Arabic Computational Linguistics: Current Implementations

A. Farghaly, Editor

May 23, 2006

CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION

Contents

1

The Language Weaver Arabic to English Statistical Machine Translation Software System

Alexander Fraser, William Wong

1.1 Introduction

1.1.1 Motivation

In the Spring of 2003, Language Weaver Inc. began developing Arabic to English Statistical Machine Translation Software (SMTS). There were several strong reasons why this language pair and methodology were chosen. The intent of developing this software was to validate the statistical approach to Machine Translation (MT) in the form of a commercial product, and it was judged that an Arabic to English MT system would achieve this. Arabic to English MT was also perceived as a good market opportunity. Available MT solutions at the time did not produce output at a high enough quality level to be useful for many applications. Another reason to release such a product was the need for a translation engine which could be integrated with automatic speech recognition and optical character recognition, both of which output imperfect Arabic text which then forms the input to the machine translation system. Statistical Machine Translation approaches appear to be more robust to problems with the quality of the input text than rule-based approaches, degrading smoothly as the quality of the input text decreases. All of these factors contributed to Language Weaver's decision to allocate significant resources for the creation of its first flagship product, Arabic to English SMTS. Language Weaver's Arabic to

Arabic Computational Linguistics: Current Implementations.
A. Farghaly, Editor.

English Statistical Machine Translation Software was the first statistical MT product to reach the market.

1.1.2 Language Weaver Technology Background

Statistical Machine Translation

Machine translation research has progressed in recent years with the development of new techniques for automatically learning rules from large corpora of translated documents, an approach which is sometimes referred to as data-driven Machine Translation or Example-Based Machine Translation. Statistical Machine Translation (SMT) is a group of approaches based on probabilistic modeling which have met with considerable success. SMT systems have had the best performance in the NIST Machine Translation evaluations[1] every year from 2001 to the present, and were the best performers in the German Verbmobil evaluation Wahlster (2000).

In the SMT approach, Brown et al. (1993), a large corpus of documents and their translations are automatically analyzed and a “word alignment”, which shows translational correspondence of words, is automatically generated for every sentence and its translation. The probabilities of alternative translations as observed in the word alignment are estimated, and a “translation model” is created. A “language model”, which is a statistical model of well-formed sentences in the target language, is used to find grammatical output in the translations proposed by the translation model. After these models are estimated, they are stored on disk. When new source language text arrives for translation, a search process is invoked to automatically determine the best hypothesis according to both the translation model and the language model. This is conceptually similar to considering a very large number of rules, each of which covers a small piece of the input sentence, to see which ones work well together (i.e., which of the rules, which cover the input sentence, generate good target language sentences).

Details on statistical models and estimation of their parameters are provided in Section 1.1.5. Example entries in the translation lexicon are presented in Section 1.1.5.

Comparison with Rule-Based Approach

The Statistical Machine Translation approach forms a strong contrast with traditional rule-based approaches. The rule-based approach requires a large level of effort by highly trained linguists for each translation direction implemented. Linguists analyze a relatively small amount of parallel data and write rules which reproduce it perfectly. The dif-

[1] `http://www.nist.gov/speech/tests/mt/`

ficulty is in creating rules which generalize well, to apply to a large variety of inputs texts, while still producing output which is correct. A rule-based system may sometimes depend on an automatically generated syntactic analysis in order to help with this generalization, but automatic syntactic analysis is not perfect and sometimes makes errors. One problem with this approach is that when the input does not conform to what is expected, it can be difficult to write fall back rules which minimize the impact of the ensuing errors.

In contrast, a purely Statistical Machine Translation approach involves no rule writing, and automatically generates a very large number of rules, some of which may memorize large portions of training sentences. Present state-of-the-art models use no syntactic analysis, though this is likely to change in the near future. Translation operations can be performed on any substring of the input source language string, and are not limited to syntactic constituents. In contrast with most rule-based approaches which require clear sentence boundaries because of syntactic analysis, phrase-based SMT does not require such boundaries, matching any sub-string. Because of this the output of phrase-based SMT systems degrades gracefully when the text lacks punctuation or there are disfluencies in the text. One disadvantage of statistical machine translation is the comparatively slow speed of the search process which finds the maxima of the translation model and the language model, which considers a very large number of potential translations.

Creating New Machine Translation Systems

The Statistical Machine Translation approach can be cheaply and quickly implemented for a new language pair if the requirements of having parallel training data for the translation model, and monolingual target language data for the language model are met. Likewise, the customization of a system for a new domain is rapid. If there is access to sufficient data, the system can be simply trained on the new data. If there is limited training data available in the new domain, statistics from the new domain can be traded-off against statistics from the general domain.

1.1.3 Software Requirements

The basic software requirement was to create a program which performs automatic translation of electronic text in Modern Standard Arabic into electronic text in English with high translation quality and at reasonable speed. Input text could come from a variety of sources. It was important that the system be robust to variations in writing style, to shifts in the domain of the texts, and be robust with respect to the quality of the input text.

The software needed to be delivered on CD-ROMs or DVDs, though it could be periodically updated. It was designed to run on workstations or a laptop under recent versions of Microsoft Windows. Two configurations were envisioned, a server configuration for groups of users and a stand-alone configuration for a single user. Because Language Weaver is primarily concerned with the development of the translation engine, it was decided that APIs would be made available so that resellers and integrators could embed the Language Weaver translation engine inside their products.

Initial market research showed that the translation speed needed to be greater than 500 words per minute, and the models needed to fit within 2 gigabytes of memory. A myriad of supported document formats were envisioned. Although the initial releases supported only text files and HTML input, subsequent releases added such formats as Microsoft Word and Adobe PDF.

There were many challenges involved in delivering such a product. Language Weaver worked with the University of Southern California's Information Sciences Institute (USC/ISI), which made available algorithms and statistical MT expertise. Roughly twenty person-years of invention and development of Drs. Kevin Knight and Daniel Marcu, Language Weaver's Founders, and their students were made available. But these algorithms were implemented as scripts for performing research; not robust commercial products. The statistical paradigm involves searches over millions of possible translations for each input sentence. For example, an experimental system might translate a single document using 24 hours of pre-computation to begin with, then further processing to find the best translations for each sentence in the document. In fact, some experimental systems rely on using many computers in parallel for hundreds or even thousands of hours of computation. Such research systems also use large parameter sets associated with the statistical models which do not fit into 2 gigabytes of memory. It was a major challenge to go from such research systems to a finished commercial product.

1.1.4 Linguistic Framework

The Language Weaver system is implemented in a low-level transfer framework. Consecutive words are analyzed into units (called phrases, even in the case that the unit is a single word) and then transferred directly to their equivalents in the target language. There is no reliance on difficult syntactic analysis, which is error-prone and can be slow.

The system is able to deal with genres of text which would be difficult to describe completely and accurately with a formal grammar. It is able to deal with malformed input such as spelling errors, input lacking punctuation and other problems. Because there is no complicated analysis step involving a search over possible syntactic structures it is easier to ensure that it always generates output than it is to ensure this for rule-based systems.

By making the probabilistic assumption, a framework becomes available in which errors can be explicitly modeled in an attempt to minimize their impact. In rule-based systems, the highly trained linguists who write the rules try to envision the consequences of every possible input. However, when the rules they write turn out to be inconsistent, due to unexpected input, it is very difficult to choose between necessary errors because there is no underlying principle under which the impact of such errors can be minimized.

The Language Weaver system is able to work on specialized genres of language which may not be easily analyzable using syntactic parsing designed to work with complete sentences. For instance, the Language Weaver system is excellent at translating headlines, which are not complete sentences, even though no additional work was put into customizing the system for this purpose.

Although the overall framework is probabilistic, the current implementation of the Language Weaver system is a hybrid system. Most of the difficult translation work is done with statistical models, but in principle anything which can be described with rules can be easily integrated into the system and such rules even interact with the probabilistic translation components of the system, as will be described below. In addition, the Language Weaver engine is available with tools allowing users to specify their own dictionaries (see Section 1.2.1), and with a translation memory which can be loaded with partial or full sentences which are designated as correct translations.

1.1.5 Computational Methodology

Current state of the art SMT systems are "phrase-based", Och and Ney (2004), Marcu and Wong (2002), Koehn et al. (2003), meaning that they operate by matching groups of one or more consecutive words called "phrases" against a probabilistic translation lexicon. There is often no analysis of inflectional morphology and the modeling of syntactic word order is based only on small windows of neighboring words (these models are called n-gram models).

A phrase-based SMT system uses a bitext of parallel source language sentences and target language sentences and an alignment of that bitext. The model estimated from the bitext is called the translation model because it maps source phrases to target phrases.

The target language text of the bitext is used to build a model of good target language sentences. Additional target language text which is not from the bitext can be used to help build a better model of target language sentences. This model is used for determining which phrases fit together best, and determining the best ordering of the phrases to form a fluent target sentence. This component of the SMT system is called the language model.

See figure 1 for an intuitive overview of how statistical machine translation systems work. In the graphic, we imagine that the translation model prefers using target language phrases of "he is", "very", and using one of the two phrases "the happy" or "happy" (in practice, the number of translations considered would be many orders of magnitude greater even for this short sentence). All possible reorderings of these phrases are considered, and the translation model assigns probabilistic scores based on the goodness of each phrase translation and the goodness of the reordering (based on a statistical model of how source language phrases are reordered into target language phrases). The language model estimates how grammatical the generated target language would be. In the example, the language model prefers "he is very happy" over the other variants considered, and the search discovers that this variant maximizes the translation model and the language model and therefore chooses this as the string to output. In practice, the search process considers a very large number of possible output strings, partially constructing many different hypotheses. The search process is described in further detail in Section 1.1.5.

The statistical basis for SMT is found in the noisy channel model, which was proposed for use in machine translation by a group working at IBM research laboratories in a group of publications, including Brown et al. (1993). For ease of exposition, the source language for the translation task is referred to as "Foreign", and the target language is referred to as "English", although these can be any language pairs in practice. The variable e represents any potential string made up of English words, while f represents any potential string made up of Foreign words. $Pr(e)$ represents the true distribution over English strings, while $Pr(f)$ represents the true distribution over Foreign strings. $Pr(f|e)$ represents the true distribution over Foreign strings generated from English strings.

