Machine Translation manuscript No.
(will be inserted by the editor)

Cunei: Open-Source Machine Translation with Relevance-Based Models of Each Translation Instance

Aaron B. Phillips

This is the author's copy which differs in formatting from the final publication.
The final publication is available at `www.springerlink.com`.
DOI: 10.1007/s10590-011-9109-6

Abstract The Cunei Machine Translation Platform is an open-source system for data-driven machine translation. Our platform is a synthesis of the traditional Example-Based MT (EBMT) and Statistical MT (SMT) paradigms. What makes Cunei unique is that it measures the relevance of each translation instance with a distance function. This distance function, represented as a log-linear model, operates over one translation instance at a time and enables us to score the translation instance relative to the specified input and/or the current target hypothesis. We describe how our system, Cunei, scores features individually for each translation instance and how it efficiently performs parameter tuning over the entire feature space. We also compare Cunei with three other open-source MT systems (Moses, CMU-EBMT, and Marclator). In our experiments involving Korean-English and Czech-English translation Cunei clearly outperforms the traditional EBMT and SMT systems.

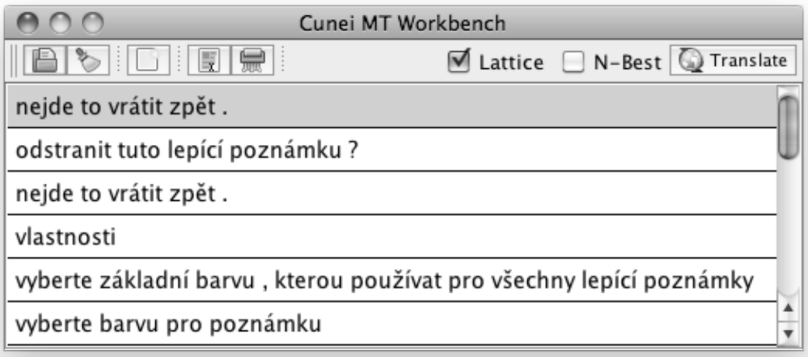
Keywords Machine Translation · SMT · EBMT · Data-Driven · Open-Source

1 Introduction

The Cunei Machine Translation Platform is an open-source system for data-driven machine translation. Our platform is a synthesis of the traditional Example-Based MT (EBMT) and Statistical MT (SMT) paradigms. In particular, our approach moves beyond the notion of modeling each translation as a combination of independent models. Instead, we treat each piece of the translation process as belonging to a broader web of information. The motivation is that this approach allows for a more robust and contextual data-driven translation model.

Cunei provides an open platform for exploring and experimenting with machine translation. In addition to an extensive suite of command line utilities, Cunei includes a simple graphical user interface (Figures 1, 2, and 3). This interface allows the user to visually inspect the translation process and learn what features cause a translation to receive its respective score. The software distribution also includes Apache Ant build

Aaron B. Phillips
Carnegie Mellon University E-mail: aphillips@cmu.edu



nejde to vrátit zpět .
odstranit tuto lepící poznámku ?
nejde to vrátit zpět .
vlastnosti
vyberte základní barvu , kterou používat pro všechny lepící poznámky
vyberte barvu pro poznámku

Fig. 1 Cunei's Graphical Workbench

scripts that can acquire freely available corpora (such as the Europarl) and automatically build a complete MT system. In conjunction, our website includes a tutorial that acquaints new users with Cunei and teaches them how to use it. Our code has been developed for research purposes and we understand that it may contain some rough edges so we also offer a mailing list for support. We take seriously our commitment to open-source and have released the software under the permissive MIT license. Unlike the GPL or LGPL, Cunei can be extended and integrated with academic *and* commercial software. As open-source software, we welcome contributions of code and ideas at `http://www.cunei.org/`.

2 The Problem

A static, log-linear model encoded in a phrase-table provides a convenient format for representing how translations are modeled. This strategy embraced by SMT is easy to understand and conceptually simple to extend or modify. Unfortunately, new features usually need to be carefully crafted to have impact on the translation process. This task of feature engineering is complex, involves significant trial and error, and is often language-dependent. Even worse, without properly engineered features otherwise good research often fails to show improvement.

Consider a situation where additional contextual information is available. For example, assume we have been provided a corpus where each sentence is annotated with a document name, genre, and sentence-level alignment probability. When we model each phrase-pair, it would be beneficial to incorporate the following features:

1. Similarity between the input document and each document in which the phrase-pair is located
2. Similarity between the input genre and the genre of each phrase-pair
3. A preference for phrase-pairs extracted from sentences with high alignment probabilities

It is important to recognize that SMT models *phrase-pairs* with each log-linear model. Phrase pairs are abstract units of translation whose features are computed based on the occurrences of that phrase-pair in the corpus. In practice it is common to assume the features are independent. Thus, we would compute Feature 1 without regard to Features 2 or 3. In some situations this may be 'good enough', but it obviously fails to account for joint dependencies. When joint dependencies are desired, a new feature must be introduced for each possible dependency. For example, adding a binary feature will *double* the size of the feature space if all joint dependencies are modeled. Furthermore, real-valued features cannot be modeled jointly until they are first made discrete (which creates even more issues). As a result, joint dependencies are often not used at all or used very selectively because they dramatically increase the complexity and size of models.

