ≥ 0.05	1342.76
Dictionary (all) + Cognates ≥ 0.10	1173.86
Dictionary (all) + Cognates ≥ 0.15	1118.13
Dictionary (all) + Cognates ≥ 0.20	1159.05
Dictionary (all) + Cognates ≥ 0.25	1222.38
Dictionary (all) + Cognates ≥ 0.30	1242.85
Dictionary (all) + Cognates ≥ 0.35	1262.06
Dictionary (all) + Cognates ≥ 0.40	1279.03
Dictionary (all) + Cognates ≥ 0.45	1198.94
Dictionary (all)	1444.88
Dictionary (co-occurring pairs)	1588.60
Cognates ≥ 0.05	1614.30
Cognates ≥ 0.10	1450.13
Cognates ≥ 0.15	1440.25
Cognates ≥ 0.20	1624.73
Cognates ≥ 0.25	1772.58
Cognates ≥ 0.30	1787.76
Cognates ≥ 0.35	1843.45
Cognates ≥ 0.40	1869.95
Cognates ≥ 0.45	1952.54
None (baseline)	1986.52

It is interesting to note that the greatest improvement from cognates came when extremely low thresholds were used (0.15). Also, when only cognates were used and the threshold was set low enough, the improvement was actually better than that gained when only the dictionary was used. The relative benefit from using all dictionary pairs as opposed to only those which co-occur is easily explained; a English word may have a synonym which co-occurs with a Czech word in the test corpus but not the training corpus.

8.5 Conclusion

These experiments 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, cheap and imperfect methods of finding cognates appear to provide improvement over baseline models and models using only a dictionary.

9 "MT in a Day" Experiment

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 would 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 in 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 previously unseen foreign-language sentences

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

- implement significant linguistic transductions to improve statistical modeling
- use dictionaries and cognates to further improve models
- evaluate translations using human judges

We decided on Chinese/English as our test language pair. We considered other language pairs, and we also 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 an English. The translations in this corpus are quite literal compared to the translations in the more typical French/English Hansard corpus, which covers parliamentary debate. The corpus has 7 million words on the English side.

We planned no linguistic significant transduction work in the 24-hour experiment. We therefore re-used our existing English tokenizer to break English sentences into words.

For Chinese, we decided to break each sentence into characters rather than words. We hoped to learn translation models for each English word that would map it onto various Chinese characters, and furthermore assign it fertility values to specify (probabilistically) how many characters it might map into during translation.

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 Chinese side.

We used our standard corpus-preparation tool (Whittle) to divide the corpus into training in testing parts. This tool also throws out sentences that exceed a specified length—in this case, we threw out any sentence pair with an English or Chinese sentence longer than 30 tokens. This resulted in 102,707 training sentence pairs, and a held-out database of 1960 testing sentence pairs.

Previous experiments with Czech and English had revealed robustness problems in our training code. In particular, if a very short sentence was paired with a very long sentence (perhaps by mistake), segmentation faults could occur. We had already adjusted our training code to throw out such pairs, and so this potential problem was already solved by the time we did our Chinese experiments (where such pairs also occurred). This resulted in a total of 80,769 usable training sentence pairs.

Using Whittle, we also produced smaller training corpora of 40,427 and 59,476 sentence pairs. All three corpora were processed by our translation modeling system GIZA without any problems. We measured test-set perplexity on all three:

Training Corpus Size	Test-Set Perplexity
40,427	859.9
59,476	611.7
80,769	489.0

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.

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

Finally, we used the Weaver decoder to translate 100 Chinese test sentences. The final translations were generated at 11:30pm on the day in question, and the final five translations were:

HUMAN: shall be guilty of an offence and shall be liable on summary conviction to a fine of $ 1000
EGYPT: commits an offence and liable on summary conviction on indictment , to a fine of $ 1000

HUMAN: (2) in this section , " authority " (@@@) includes any authorized person
EGYPT: (2) in this section , " authority " (@@@) means any authorized person

HUMAN: (b) it appears that the testator intended by his signature to give effect to the will
EGYPT: (b) purports wills is sought its signed it will effect

HUMAN: (a) which does not amount to a contract to purchase those shares ;
EGYPT: (a) the contract not constitute the likely to purchase those shares contract ;

HUMAN: (a) a company authorized under section 8 to carry on that class of insurance business
EGYPT: (a) under section 8 of the authorized undertaking class insurance business company

We subsequently did a quick low-level, word-level analysis of some of these results. Splitting Chinese into single characters turned out to have many strengths, and also some weaknesses. The words "Hong" and "Kong" were translated into separate characters. Had we done more linguistic analysis, the two characters might have formed a single token. In that case, we might have had some problems, due to the asymmetric nature of our translation model (IBM Model 3). In this model, each English word can produce zero or more Chinese tokens, but it is difficult for to English words to collaborate to produce a single Chinese token. This would not be a show-stopper, however, as "Hong" can take responsibility for the Chinese word, and "Kong" can be assigned a learned fertility of zero. (It is then up to the language model to notice a floating "Hong" and insert a "Kong.") In any case, remedying this situation with a more symmetric phrase-based translation model is part of the funded follow-on proposal by Franz-Josef Och.