The translation problem is defined as follows. Given a Foreign string f, find the English string $\hat{e}$ according to Equation 1.1. The second step is an application of Bayes' Rule, while the third step is true due to

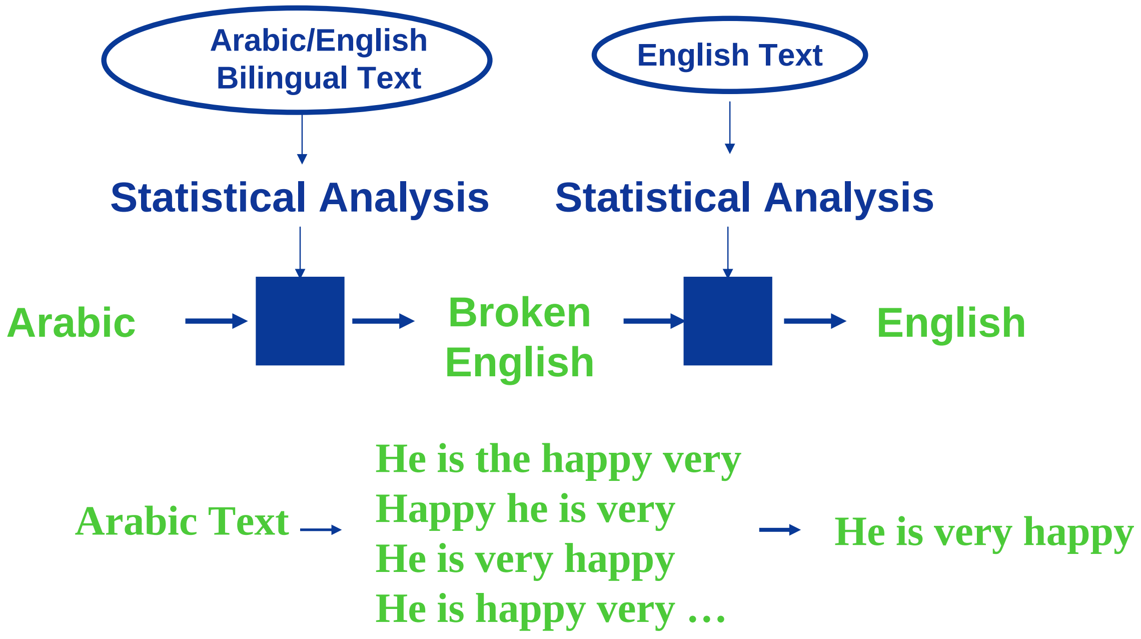


FIGURE 1 Statistical Machine Translation

the properties of the argmax operator (and the fact that f is constant, since it is the Foreign sentence we wish to translate to English).

$$\begin{aligned}
\hat{e} &= \operatorname*{argmax}_{e} Pr(e|f) \\
&= \operatorname*{argmax}_{e} Pr(e) * Pr(f|e)/Pr(f) \qquad (1.1) \\
&= \operatorname*{argmax}_{e} Pr(e) * Pr(f|e)
\end{aligned}$$

Brown et al. (1993) developed five statistical models of translation (IBM Models 1 through 5) and parameter estimation techniques for them. The models were designed to be used in a pipeline, where each model is bootstrapped from the previous model. The models are trained using variants of the Expectation Maximization algorithm. Expectation-Maximization Dempster et al. (1977), or EM, is an algorithm for finding parameter settings of a model which maximize the expected likelihood of the observed and the unobserved data (this is called the complete data likelihood; the incomplete data likelihood is the likelihood of only the observed data). Intuitively, in statistical word alignment, the E-step corresponds to calculating the probability of all word alignments according to a current model estimate, while the M-step is the creation of a new model estimate given a probability distribution over possible word alignments (which was calculated in the E-step).

These distributions are modeled using parameterized models. The model of $Pr(e)$ is referred to as the language model, while the model of $Pr(f|e)$ is referred to as the translation model. The interested reader is referred to tutorials on the IBM models, Knight (1999b). Note that in modern phrase-based Statistical Machine Translation systems there may be several probabilistic models responsible for analyzing potential target language output strings and several other probabilistic models of the correspondence between source and target strings, but conceptually these can still be divided into a language model (responsible for determining the goodness of target language strings) and a translation model (responsible for producing scored mappings of source strings to target strings). The interested reader is referred to the sections of Callison-Burch and Koehn (2005) which present phrase-based Statistical Machine translation for further details.

Training - Estimating the Parameters of the Statistical Models

The process of "Training" is the process of estimating the parameters of a statistical model. This is sometimes referred to as "learning" the parameters of the model. This process is central to the SMT approach. The basic idea is to perform automatic annotation of data, rather than manually writing rules. The process of automatic annotation used is called word alignment, and before we describe it, we discuss the preparation of the data used.

The training process begins with "parallel" training data. This consists of documents and their translations. In some cases large collections of texts and their translations must be automatically "document aligned", meaning that the mapping of each document to the document that is its translation must be automatically determined.

Once a collection of aligned documents which does not overlap with the test sets is available, preprocessing is performed. In the Language Weaver training process, preprocessing the training data consists of first performing sentence breaking (breaking the text into sentences) and tokenization (separation of punctuation attached to words). This is followed by automatic sentence alignment, where the sentences which are translations of one another are identified using statistical models.

The only data which is manually sentence aligned is data used for test sets. Recently, the MT community has begun to use automatic evaluation metrics which measure the overlap of output text with one or more sets of gold standard translations. These sets of translations are called the "references". Multiple reference test sets are often translated by translation companies specifically for the purpose of MT evaluation. Sometimes single reference test sets are extracted from existing parallel data, in which case the sentence alignment must be manually verified. It is extremely important that the test sets not overlap with the training data. If there is such overlap, the estimate of translation quality will be too optimistic.

The next step is word alignment. In this step an automatic word alignment for each pair of parallel sentences in the bitext is generated. A word alignment indicates which words are translations of one another. Word alignments are generated using statistical models as in Brown et al. (1993), Marcu and Wong (2002). An example word alignment for a parallel segment taken from the Linguistic Data Consortium's Arabic Treebank is shown in Figure 2.

A "phrase-to-phrase" alignment is then generated. The term "phrase" here refers to a group consisting of one or more consecutive words. This

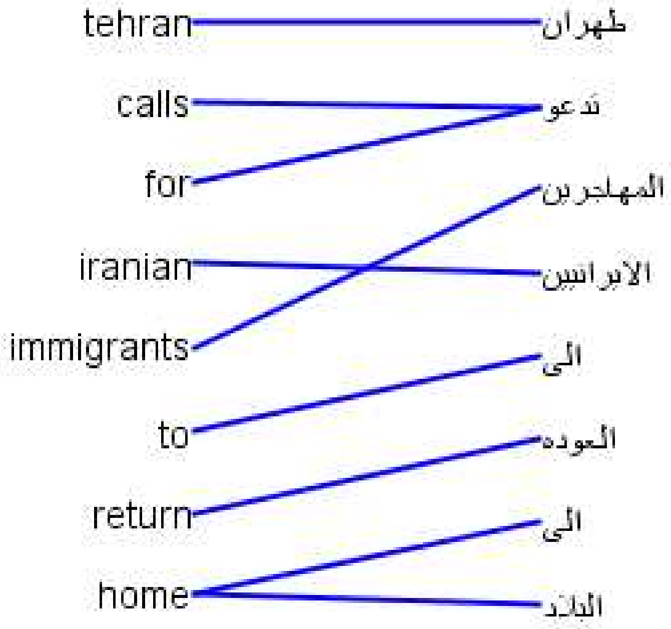
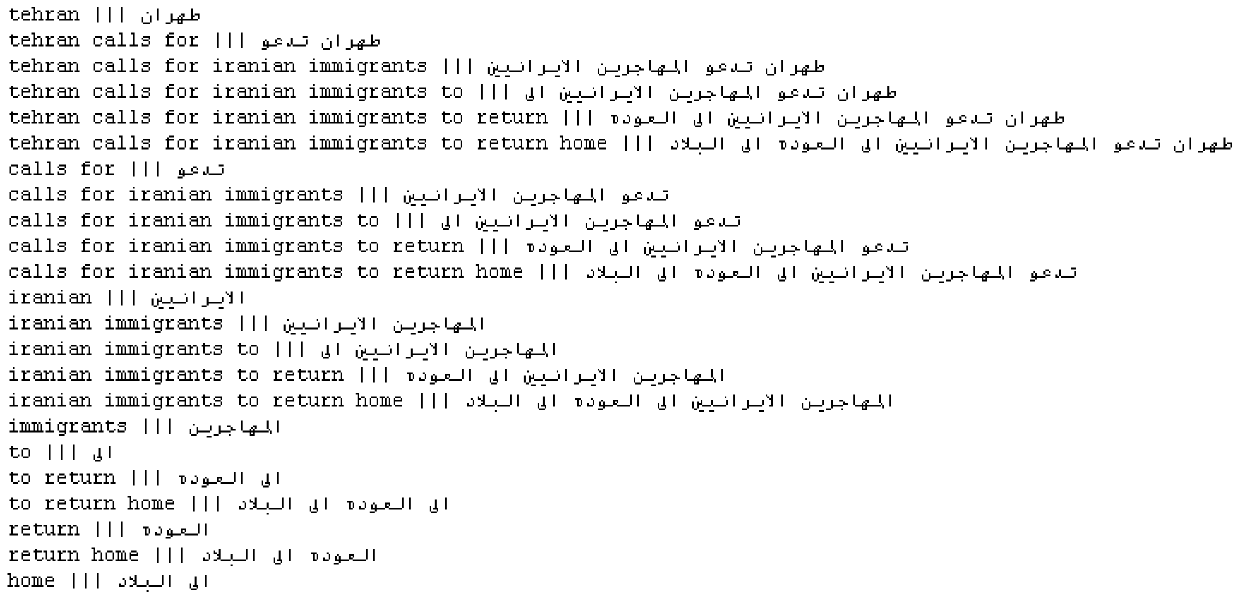