Unfortunately, knowing which features and dependencies to use and which to ignore is not well-defined in SMT. While one might argue that this is the role of parameter tuning, minimum error rate training (MERT) is known to fail when there are more than a few dozen features (Chiang et al, 2008; Och et al, 2004). More sophisticated approaches and increased randomization help, but do not completely alleviate the problem as the error surface contains many local minima. In addition, the cost of calculating all of these joint features and storing them in memory can be equally prohibitive.

As mentioned earlier, a phrase-pair is an abstraction over many translation instances. Yet, the additional context information we have is specific to each translation instance. Going back to our example, there is, thus, no straightforward understanding of how to even represent the document name, genre, or alignment probability for a phrase-pair when these values change from instance to instance of translation. Often, the solution is to take the average over the set of translation instances, but this approach discards precisely that information which our model is attempting to capture–that some translation instances are more relevant than others. Thus, the traditional SMT features for a phrase-pair are by definition more limited in scope because they are an abstract representation of many translation instances.

These two issues–which features and dependencies are important to maintain and what level of abstraction for the phrase-pair is most appropriate–place a heavy burden on the user in determining the 'right' structure of the model. It is deeply unsatisfying that in SMT, which prides itself on well-defined statistical models, the process for incorporating new features into these models relies on heuristics, intuition, and trial-and-error. An illustrative example is the paper by (Shen et al, 2009) that states clearly their unique contribution is that "Feature functions defined in this way are robust and ideal for practical translation tasks." The types of linguistic and contextual information their features capture is not new, but as they demonstrate, other approaches have been imperfect and have struggled to incorporate it. In a complex system like machine translation, we do not always know *a priori* what information is useful and how it should be expressed; we would prefer if the model took care of this automatically.

3 Approach

Due to ever-increasing bilingual corpora, data-driven machine translation is able to locate many potential translation instances associated with each input phrase. Yet, these translation instances are not all equally suitable for the translation task at hand. Each translation instance retrieved from the corpus occurs within a unique linguistic context. For example, each translation instance–even if it predicts the same target hypothesis–may be associated with a different parse tree or morpho-syntactic information. What makes Cunei unique is that it is able to score this information (by comparing it to the input or target hypothesis) *individually* for each instance of translation. We exploit this per-instance information to model that some translation instances are more relevant than others.

The translation process begins by encoding in a lattice all possible contiguous phrases from the input. For each source phrase in the lattice, Cunei locates instances of it in the corpus and then identifies the aligned target phrase(s). Each *occurrence* of an aligned source and target phrase is extracted as a unique translation instance. A distance function, represented by a log-linear model, scores the relevance of each of these translation instances. The model then sums the scores of translation instances that predict the same target hypothesis.

3.1 Formalism

In order to compose a complete sentence, data-driven machine translation systems identify small units of translation and select the fragments that when combined together yield the best score. The decision rule given the source sequence $s_1, s_2 ... s_n$ is:

$$\tilde{t} = \arg \max_{t_1, t_2 ... t_n} \sum_{i=0}^{n} m(s_i, t_i, \lambda) \tag{1}$$

Here a model m scores the translation unit which consists of a target phrase t_i and the corresponding source span s_i. The sequence of target phrases $t_i, t_2, ... t_n$ that maximize the score compose the target sentence $\tilde{t}$.

A typical SMT model will score each translation unit using a log-linear model with features θ and weights λ:

$$m(s, t, \lambda) = \sum_i \lambda_i \theta_i(s, t) \tag{2}$$

An EBMT system identifies features for each example in the corpus. There is no prototypical model for EBMT, but Equation 3 demonstrates one method that calculates the total score by summing over all examples with source s' and target t' that are consistent with the phrase pair being modeled. In the strictest form $s = s'$ and $t = t'$, but typically an EBMT system will have some notion of similarity and use examples that do not exactly match the input. Notice that the feature function ϕ in this model is provided a specific example (s', t').

$$m(s, t, \lambda) = \ln \sum_{s', t'} e^{\sum_i \lambda_i \phi_i(s, s', t', t)} \tag{3}$$

Finding the derivative of the above model or recomputing the score for a new λ' (necessary during optimization) requires the system to retain the features ϕ for every example used in translation. This is unfeasible for a real-world translation system as even an artificially small corpus will contain an enormous number of translation examples.