We also noticed that the word like "June" had several character translations, including the character for "month" and character for "six," as expected. It was typically assigned a fertility of 3, also grabbing the character for "day"—which we expected to be produced by the day of the month, such as "8" in "June 8." Date translations seemed to work fine, however. To get a more intuitive alignment, we should use the relative distortions of Model 4, also part of the proposed follow-on work.

Finally, we noticed that some decodings would translate two Chinese characters (rather literally) as two English words, when the correct answer was a single English word. In these cases, the language model was failing to enjoy the single-word translation enough.

Over the next three days, we did some work to compare character-based tokenization with word-based tokenization, experimenting with some of the behaviors just described. We also located another substantial Chinese/English corpus, of the nature similar to the parliamentary debates of the Hansard corpus, and we were able to obtain permission to use and distribute it for research purposes. In the future, it would be good to perform some of the experiments described above (phrases, Model 4), to experiment with larger combined corpora, to make use of the Chinese/English bilingual dictionary that we obtained, and to evaluate translations using human judges, as these activities were not within the scope of the 24-hour project.

10 Future Work

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 up into the realms of syntax and semantics. We will not anticipate them here. However, we know of certain enhancements that would improve the immediate usefulness of Egypt, so we list these:

- Context-dependent translation probabilities. [Brown *et al.*, 1991; Berger *et al.*, 1996; Melamed, 1998a] describe techniques for assigning "senses" to words and/or translating them differently based on context. This would relieve substantial amounts of pressure currently on the language model to disambiguate words in the input text.
- Faster training. We have found speedy training to be a boon for rapid experimentation and debugging, but much remains to be done. Currently, Model 3 training (with no pegging) is fast, but pegging slows things down quite a bit. Also, Model 4 is slow to train, with or without pegging.
- English morphology. While 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 frequently possible to locate word sequences that translates as a whole (perhaps non-compositionally). It would be useful for translation models to uncover these and for decoding to take these into account (see [Och *et al.*, 1999; Melamed, 1997]).
- Better initial alignments. Models 3, 4, and 5 collect their counts over a small subset alignments, so it is critical that the simple models used to bootstrap them are accurate, particularly with small data sets (see [Melamed, 1998a]).

11 Acknowledgements

Thanks to staff of IFAL, Charles University, Prague, especially to Jan Hajič and Barbora Hladká for providing tools for Czech morphological analysis, tagging and lemmatization, and to Michael Collins for the possibility to use his statistical parser. Thanks to Martin Čmejrek for collaboration on parallel Czech/English data preparation and to Lenka Kadlčáková and Martin Čmejrek for the prompt evaluation of translations. Special thanks to Reader's Digest Výběr (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 ihs maximum entropy tagger, and the RALI group at the University of Montreal for permission to experiment with hand-tagged French data.

Thanks to West Group for supporting Dan Melamed's participation in the workshop. Also, thanks to the Lehrstuhl fuer Informatik VI at the RWTH Aachen for supporting Franz Josef Och's participation.

Thanks to Yolanda, Angela, and Victoria, and to frequent-flyer programs everywhere.

12 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. His research interests include: Natural Language Processing, Statistical Machine Translation, and Machine Learning.

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 natural language processing, and his main fields of activity are machine translation and parallel-corpora processing.

Michael Jahr is a junior studying computer science at Stanford University. He graduated from the Karachi American School in Karachi, Pakistan in 1996. He is fluent in French, and his research experience includes working with cognitive psychologists at Stanford and with the generation and interpretation of digital maps at Daimler-Benz Research and Technology Center.

Kevin Knight is a Senior Research Scientist at the USC/Information Sciences Institute. He is also a Research Assistant Professor in the Computer Science Department at USC. He received his Ph.D. from Carnegie Mellon University in 1991 and his BA from Harvard University in 1986. He is co-author of the textbook Artificial Intelligence (McGraw-Hill, 1991).

John Lafferty is an Associate Professor at Carnegie Mellon University in the Computer Science Department and Language Technologies Institute. He is also an affiliated faculty member of CMU's Center for Automated Learning and Discovery and the Program in Algorithms, Combinatorics, and Optimization. His research interests include statistical learning algorithms, language modeling, information theory, codes and graphical models, and natural language applications.

Dan Melamed is a research scientist in the Computer Science Research Department of West Group. His current research interests include the theory and applications of statistical translation models and various other applications of statistical NLP. The MIT Press will publish his *Empirical Methods for Exploiting Parallel Texts* in 2000, and most of his other publications are downloadable from `http://www.cis.upenn.edu/~melamed`.

Franz-Josef Och studied computer science in the University of Erlangen and at the University of Bologna. He received his Diplom in computer science with honours in the beginning of 1998 from the University of Erlangen. Since then he is a graduate student at the Lehrstuhl für Informatik VI at the Technical University RWTH Aachen. His scientific interests include natural language processing, statistical machine translation and machine learning.