FIGURE 2 Word Alignment of a Sentence from the LDC Arabic Treebank



```
tehran ||| طهران
tehran calls for ||| طهران تدعو
tehran calls for iranian immigrants ||| طهران تدعو المهاجرين الايرانيين
tehran calls for iranian immigrants to ||| طهران تدعو المهاجرين الايرانيين الى
tehran calls for iranian immigrants to return ||| طهران تدعو المهاجرين الايرانيين الى العوده
tehran calls for iranian immigrants to return home ||| طهران تدعو المهاجرين الايرانيين الى العوده الى البلاد
calls for ||| تدعو
calls for iranian immigrants ||| تدعو المهاجرين الايرانيين
calls for iranian immigrants to ||| تدعو المهاجرين الايرانيين الى
calls for iranian immigrants to return ||| تدعو المهاجرين الايرانيين الى العوده
calls for iranian immigrants to return home ||| تدعو المهاجرين الايرانيين الى العوده الى البلاد
iranian ||| الايرانيين
iranian immigrants ||| المهاجرين الايرانيين
iranian immigrants to ||| المهاجرين الايرانيين الى
iranian immigrants to return ||| المهاجرين الايرانيين الى العوده
iranian immigrants to return home ||| المهاجرين الايرانيين الى العوده الى البلاد
immigrants ||| المهاجرين
to ||| الى
to return ||| الى العوده
to return home ||| الى العوده الى البلاد
return ||| العوده
return home ||| العوده الى البلاد
home ||| الى البلاد
```

FIGURE 3 Phrases Extracted From the Word Alignment in Figure 2

is done using the techniques outlined in Marcu and Wong (2002) and/or using simple heuristics on the word-to-word alignment, Och and Ney (2003), Koehn et al. (2003). In the latter case, possible consecutive-word phrase pairs, as indicated by the word alignment, are extracted using an algorithm similar to that outlined in Koehn et al. (2003). Figure 3 shows the possible phrase translations extracted from the word alignment shown in Figure 2. The result of the phrase-to-phrase alignment is a "phrase lexicon", which is a list of consecutive-word "phrases" with their translations. Each of these is annotated with one or more probabilistic scores. The phrase lexicon is the most important component of the translation model and is the only component used for mapping source language phrases to target language phrases.

There are several other important components of the translation model. The models responsible for word movement are one such set of components. Word movement is the problem of how to reorder phrases as they are translated. For instance see the example in Section 1.1.5. The goodness of the reordering is a factor of not only the location in the target string of the translated words, but also the context (words nearby) and the translation choices made. Additional models include smoothed estimates of the phrase translation probabilities using word-level translation probabilities.

Next, the language model is constructed. This is performed using the monolingual target language text from the parallel data and additional target language text. Generally, there is access to a much larger amount of target language text which matches the domain, and this will help the estimation of the language model. The type of language model generally used in the SMT approach is called an n-gram model. Such a model uses the Markov assumption to limit the relative context and has been shown to perform well for Statistical Machine Translation and other data-driven MT variants. The formula for a trigram (3-gram) model, where the probability of the current word depends only on the two previous words is shown in Equation 1.2, where e_1 is the first word of the n word English string e and S is a special symbol denoting the beginning of a string.

$$p(e) = p(e_1|S)p(e_2|S, e_1)\prod_{i=3}^{n} p(e_i|e_{i-2}, e_{i-1}) \tag{1.2}$$

It is well known that direct estimation of the relevant probabilities with maximum likelihood fairs poorly due to data sparsity, and in practice this estimate must be smoothed. There are many well-studied smoothing techniques. For example one effective technique is interpolation of zerogram, unigram and bigram estimates with the trigram estimate. In addition, machine translation has a vocabulary which can not be completely specified in advance and steps must be taken to make the language model robust to this (generally by replacing out of vocabulary items with one or more special symbols for which parameters are specially estimated). Syntactic language models such as Charniak et al. (2003) can also be used in statistical machine translation systems. Language Weaver has conducted extensive experimentation on the estimation of and smoothing of n-gram language models for machine translation.

In practice, several translation models and several language models are used and their estimates are combined in a log-linear fashion. The weights of the log-linear model are trained using the Maximum BLEU training algorithm defined in Och (2003). The resulting translation systems are then evaluated by translating a held out test set and evaluated.

Search

The search problem is the problem of finding a hypothesized translation which represents a maxima of both the translation model and the language model. Search is a difficult problem since it considers hypotheses which exist in an exponential space. Even for simple models performing a complete search has been shown to be NP-complete, Knight (1999a), so sophisticated approximate searches are performed instead. The algorithms used only expand a polynomial number of hypotheses but still generally find a very good candidate. Two different types of search have been reported in the literature for phrase-based SMT. These include hillclimbing, Marcu and Wong (2002), which is a form of local search where a limited number of operations are applied to complete hypotheses in order to find better hypotheses, and beam search with forward cost estimates such as is described in Koehn et al. (2003).

The search process, which looks for a hypothesis which has maximal translation model and language model scores, is essentially like looking through a huge number of rules to see which ones fit together well. This is a form of integrated analysis and generation, where the decision made for the first word can have a large effect on the decision made on the next word, or even several words away. The number of "states" involved, i.e. ways in which a translation decision can affect a subsequent decision, is very large. Approaches to rule-based MT also model the effects of initial translation decisions on subsequent ones. However, each decision places the system into a state which was explicitly programmed by a trained linguist, so the number of such states is necessarily limited. Of course, in the rule-based case, the states themselves are far more informative and sophisticated rules can be written which condition on the state of the system at that point in the translation process.

Evaluation

Evaluation is a very important aspect of MT development under all paradigms, whether data-driven (like SMT) or rule-based. It is vital to "hold out" testing data because otherwise developers will tend to optimistically overestimate the applicability of the new models or rules they are developing.

Ideally, developers will never view the true test data. In general, a hierarchy of test sets is used. One test set is used for measuring day-to-day performance, which developers are allowed to view. This is called the development set. There is then a test set, which is used periodically to verify that a new feature is improving performance. Finally, the blind test set is a set which is strictly controlled and used to verify that the entire system has improved since the previous blind test.

Human evaluation of accuracy is the gold standard for measuring machine translation accuracy. Such evaluation can be based on assigning absolute values of goodness to sentences in isolation, ranking possible outputs comparatively, or on post-editing. However, conducting such evaluations is time consuming and it can be difficult to ensure that the evaluation is sufficiently objective. In practice, human evaluations are expensive, and would be impractical for measuring day to day improvements.

Language Weaver uses automatic evaluation heavily in the development process. Automatic evaluations are useful for automatically testing MT performance without the overhead of manually judging sentences. An array of automatic scores such as the BLEU score, Papineni et al. (2001), are used for internal development. These automatic scores have been shown to reflect translation quality judgments by human annotators.

In practice, careful attention is paid to comparing a baseline system with the same system in which only one feature is changed, making it unlikely that important errors will be made using automatic evaluation metrics to guide decisions. Typically, when changes are made, an automatic evaluation of the original system and the changed system is done and either the score difference is adequately explained or the change is backed out of. Incidentally, this shows that automatic evaluation can also be useful as a form of regression test, which will be discussed in a subsequent section. Automatic evaluation metrics can also be used to measure the effects of changes in data processing, though care must be taken if these changes affect the test sets (such as, for example, changes of target language tokenization), as scores will generally not be directly comparable.

The success of SMT systems depends on constant experimentation. For this reason, a significant amount of attention is paid to optimizing experiments to run quickly, and a significant amount of effort is put into the ability to test many conditions in parallel.

Hybridization: Integration of Rule-Based Components

A strong advantage of using the statistical approach is that it easily allows for hybridization. Since the underlying approach is statistical, hybridization implies the integration of rule-based components. Rule-based components can be viewed as another source of knowledge about either the suitability of the mapping of the source string to the target string, or the goodness of the target string. In practice we experimented with two different ways to hybridize the translation process. The simpler method was to view the rule-based components as a preprocessing step which alter the input to the statistical process. A more complex approach allowed rule-based components to replace parameters of the statistical model on the fly as input source language text was processed. For instance, one rule-based system specifies possible translations of a string it has recognized as a date. It may even specify probabilistic weights which should be associated with each variant. During the search process the language model may prefer one or another of the date variants based on decisions made for other strings (in the nearby context) which are translated using the statistical models.

Comparison with Rule-Based Translation Process

We have conducted limited comparisons of our methods with rule-based systems, and found that we have comparable or better translation quality in the domains which our systems are targeted for. We made several observations about the strengths of the SMT approach. Empirically, we found that word sense disambiguation is not as big a problem for our systems as for rule-based systems, because our systems memorize large amounts of context in the form of consecutive-word phrases. Using purely syntactic models of context does not seem to be sufficient (and Koehn et al. (2003) provides some evidence to this effect), though it would be logical to build SMT systems which were able to fall back to this type of context in the event that other contexts were not present in the training data. Another observation we made is that jargon and mal-formed input frequently cause problems for rule-based translation systems which require strong syntactic analysis, but have much less of an impact on our system.

1.1.6 Building an MT system for a new language pair and/or domain

Building an MT system for a new language pair and/or domain is straight-forward. First appropriate test sets are defined. Then parallel training data for the appropriate domain or domains is identified. Next, monolingual training data for the language model is specified.

An initial system is built and evaluated on development data. The system is then iteratively refined. This refinement may include both work on new statistical components or estimation techniques, or it may focus on increasing the coverage of the rule-based components. Sometimes, additional data may be mined from comparable corpora using techniques similar to those outlined in Munteanu et al. (2004).

One important aspect of building a new MT system is handling the parallel data. Language Weaver has a commercial product called the

“TM Generator”. The TM Generator is a suite of tools, built around a database application, for storing and reusing previously translated text, which is used internally at Language Weaver and can also be used by end users, with or without the SMT translation engine.

The TM Generator suite consists of three components, which are called Build, Search and Match. The Build component extracts text from common file formats (HTML, PDF, DOC, plaintext, XML) and uses statistical sentence alignment algorithms to automatically align each text segment in one language with its foreign language counterpart. Users may review and adjust alignments in a graphical environment before saving the bilingual database as a TMX file (TMX is a commonly used translation memory format). The Search component provides a flexible search interface to allow users to browse TMX files. The Match component utilizes previously translated material to semi-automatically translate new text. It has a flexible database search algorithm for translating both exact matches and for helping users make minimal modifications to previous translations to obtain a match. Once documents in a parallel corpora are document-aligned, Language Weaver engineers use the Build component of the TM Generator suite to process the parallel corpora necessary to build a system.

Training Data

The most critical component of building a new translation system is the choice of training data for the system. The parallel data used to build a Language Weaver translation system can be in any number of formats including:

1. translation-memory data (often in TMX format)
2. glossaries
3. translated archives
4. data extracted from comparable corpora

Target language texts necessary to build the language model are also very important. It is often possible to acquire monolingual training data which is many orders of magnitude greater than the bilingual training data, and experiments have shown that this improves translation performance.

In practice, there are many possible sources of data. Customers may have data on their intranets, hard-drives/tapes and/or websites. Language Weaver may also have access to relevant data which was created by Language Weaver, licensed from third parties, or in the public domain.

Language Weaver Learner

In the first step of the training process, parallel documents are identified. Once parallel documents have been identified, sentences of the source and target language text are aligned to create a parallel corpus. The Language Weaver Learner processes this corpus. It extracts probabilistic translation dictionaries and patterns which are called translation parameters, as mentioned in section 1.1.5.

Language Weaver Decoder

The translation parameters are used by a statistical translator, called the Decoder, to translate new texts for users. The software uses these stored associations of source phrases to target phrases to produce translations of new documents. The language model will be used to find grammatically correct output in the translations proposed. The search process which, given source text, produces target text by maximizing the translation model and the target model is what is implemented in the Decoder. The Decoder is the component delivered to customers.

Delivery of Software to Customers

The translation software can be delivered to customers in several forms. The most important component, which is always present, is the core translation system, which includes the Decoder and the translation parameters which it depends on. There are several user interfaces available, which are directly useful to end users. In addition, there are a large number of APIs available which enable the Decoder to be integrated into client products. The TM generator is also available to allow users to take advantage of their post-editing of machine translation output. These software packages are distributed on DVDs and fully self-contained.

Periodic updates of the translation parameters are available to customers. Each update of the parameters is certified to work with a particular version of the software. Language Weaver also issues new versions of the software on a schedule that is less frequent than the translation parameter updates. Both of these kinds of updates are also distributed on DVDs and are fully self-contained.

Overview

See figure 4 for an overview of building a new MT system.

1.1.7 The Development Plan

There were three major phases in the development plan for the Language Weaver Arabic to English Statistical Machine Translation System Software, see below. The goal was to build a commercial system

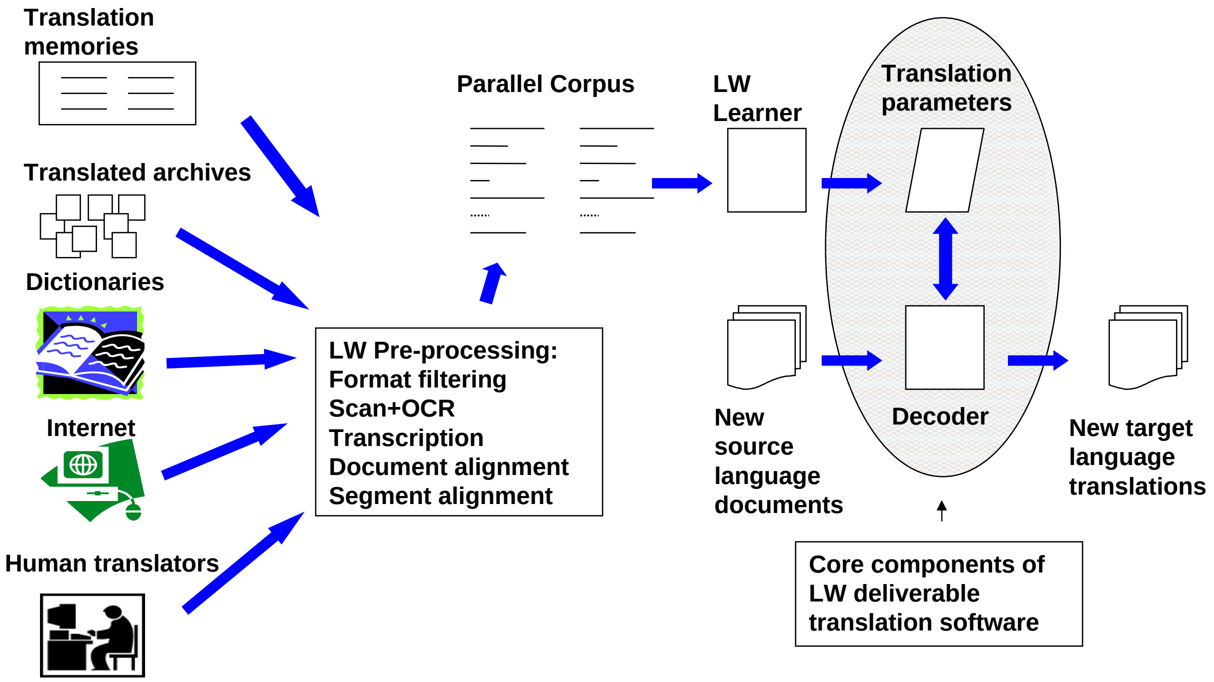


FIGURE 4 Building Statistical Machine Translation Software Systems

performing at quality levels consistent with the state of the art results obtained by research systems, but also had a reasonable translation speed and was a robust, supported product which could be installed at a customer site by customer staff.

1.1.8 View of Final Product

We were happy with the translation quality of the final product. It is self-contained, installing off of a DVD onto a desktop or laptop. It is capable of robustly translating at speeds several orders of magnitude higher than research systems, with virtually no loss in translation quality. The engine itself can translate 5,000 to 14,000 words per minute, per CPU, which is an impressive improvement over research systems. The engine actually scales up to 500,000 words per minute on a high throughput network using parallel computation on multiple computers.

See figure 5 for an example screen-shot of the standalone product translating a document.

1.2 Implementation

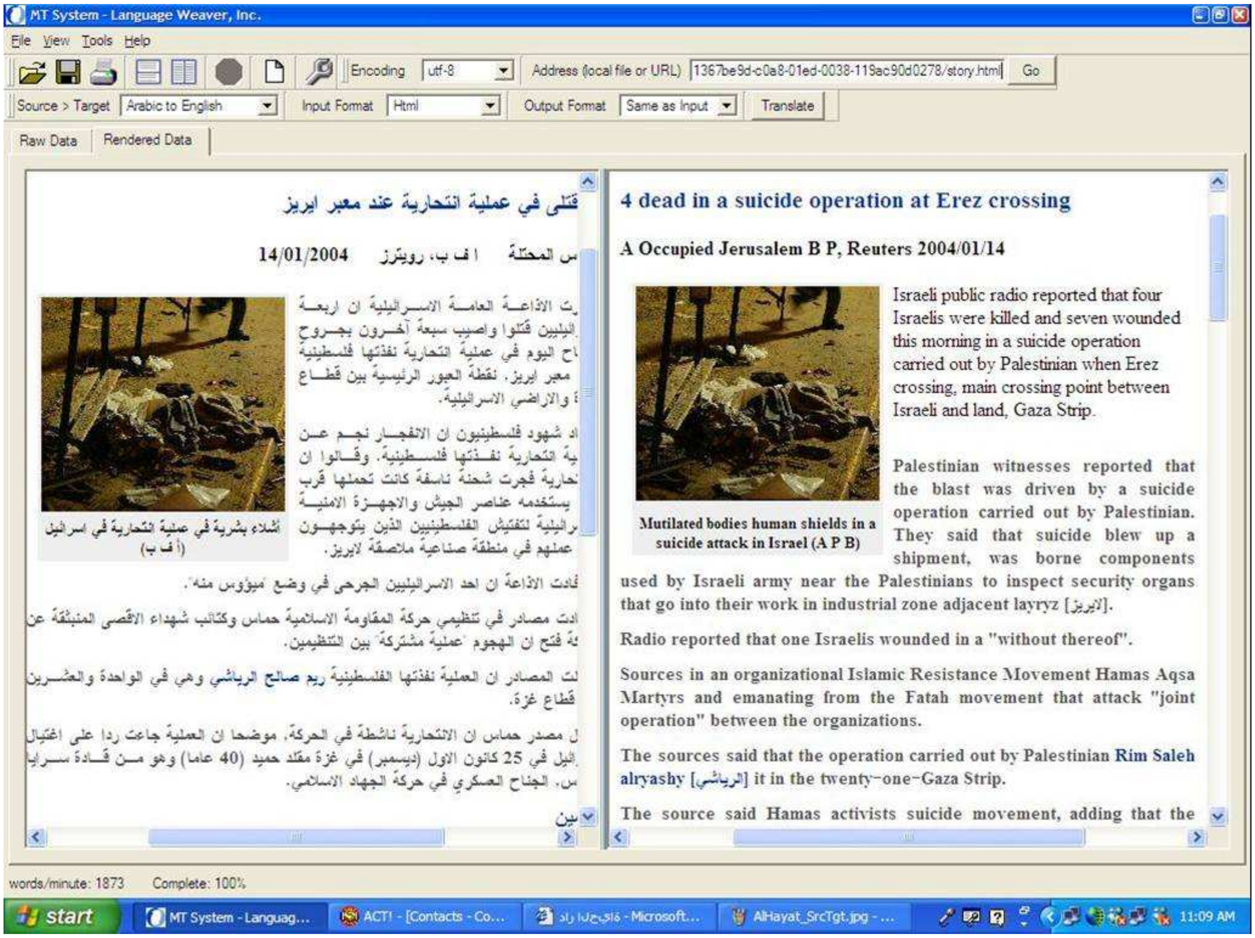
Planning for the Language Weaver Arabic to English statistical machine translation software system lasted 2 months and involved researchers, software engineers and quality assurance personnel as well as management and marketing.