David Purdy works at the Department of Defense, doing research in natural language processing. He graduated from Harvard with honors in June 1999, with a BA in mathematics; his senior thesis was under Professor Noam Elkies, on finding rational elliptic curves with high rank.

Noah A. Smith is a rising junior at the University of Maryland at College Park, pursuing a double degree in Computer Science and Linguistics; he is a native Marylander. He will be studying at the University of Edinburgh in Scotland in the spring of 2000. His interests are in theoretical linguistics, mainly syntax, and natural language processing, particularly machine translation and information retrieval.

David Yarowsky is an Assistant Professor in the Department of Computer Science at Johns Hopkins University and a member of the Center for Language and Speech Processing. He received his masters and Ph.D. from the University of Pennsylvania, and his AB from Harvard University. His research interests include natural language processing and spoken language systems, machine translation, information retrieval, very large text databases, and machine learning. He serves on the editorial board of *Computational Linguistics*, is Secretary-Treasurer of SIGDAT (ACL special interest group for statistical/data-driven methods in natural language), and was program chair for the Third Workshop on Very Large Corpora.

References

Alshawi, H., A. Buchsbaum, and F. Xia. 1997. A comparison of head transducers and transfer for a limited domain translation applications. In *Proc. ACL*.

Berger, A., P. Brown, S. Della-Pietra, V. Della-Pietra, J. Gillett, J. Lafferty, R. Mercer, H. Printz, and L. Ures. 1994. The Candide system for machine translation. In *Proceedings of the ARPA Human Language Technology Workshop*.

Berger, A., S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics* 22(1).

Brown, P., S. Della Pietra, V. Della Pietra, and R. Mercer. 1991. Word-sense disambiguation using statistical methods. In *Proc. ACL*.

Brown, P. F., V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics* 19(2).

Brown, Peter, Stephen Della Pietra, Vincent Della Pietra, Meredith Goldsmith, Jan Hajic, Robert Mercer, and Surya Mohanty. 1993. But dictionaries are data too. In *Proceedings of the ARPA Human Language Technology Workshop.*

Dagan, I., K. Church, and W. Gale. 1993. Robust word alignment for machine aided translation. In *Proceedings of the Workshop on Very Large Corpora: Academic and Industrial Perspectives.*

Florian, R. and D. Yarowsky. 1999. Dynamic non-local language modeling via hierarchical topic-based adaptation. In *Proc. ACL.*

Gale, W. and K. Church. 1993. A program for aligning sentences in bilingual corpora. *Computational Linguistics* 19(1).

Hajič, Jan, Eric Brill, Michael Collins, Barbora Hladká, Douglas Jones, Cynthia Kuo, Lance Ramshaw, Oren Schwartz, Christoph Tillmann, and Daniel Zeman. 1998. *Core Natural Language Processing Technology Applicable to Multiple Languages (Final Report, Summer Workshop'98).* Tech. Rep. ?, Center for Speech and Language Processing, Johns Hopkins University.

Hajič, Jan and Barbora Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of Coling/ACL.*

Knight, K. 1997. Automating knowledge acquisition for machine translation. *AI Magazine* 18(4).

Knight, K. 1999. *A Statistical Machine Translation Tutorial Workbook.* Tech. Rep. ?, USC/ISI. (available at http://www.clsp.jhu.edu/ws99/projects/mt/wkbk.rtf).

Knight, K. and Y. Al-Onaizan. 1998. Translation with finite-state devices. In *Proc. AMTA.*

Melamed, I. Dan. 1995. Automatic evaluation and uniform filter cascades for inducing n-best translation lexicons. In *Proceedings of the Third Workshop on Very Large Corpora.*

Melamed, I. Dan. 1996. A geometric approach to mapping bitext correspondence. In *Proceedings of the First Conference on Empirical Methods in Natural Language Processing.*

Melamed, I. Dan. 1997. Automatic discovery of non-compositional compounds. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing.*

Melamed, I. Dan. 1998a. *Empirical Methods for Exploiting Parallel Texts.* PhD thesis, University of Pennsylvania.

Melamed, I. Dan. 1998b. *Models of Co-occurrence.* Tech. Rep. 98-05, IRCS, University of Pennsylvania.

Och, F. J., C. Tillmann, and H. Ney. 1999. Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora.*

Resnik, P. 1999. Mining the web for bilingual text. In *Proc. ACL.*

Su, K.-Y., J.-S. Chang, and Y.-L. U. Hsu. 1995. A corpus-based two-way design for parameterized mt systems: Rationale, architecture and training issues. In *Proc. TMI.*

Tiedemann, Jorg. 1999. Automatic construction of weighted string similarity measures. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.*

Vogel, S., H. Ney, and C. Tillmann. 1996. Hmm based word alignment in statistical translation. In *Proc. of the 16th Int. Conf. on Computational Linguistics.*

White, John and Teri O'Connell. 1994. Evaluation in the ARPA machine translation program: 1993 methodology. In *Proceedings of the ARPA Human Language Technology Workshop.*

Wu, D. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics* 23(3).