The first step in planning was the creation of a design document. This document specified the desired capabilities of the product without specifying the technical implementation. For the first release this was written by the researchers and management, but after this release the marketing department presented customer feedback and this was strongly taken into account in writing subsequent design documents.

A configuration management plan describing how modifications were to be implemented, controlled and tracked throughout the project was then created, which was another key component of the planning process. There were several other important documents created during the implementation phases. Of particular importance were an evaluation plan, detailing the tests of accuracy which are described below, and a quality assurance plan.

For each release, a detailed technical specification was then written primarily by software engineers, showing exactly how each feature mentioned in the design document would be implemented, and also showing how different components of the system would intercommunicate.

Finally, plans for the creation of the installation software were created, as were plans and systems for supporting customers and tracking customer requests for new features.



قتلى في عملية انتحارية عند معبر ايريز

س المحتلة   ا ف ب، رويترز   14/01/2004

ت الاذاعة العامة الاسرائيلية ان اربعة
اليليين قتلوا واصيب سبعة آخرون بجروح
اح اليوم في عملية انتحارية نفذتها فلسطينية
معبر ايريز، نقطة العبور الرئيسية بين قطاع
ة والاراضي الاسرائيلية.

اد شهود فلسطينيون ان الانفجار نجم عن
ية انتحارية نفذتها فلسطينية. وقالوا ان
نحارية فجرت شحنة ناسفة كانت تحملها قرب
يستخدمه عناصر الجيش والاجهزة الامنية
رائيلية لتفتيش الفلسطينيين الذين يتوجهون
عملهم في منطقة صناعية ملاصقة لايريز.

أشلاء بشرية في عملية انتحارية في اسرائيل
(أ ف ب)

فادت الاذاعة ان احد الاسرائيليين الجرحى في وضع "ميؤوس منه".

ادت مصادر في تنظيمي حركة المقاومة الاسلامية حماس وكتائب شهداء الاقصى المنبثقة عن
ة فتح ان الهجوم "عملية مشتركة" بين التنظيمين.

لت المصادر ان العملية نفذتها الفلسطينية ريم صالح الرياشي وهي في الواحدة والعشرين
قطاع غزة.

ل مصدر حماس ان الانتحارية ناشطة في الحركة، موضحا ان العملية جاءت ردا على اغتيال
ائيل في 25 كانون الاول (ديسمبر) في غزة مقلد حميد (40 عاما) وهو من قادة سرايا
س، الجناح العسكري في حركة الجهاد الاسلامي.

بين

4 dead in a suicide operation at Erez crossing

A Occupied Jerusalem B P, Reuters 2004/01/14

Israeli public radio reported that four Israelis were killed and seven wounded this morning in a suicide operation carried out by Palestinian when Erez crossing, main crossing point between Israeli and land, Gaza Strip.

Mutilated bodies human shields in a suicide attack in Israel (A P B)

Palestinian witnesses reported that the blast was driven by a suicide operation carried out by Palestinian. They said that suicide blew up a shipment, was borne components used by Israeli army near the Palestinians to inspect security organs that go into their work in industrial zone adjacent layryz [لايريز].

Radio reported that one Israelis wounded in a "without thereof".

Sources in an organizational Islamic Resistance Movement Hamas Aqsa Martyrs and emanating from the Fatah movement that attack "joint operation" between the organizations.

The sources said that the operation carried out by Palestinian Rim Saleh alryashy [الرياشي] it in the twenty-one-Gaza Strip.

The source said Hamas activists suicide movement, adding that the

FIGURE 5 Language Weaver Statistical Machine Translation Software

The full project, involving three releases, was originally specified for 18 months but it was eventually successfully delivered in just under 14 months.

1.2.1 Stages of Implementation

First Release

The priority in the first release was having good translation accuracy. The main features were designed so as to have a fully functional product which showed the possibilities of Statistical Machine Translation but had only a simple user interface without many bells and whistles. The first release supported only the UTF-8 character set. It supported HTML filtering but only stripped the markup, rather than re-inserting translations into the original HTML markup. Other important components of the first release included robust sentence breaking and a robust tokenizer. The first release was available in 1 and 2 gigabyte versions. Both of these versions were available in a standalone version which could browse and translate individual web pages, or in a server version compatible with the Apache web server. Both versions translated at 500 words per minute. The API was client-server using CORBA and APIs were made available to customers for both the C and the C++ programming languages. The engine supported settings which allowed for both a high quality mode aimed at obtaining the highest translation accuracy possible, and a draft mode, aimed at higher speed with a small loss in accuracy. The software was delivered for the current versions of Microsoft Windows, and included both server and desktop versions.

Second Release

The second release had improved translation accuracy over the first release. It included support for the UTF-8, UCS-2 (UTF-16), CP-1256 and ISO-8859 character sets, which were all of the Arabic character sets used in the data which we were working with. It supported "HTML refiltering", which meant that translated web pages would be displayed with the same formatting (and pictures) as the original pages. The engine tracked how each word was translated and used a sophisticated algorithm to restore markup within sentences so that the translation of a bolded word in the source text would be shown as a bolded word (or words) in the translated target text. An important issue handled in this release was character normalization, which we will describe below. This version added a rule-based component for transliteration of unknown words based on a simple greedy search which matches individual characters or groups of characters, and a rule-based component based on finite state approaches for translation of easily characterizable phenomena such as dates and times which was fully integrated into the statistical search process. This rule-based component allowed the search process to pick the best way to translate these phenomena given the context. This release also added a morphological analyzer, also based on finite state techniques, which was intended to increase the covered vocabulary by mapping unknown words to known words within a phrasal context using simple transformations based on stripping or adding common prefixes and suffixes until a match with known vocabulary is achieved. Language Weaver developed a proprietary pipeline allowing the various components to communicate with each other using XML. On the software side, the engine was at this point fully multi-threaded, and the APIs offered were expanded to include a simple and powerful CGI API. Document types supported were expanded to include Microsoft Word and Adobe PDF.

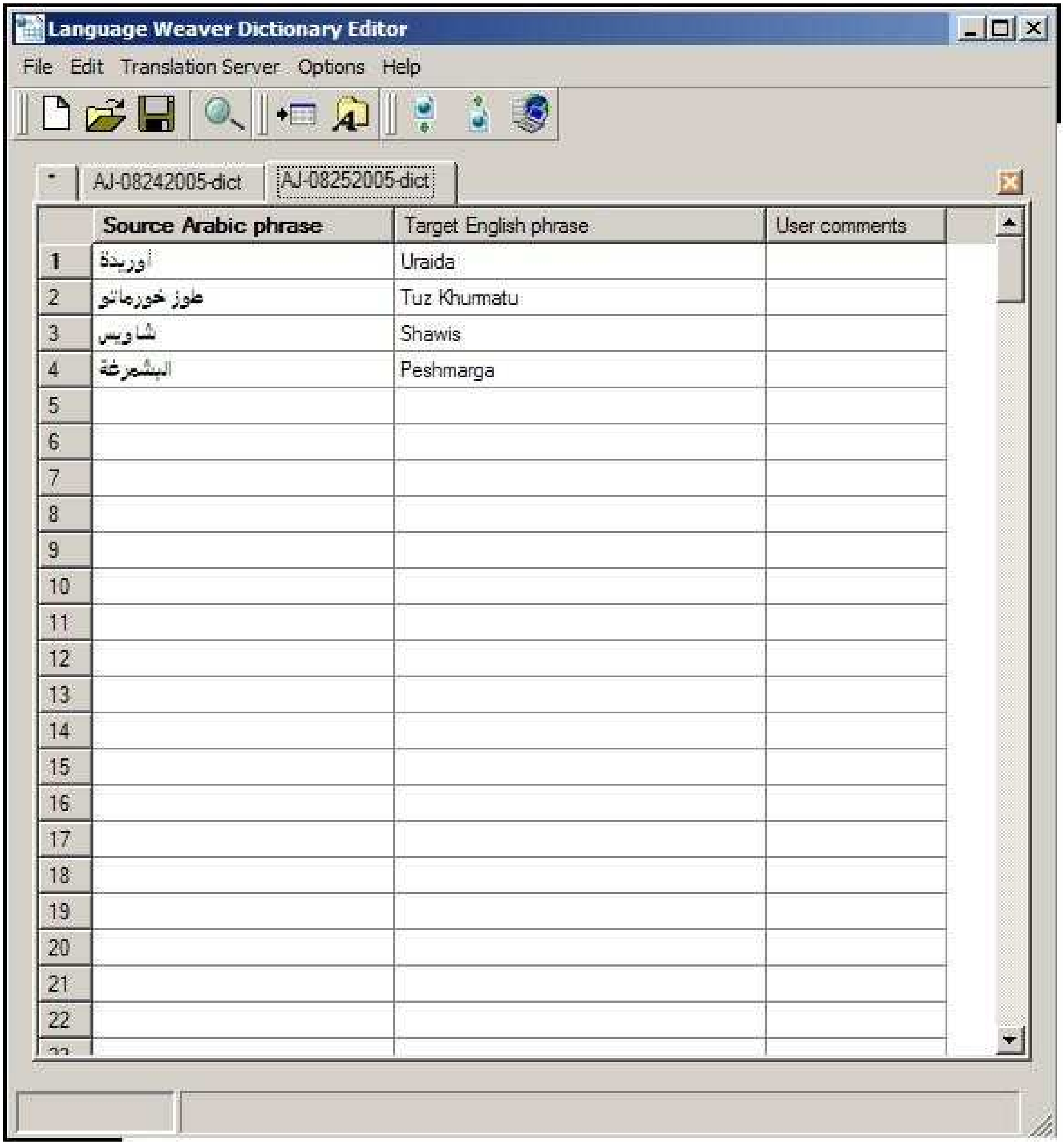
Third Release

The third release obtained excellent translation accuracy, which we believe established a new level of quality in Arabic machine translation. An important new feature was user customization of translation lexicons. New APIs included a Webservices API based on SOAP and a Java API. It featured a simple server interface which was similar in appearance to a printer queue and allowed different users to queue up large numbers of translation jobs to be serviced. It also provided seamless click-through so that users could browse a translated version of the web with formatting intact. Finally, for large customers, it supported load balancing across multiple CPUs.

Another important part of the third release was the adding of user defined lexicons. The Language Weaver dictionary tool was developed, which allows users to enter phrase translations (one or more consecutive words). Figure 6 shows phrase entries which were added by a translator translating a news article. An associated API was also made available, allowing third parties to write code which performs operations on user dictionaries such as creating new dictionaries, translating documents with a particular dictionary, and managing individual entries in a particular dictionary, as well as importing and exporting dictionaries in various translation memory formats.

1.2.2 Difficulties in Implementation

There were several difficulties in implementation to which it was necessary to react. Each of these occasioned changes in the technical specification, and some of them required changes to the design document



	Source Arabic phrase	Target English phrase	User comments
1	أوريدة	Uraida	
2	طوز خورماتو	Tuz Khurmatu	
3	شاويس	Shawis	
4	البشمرغة	Peshmarga	
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			

FIGURE 6 Language Weaver Dictionary Tool

which were negotiated with management.

The first difficulty we encountered was in dealing with Arabic character encodings. There are three main encodings used, UTF-8, CP-1256 and ISO-8859. At first we had planned to allow the user to specify an encoding if the input were a document, while if the input were from the World Wide Web (WWW), we expected information on the encoding would be present in the header returned from the WWW server. However we found, much to our surprise, that in the data we were working with two or more of the encodings were sometimes mixed in ways which were surprising and there were also frequently LATIN-1 characters interspersed. Sometimes this even occurred within the context of a single word, number or acronym. An example of this is that occasionally we found that the Arabic letter Reh was misused as a comma inside of numbers. Another problem was that WWW servers frequently returned headers indicating the incorrect encoding for a web page.

The second set of problems was character normalization issues. The first author had been previously involved in experimentation in the area of Information Retrieval for Arabic. This is documented in Xu et al. (2002), Fraser et al. (2002), and this research shows how many of these issues can be dealt with in an Information Retrieval context. Some specific problems encountered were that Kashida, short vowels and Shadda were irregularly present in much of the data we were working on. One large issue was handling the different Arabic writing styles. This involved handling the different Alef variants, i.e. those Alef variants with and without Hamza and/or Madda. The Alef Maqsura/Yeh alternation and the Heh/Teh Marbuta alternations which seem to occur most frequently in Egyptian text were also sometimes present in the data. The other Hamza variants (not involving Alef) did not seem to be as large a problem, as they were usually consistently present. See Section 1.2.3 for details of our solution to this problem. Another issue we encountered frequently was the use of Persian/Farsi characters in transliterated names[2]).

Another set of issues revolved around speed. As we have previously mentioned, the search for a maxima of the statistical models is expensive. The first Language Weaver reimplementations of research systems were unacceptably slow, and it turned into a major challenge to produce a system with acceptable speed and translation quality.

Memory issues revolved around the need for statistical models to be held in memory. The probabilistic lexicons are very large as they

[2] For instance, the Arabic transliteration of Paris would sound like “Baris”, but there is a Farsi letter pronounced in a similar way to the English letter “p” which is used by some writers of Arabic.

contain all possible translations within the training data, see Figure 3 for an example of the types of translations stored. The primary concern here was how to prune the statistical models so that they would fit in the desired memory footprints of 1 and 2 GB. Again, extensive experimentation was performed to determine the best trade-offs such that translation accuracy was very close to research system levels while the parameters were able to fit into the designated memory space.

There were also issues involving vocabulary coverage. These problems could be broken down into two different types. The first type was the infinite variation of more easily describable phenomena such as dates and numbers. However, we also had problems with morphological variants. Sometimes, even though we had observed several morphological variants of a word, we would encounter a different variant in material we were translating, indicating that handling morphological variants was important for extending the coverage of the parallel training data.

Finally, we needed to deal with mixing of Arabic and Hindi numbers and a surprisingly large number of different comma characters in numbers (including, most surprisingly, the Arabic character Reh, which appeared on some web pages).

1.2.3 Changes made

In response to these problems, we made a number of changes. For character encodings, we developed a statistical encoding detector, and we strengthened our tokenizers' capabilities to make them better able to deal with tokens which contained more than one encoding.

We then implemented character normalization. There is some cost in ambiguity, but we empirically verified that this is not a large problem in practice. This does not degrade the translation results except in very isolated cases which practically never occur.

For the translation system, in early implementations of the system we found that it was important to strip Kashida, short vowels and Shadda, since these were generally not present in much of the data we were working with[3]. We found that when vowels or Shadda were present the additional information added tended to be minimal. For instance in our evaluation the system did not have abnormal difficulty with the passive/active verbal distinction which is sometimes marked

[3] For the reader unfamiliar with Arabic writing, Kashida is a calligraphic extension character which does not carry any semantic information, Shadda is a consonant doubling marker which is often not written, and the short vowels in Arabic are rarely written except in some cases where they serve to disambiguate otherwise confusable words.

in text by placing a short vowel which would not otherwise be written. So we adopted the consistent approach of simply stripping these markers. There are plans to revisit this decision in future versions of the product; it is possible that a more sophisticated morphological analyzer will be included and in this case a probabilistic model could be envisioned which would be able to make a better estimate of the translation probabilities in the event that vowels or Shadda were present.

It was particularly important to handle the different Alef variants (with and without Hamza). The same word appearing with a simple Alef or with an Alef and a Hamza marker frequently occurred. We adopted a fall back normalization approach where different normalizations are applied until a match with known vocabulary occurs. The Alef Maqsura/Yeh alternation and the Heh/Teh Marbuta alternation which occurred frequently in Egyptian text were also handled in a similar fashion. The other Hamza variants (not involving Alef) were more generally consistent and did not seem to be as large a problem for the translation system though some engineering effort was put into handling these as well. Alef with Wasla was also occasionally present, and appropriate steps were taken to ensure that it was correctly handled. Finally, Persian/Farsi character variants were handled using the same fall back methodology.

A significant amount of effort went into improving the speed of the product. This was done both by programming code optimization and by working on the statistical modeling. Programming code was extensively monitored and examined for speed ups, with the engineering team reviewing one another's code to offer suggestions for optimizing algorithms. With respect to statistical modeling extensive effort was placed into trading off speed versus accuracy where the primary goal is to minimize loss of accuracy due to higher speed. This was primarily accomplished through extensive experimentation in using pruning on both the parameter files and within the search. In addition, the search algorithm was extensively engineered at a cost of many man months. The resulting proprietary algorithms are very fast with virtually no loss in accuracy.

The vocabulary issues we were able to easily solve were problems with coverage of date/time/number expressions, in particular the two date systems used in the Arab world and complex date/time expressions. This was relatively straight-forward to solve, by mining the web for relevant expressions and then expressing them as grammars so that they could be processed using finite state techniques. We developed components for translation of regular phenomena like dates, times and numbers. One example of this type of component was a rule-based component for translating dates, including the frequent mentioning of months by two alternative words which occurs in many new articles, and handling of Hijri dates. Alternative translations are passed along with probabilistic weights to the translation engine, which then chooses the best alternative given the context during the search process. This is implemented in a transfer framework which uses regular expressions for analysis and communicates possible outputs to the Language Weaver Decoder to be further processed during the search.

A rule-based component for morphology was implemented, primarily aimed at extending vocabulary coverage. This was implemented as a rule-based approach, which converts unknown vocabulary into known vocabulary by using an extensive set of rules written by Language Weaver specialists. The analyzer communicates many possible variants to the Language Weaver Decoder and attaches both probabilities and constraints on the context in which each variant should occur. If the constraints are not satisfied then the word is output in a transliteration which is calculated using a statistical model. In future revisions of the product Language Weaver plans to add more powerful morphological processing of Arabic.

1.2.4 Roles of personnel

We had three main roles of personnel in the development process. The first role was research. Researchers were responsible for translation accuracy, including vocabulary coverage. Researchers made algorithmic decisions about the statistical modeling. Research was also jointly responsible with Engineering for finding the best trade-off between speed and translation accuracy, and carried out the experiments necessary for this.

Engineering and Development was the second role. Data processing was a vital part of this. Correctness and speed of the implementations of the research algorithms was a primary consideration. Much attention was also paid to optimizing memory usage. Engineering interacted with Quality Assurance personnel with respect to software robustness, and in particular ensured that sufficient regression tests were implemented to ensure that bugs were found before software was shipped. The engineering team implemented the APIs and the stand-alone interfaces, and was ultimately responsible for the software.

Quality Assurance (QA) was the third role. QA was very important for this product, since the experimental algorithms had never before been implemented in a robust fashion with consistent memory management, and good software engineering practices. Quality Assurance helped ensure that adequate testing of the product was performed, and played a role in ensuring that speed and memory targets were met. Installation procedures and license control was another area which proved to be important and this was handled in conjunction with Engineering.

Naturally Management and Marketing also carried out their customary roles. QA and Marketing both dealt with customers in their different capacities, and after the first release this became very important in the development process as they reported customer feedback to the research and engineering groups.

1.2.5 Lessons learned

The strongest lesson learned was that the SMT approach is effective for comparatively high accuracy translation.

Closed class phenomena, such as dates and times, can be handled well with rules and possible variants proposed using these rules can be chosen within the statistical framework conditioned on context.

Morphological processing was useful. Here it was limited primarily to the role of extending vocabulary coverage. We anticipate morphology will have a bigger future role. Work on extending the strength of the lexicon is already underway.

It is important to emphasize that the Language Weaver system is now a hybrid system. The main component is a phrase-based SMT engine. However, it has additional rule-based components. Character normalization, morphology and date/time/number translation are all handled using rules, and it is anticipated that additional rule-based power will be added to the system. We believe that hybridization is the best way forward for all MT developers to achieve maximum performance regardless of the initial approach they chose.

1.3 Results and Evaluation

1.3.1 Evaluation criteria

The most important evaluation criterion at Language Weaver is translation accuracy. This is evaluated using both automatic metrics and human evaluation. However, speed and memory are also important considerations because the product is sold as a self-contained piece of software. In addition, there are constant regression tests for robustness and stability of the software.

Rule-based components can be tested both for their impact on accuracy and using regression tests (for rules where the output is deterministic and will not be further processed by the Decoder). There are also tests for vocabulary coverage which enable the easy identification of where the system needs to be strengthened.

Customer feedback is critically important to Language Weaver. In particular, additional features are strongly determined by customer requests.

1.3.2 Tests implemented

As we indicated previously, the most important tests implemented are those which involve translation accuracy. The human evaluation is conducted using proprietary post-editing techniques which we hope to report on at a future time. For automatic metrics, automatic metrics such as the IBM BLEU metric, Papineni et al. (2001), are used. Automatic metrics of this kind are a fast and objective end-to-end evaluation. They are used in-house extensively, and are also used as part of regression tests for some components. Automatic metrics are critical for speeding incremental development. They can also be useful for quick comparisons with other MT systems but it is important to control for effects of the domain of the test if there is not reason to believe that both systems were developed for that domain.

Unfortunately the BLEU metric is not useful for showing absolute translation quality. BLEU scores are not comparable across test corpora. A BLEU score of 0.30 on one test corpus from one system and a BLEU score of 0.40 on another test corpus by a different system are not comparable. An automatic metric (and associated test sets) for showing absolute translation quality is an unsolved problem which is of interest to all MT developers. Even manual evaluation of absolute quality is difficult to do in a systematic fashion.

Speed, memory usage and robustness are tested automatically. These tests involve both standardized benchmark sets for measuring speed and memory, and simulations of user activity over long periods of time which are aimed at detecting problems like memory leaks and software instability.

Regression tests are implemented to test the robustness of each component of the software. The Quality Assurance team reports to each developer what percentage of her/his code is covered by these tests, and developers were strongly encouraged to provide more code coverage by writing more tests.

Rule-based components are evaluated using a combination of automatic metrics and regression tests. Special test sets for rule-based components were developed which consisted of developer-visible development data and held out data specifically aimed at measuring the effectiveness of a particular rule-based component. These were mined from the web and from training data. In addition, standard accuracy tests are used to see the overall impact on translation quality. Regression tests are used to ensure that rule-based components are stable and sufficiently fast, and are also used to test the software in cases where there is no interaction with the underlying machine translation engine (such as is the case for translation memory, or for those rules which only provide a single translation alternative to the translation engine).

Vocabulary coverage can also be easily automatically tested. When out of vocabulary terms appear efforts are made to mine relevant parallel data containing that term, or if this is not possible the term can be added directly to the translation lexicon. In either case the language model is updated with relevant target language text.

1.3.3 Results

The results of the testing regimen implemented have been entirely satisfactory. Language Weaver will continue to improve translation accuracy as quickly as possible. Speed, memory usage and robustness of components are all tested in a special QA cycle after the code and translation parameters are frozen prior to each release. Customer comments and any possible problems are carefully tracked and reported to the relevant team.

1.3.4 General Lessons

The general lessons learned are that there is a need to constantly measure the things we are working on. Accuracy is particularly important, but it is also very important to automate tests of speed, memory, and robustness. For accuracy, it is important to ensure that data is properly "held out", meaning that developers do not inadvertently have access to test sets and that there is no accidental contamination of training data. Both programming code and translation parameters are carefully tracked to ensure that experimental conditions are repeatable and the changed condition is carefully documented. Language Weaver has also developed an in-house data handling system which carefully tracks the statistics of all data collected. This has been an important factor in managing the large quantities of data Language Weaver engineers work with. Finally, the other lesson learned was with respect to prioritization of modeling efforts. It is important to carefully evaluate what can be easily covered with rules without worrying about very difficult to describe phenomena which will be covered by either better statistical models or new rules in future versions of the product.

1.4 Modifications and Implementation of Changes

1.4.1 Modified plan

The largest modification of the original plan was that we decided to integrate the new more powerful rule-based systems. This required extensive design work, which was primarily the role of the research group, but also heavily involved the engineering group in the implementation stage. Morphological analysis was also implemented. Finally a substantial effort was put into figuring out how to ensure the quality of these new components, both with respect to translation accuracy and with respect to software considerations such as speed, memory and robustness.

1.4.2 Modification of the specifications

The specifications were changed to reflect the added rule-based and statistical models. One important area of modification involved statistical models for encoding detection and the new tokenization capabilities for dealing with multiple character encodings. The rules for normalization to match known vocabulary were also extensively researched, resulting in further changes to the specification. Rule-based components for date, time and number translation were added and the rule-based component for morphology was also implemented.

1.4.3 Implementation of Modifications

The modifications were implemented within the original time frame, and in the end the project was still delivered early. All in all, the engineering group only required two months to implement the changes beyond the time already allocated, which was a very fast turn-around. One reason that it was possible to do this was that we were enhancing the system by adding the sub-components rather than creating a new pipeline from scratch. In particular, we were extending the statistical framework with rules to make a hybrid system. We had designed the component inter-communication protocol and the necessary communication "hooks" from the beginning, and it was easy to ensure that the rule-based systems had the desired impact on the final translation within the probabilistic framework. In light of our experience implementing these changes, we believe adopting a statistical paradigm from the beginning and enhancing it with rule-based components is an excellent commercial approach.

1.4.4 Cycles of Development

The cycles of development at Language Weaver are:

1. set overall goals
2. design and research
3. implementation
4. testing
5. modification

TABLE 1 Software Speed in Words Per Minute (Quality Level versus Memory Size)

Quality	512 MB	1 GB	2 GB
High	6,000	4,500	2,000
Medium	9,000	7,000	5,000
Low	12,500	9,000	7,000

6. testing (of modifications)
7. code freeze
8. quality assurance
9. delivery

We found it vital to have a clearly separated research cycles, development cycles and QA cycles. We followed each of these stages for each release of the product. The deadlines were carefully planned, with dependencies explicitly specified. We found in practice that many of the subcomponents can be developed in parallel because of the proprietary XML intercommunication interface. All APIs and interfaces, whether they are made available to customers or not, are carefully documented and signed off for.

1.5 Final Results

1.5.1 Final testing of each model/sub-component

The final testing went well. Each model and sub-component passed the previously mentioned tests. The final product was then released. Because of the stringent quality assurance, the number of customer issues with the software was fairly small.

Table 1 shows the speed of the different configurations. Note that products which use a larger memory size are slower. This is because more alternative translations can be considered.

1.5.2 Error analysis / Acceptance study

Language Weaver decided to conduct an error analysis and acceptance study after the third release of the software. Using its proprietary post-editing evaluation techniques in conjunction with extensive customer interviews, it was determined that the translation accuracy is adequate for many purposes, some of which are listed below. This study also pointed the way forward towards even better translation quality and allowed for the determination of the priority of new statistical modeling approaches and features.

1.5.3 Results

The final release has been received highly positively by the users of the Language Weaver Arabic to English Statistical Machine Translation System. Sample outputs are in Figure 7.

However, Language Weaver output is not perfect. In some instances, the local syntax of noun phrases is mangled. In other instances, the verbs appear in sentence initial position (as in the original Arabic), rather than after the subject. In other instances, prepositions or content words are mistranslated. Efforts at Language Weaver to improve the translation quality are constantly ongoing. Based on the trends we have been observing in terms of significantly improved quality with each release of the software, we have high hopes we will continue to improve quality for the Arabic/English language pair.

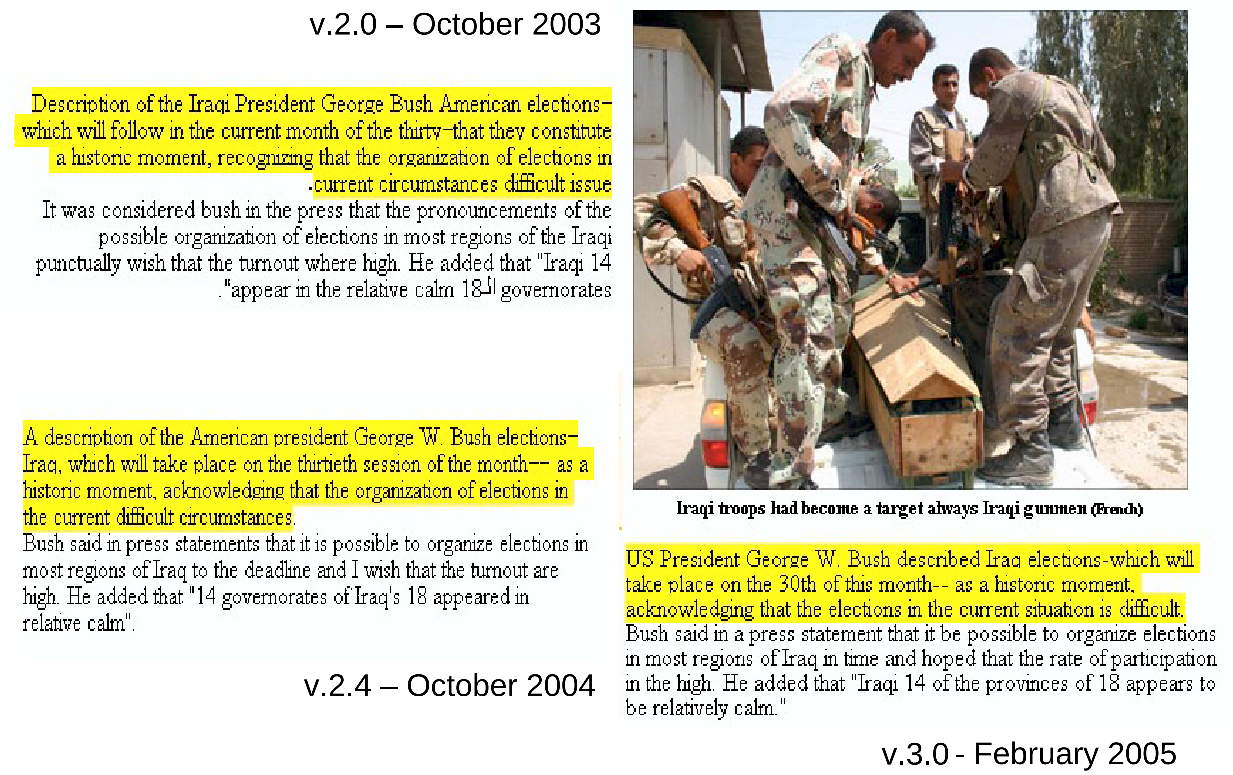
1.5.4 Conclusion

Future Plans For Improving the System

There are several areas for which there are future plans. One plan involves integrating the existing Statistical Machine Translation methodology with statistically based syntactic analysis. This integration will make syntactic analysis robust to noisy inputs, which is a persistent problem in traditional approaches to MT based around syntactic transfer. It will also dramatically increase the coverage of syntactic analysis to less likely phenomena which human linguists would not have time to account for. Syntactic SMT has been an area in which USC/ISI has already conducted a large amount of research, Yamada and Knight (2001, 2002), Koehn and Knight (2003), Galley et al. (2004), and there is a general agreement in the SMT research community that this is an important direction to investigate, Wu (1996), Alshawi et al. (2000), Cherry and Lin (2003), Och et al. (2004), Niessen and Ney (2004), Drabek and Yarowsky (2004), Melamed (2004), Zhang and Gildea (2004), Chiang (2005), Collins et al. (2005), Quirk et al. (2005), Burbank et al. (2005).

Another plan involves using more sophisticated morphology, particularly for the Arabic to English system. There are several approaches being considered for anaphora resolution (recognition and correct translation of pronouns and other referential expressions) which is particularly problematic in Arabic as pronouns are often specified only by ambiguous verb morphology.

Other areas of research under consideration for integration into the product include name translation and improved analysis of parallel corpora. One idea for name translation would be to add a specialized translation component for dealing with words which are transliterated



v.2.0 – October 2003

Description of the Iraqi President George Bush American elections–
which will follow in the current month of the thirty–that they constitute
a historic moment, recognizing that the organization of elections in
.current circumstances difficult issue
It was considered bush in the press that the pronouncements of the
possible organization of elections in most regions of the Iraqi
punctually wish that the turnout where high. He added that "Iraqi 14
."appear in the relative calm 18الـ governorates

A description of the American president George W. Bush elections–
Iraq, which will take place on the thirtieth session of the month–– as a
historic moment, acknowledging that the organization of elections in
the current difficult circumstances.
Bush said in press statements that it is possible to organize elections in
most regions of Iraq to the deadline and I wish that the turnout are
high. He added that "14 governorates of Iraq's 18 appeared in
relative calm".

v.2.4 – October 2004

Iraqi troops had become a target always Iraqi gunmen (French)

US President George W. Bush described Iraq elections-which will
take place on the 30th of this month-- as a historic moment,
acknowledging that the elections in the current situation is difficult.
Bush said in a press statement that it be possible to organize elections
in most regions of Iraq in time and hoped that the rate of participation
in the high. He added that "Iraqi 14 of the provinces of 18 appears to
be relatively calm."

v.3.0 - February 2005

FIGURE 7 Sample Outputs

(or back-transliterated) such as names of people and organizations. Research on this for Arabic has been conducted at USC/ISI, Al-Onaizan and Knight (2002). Approaches to improving the analysis of the parallel text would follow research into combining morphological processing and discriminative training for word alignment, such as in Fraser and Marcu (2005), Ittcheriah and Roukos (2005), Moore (2005), Taskar et al. (2005).

One exciting possibility which is new to the statistical approach to machine translation is adaptation of the statistical models to the customer's corpora. Traditional MT systems allow customers limited access to customize their systems, generally by allowing customers to define new entries in the translation lexicon without giving them the ability to write new rules. Translation memory has similar capabilities to this in that it allows customers to reuse translations which they select. It is not generally possible for the customer to craft her/his own rules because this requires specialized training. The unique opportunity available with statistical systems is to deliver a "learning" component to the customer. This will allow the customer to update the full translation engine after post-editing each translation, thus creating a possibility for improving with each post-edited translation with virtually no effort by the customer. This is possible because the statistics are estimated automatically. This system will be able to learn from documents prepared using the TM Builder tool described previously and data exported from the customers' translation memory tools. This will allow users to customize systems much more quickly and in a more powerful way than the custom editing of lexicon entries available with traditional systems, and will put more of the power of the systems' developers into the hands of users than has been previously possible.

General Conclusions

Language Weaver's Arabic MT system is a state of the art MT system which is self-contained and can be used for a myriad of tasks. In the meanwhile, it has become the system of choice for many customers interested in Arabic to English MT. Without the need for a large team of trained linguists, our system is able to learn a good statistical model of translation which captures many aspects of discourse (such as style).

Arabic to English machine translation is a difficult task and will be an interesting research topic for many years to come. For Arabic, we have worked on a surprisingly high number of low-level issues, and Language Weaver is now moving on to work on higher level modeling issues which will capture some of the finer nuances of Arabic to English translation.

We learned several important lessons. First, we learned that we can use the statistical framework in a reasonable way to arbitrate between different components of the system, some of which are not statistical. We have found that constant error analysis and evaluation methodology are two critical components of all MT efforts. We learned many lessons about the process of technological transfer from academia to industry, something which Language Weaver expects to use frequently in its partnership with USC/ISI.

Finally, the translation results validate the computational and linguistic approaches chosen, and have trail-blazed the path for new statistically-based Natural Language Processing products to be released in the future. We also hope that Language Weaver's Arabic to English Statistical Machine Translation Software system plays a role in increasing the profile of the MT industry as a whole.

The authors would like to thank the entire team at Language Weaver, all of whom played a critical role in producing the machine translation system whose development is described here.

References

Al-Onaizan, Yaser and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 400–408.

Alshawi, Hiyan, Srinivas Bangalore, and Shona Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics* 26(1):45–60.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics* 19(2):263–311.

Burbank, A., M. Carpuat, S. Clark, M. Dreyer, P. Fox, D. Groves, K. Hall, M. Hearne, D. Melamed, Y. Shen, A. Way, B. Wellington, and D. Wu. 2005. Statistical machine translation by parsing: Final report, JHU/CLSP workshop. `http://www.clsp.jhu.edu/ws2005/groups/statistical`.

Callison-Burch, Chris and Philipp Koehn. 2005. Introduction to statistical machine translation. Available from `http://www.statmt.org`.

Charniak, Eugene, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for machine translation. In *MT Summit IX*, pages 40–46. New Orleans, LO.

Cherry, Colin and Dekang Lin. 2003. A probability model to improve word alignment. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*. Sapporo, Japan.

Chiang, David. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 263–270. Ann Arbor, MI.

Collins, Michael, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540. Ann Arbor, Michigan.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B* 39(1):1–22.

Drabek, Elliott Franco and David Yarowsky. 2004. Improving bitext word alignments via syntax-based reordering of english. In *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics*, pages 146–149. Barcelona, Spain.

Fraser, Alexander and Daniel Marcu. 2005. Isi's participation in the romanian-english alignment task. In *ACL 2005 Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*. Ann Arbor, Michigan.

Fraser, Alexander, Jinxi Xu, and Ralph Weischedel. 2002. Trec 2002 cross-lingual retrieval at bbn. In *E.M. Voorhees and D.K. Harman (Eds) The Eleventh Text Retrieval Conference, TREC 2002. NIST Special Publication 500-251..*

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, pages 273–280.

Ittcheriah, Abraham and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia.

Knight, Kevin. 1999a. Decoding complexity in word-replacement translation models. *Computational Linguistics* 25(4):607–615.

Knight, Kevin. 1999b. A statistical machine translation tutorial workbook. `http://www.isi.edu/natural-language/mt/wkbk.rtf`.

Koehn, Philipp and Kevin Knight. 2003. Feature-rich statistical translation of noun phrases. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*. Sapporo, Japan.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, pages 127–133. Edmonton, Canada.

Marcu, Daniel and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*, pages 133–139. Philadelphia, PA.

Melamed, I. Dan. 2004. Statistical machine translation by parsing. In *Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*. Barcelona, Spain.

Moore, Robert C. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia.

Munteanu, Dragos Stefan, Alexander Fraser, and Daniel Marcu. 2004. Improved machine translation performance via parallel sentence extraction from comparable corpora. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*. Boston, Massachusetts.

Niessen, Sonja and Hermann Ney. 2004. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics* 30(2):181–204.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167. Sapporo, Japan.

Och, Franz Josef, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-04)*. Boston.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1):19–51.

Och, Franz Josef and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics* 30(1):417–449.

Papineni, Kishore A., Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Tech. Rep. RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY.

Quirk, Chris, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*. Ann Arbor, MI.

Taskar, Ben, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia.

Wahlster, Wolfgang, ed. 2000. *Verbmobil: Foundations of speech-to-speech translations*. Berlin, Germany: Springer Verlag.

Wu, Dekai. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. of the 34th Annual Conf. of the Association for Computational Linguistics (ACL '96)*, pages 152–158. Santa Cruz, CA.

Xu, Jinxi, Alexander Fraser, and Ralph Weischedel. 2002. Empirical studies in strategies for arabic retrieval. In *Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pages 269–274. Tampere, Finland.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 523–530. Toulouse, France.

Yamada, K. and K. Knight. 2002. A decoder for syntax-based MT. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zhang, Hao and Daniel Gildea. 2004. Syntax-based alignment: Supervised or unsupervised? In *COLING '04: The 20th Int. Conf. on Computational Linguistics*. Geneva, Switzerland.