Even though the summation of multiple log-linear models is no longer log-linear, we can approximate the summation with another log-linear model. Cunei's approach, as shown below, is to approximate the EBMT model in Equation 3 with a first-order Taylor series. (Similarly, we could also extend this to a higher-order Taylor approximations.)

$$\begin{aligned} m(s,t,\lambda') &\approx m(s,t,\lambda) + \sum_q (\lambda'_q - \lambda_q) \frac{\partial}{\partial \lambda_q} m(s,t,\lambda) \\ &\approx \ln \sum_{s',t'} e^{\sum_i \lambda_i \phi_i(s,s',t',t)} + \sum_q (\lambda'_q - \lambda_q) \frac{\sum_{s',t'} \phi_q(s,s',t',t) e^{\sum_i \lambda_i \phi_i(s,s',t',t)}}{\sum_{s',t'} e^{\sum_i \lambda_i \phi_i(s,s',t',t)}} \end{aligned}$$

By rearranging the terms, we can simplify Cunei's model to Equation 4, which is simply another log-linear model. The offset δ is dependent on the original λ, computed once, and does not change with λ'. While the new feature function ψ initially appears complex, it is simply the expectation of each feature under the distribution of translation instances and can be efficiently computed with an online update.

$$m(s,t,\lambda') = \delta + \sum_q \lambda'_q \psi_q \tag{4}$$

$$\delta = \ln \sum_{s',t'} e^{\sum_i \lambda_i \phi_i(s,s',t',t)} - \sum_q \lambda_q \psi_q$$

$$\psi_q = \frac{\sum_{s',t'} \phi_q(s,s',t',t) e^{\sum_i \lambda_i \phi_i(s,s',t',t)}}{\sum_{s',t'} e^{\sum_i \lambda_i \phi_i(s,s',t',t)}}$$

Reducing the collection of instances to one log-linear model considerably simplifies decoding and optimization. Furthermore, this approximation exactly represents the score and the gradient of the entire collection of log-linear models when $\lambda = \lambda'$. Only during optimization, when the parameters are modified, will the reduced model be an approximation.

Interestingly, calculating the model in this manner ties together the two different modeling approaches pursued by SMT and EBMT. First, our model allows us to integrate instance-specific features $\phi(s,s',t',t)$ while maintaining the simplicity of a log-linear model. Second, the SMT model of Equation 2 is merely a special case of our model when the features for all instances of a translation are constant such that $\phi_k(s,s',t',t) = \theta_k(s,t)\ \forall s',t'$.

4 Run-Time Execution

Sufficient statistics for each instance of a translation could be computed off-line, but the space requirements would be quite large. Furthermore, most of the model is unnecessary for a particular input. As a result, Cunei delays as much computation as possible until run-time. In particular, translations are not retrieved from a pre-built phrase-table, but rather generated dynamically at run-time. The remainder of this section will describe in detail how these translations are constructed.

4.1 Locating Matches

In order to compose a translation, Cunei must be able to locate instances of translation that match the input. To support this retrieval, Cunei constructs a suffix array index for each type of sequence present in the parallel corpus. Suffix arrays provide a compact and efficient data structure for locating arbitrary sequences of tokens within a large corpus (Yamamoto and Church, 2001). The search algorithm has a worst-case time complexity of $O(m \log_2 n)$ where n is the number of tokens in the index and m is the number of tokens in the phrase being looked up.[1] As evidenced by the work of Brown (2004), Zhang and Vogel (2005), Callison-Burch et al (2005), and Lopez (2008), suffix arrays are also increasingly popular in machine translation. Furthermore, Cunei is capable of maintaining multiple indexes if we want to store and query additional information from the corpus. For example, one might index lemmas, part-of-speech, or statistical cluster labels in addition to the lexical type sequence.

When Cunei is presented with a sentence to translate, the corpus is scoured for all partial matches of the input. For each type of sequence present in the input, the respective suffix array for that type of sequence is queried. A match may contain as few as one of the tokens from the input or exactly match the entire input. We start searching for the smallest possible match–one token–and incrementally add one more token to the sequence. We are able to efficiently search for additional tokens by storing with each match its bounds in the suffix array. Furthermore, once we are no longer able to locate a particular sequence of tokens in the corpus, we can also stop searching for any sequences that subsume it. The collection of corpus matches is stored in a lattice with each entry indexed by the span of the input it covers.

In addition, Cunei is capable of locating translation instances that are not exact matches of the input. For example, a source phrase retrieved by matching only a part-of-speech sequence may be structurally similar to the input, but it is likely to be semantically unrelated. Matches such as these do not as-is provide valid translations, but they do still contain useful information about the translation process. This has not been a focus of our present research, but in the future we plan to exploit this capability to model similar and discontiguous phrases.

4.2 Sampling Matches

The process of locating all matches in the corpus is relatively cheap, but what we do with each match–notably perform alignment and generate a translation instance–is expensive. Furthermore, we have found in practice that using every last match is overkill and an adequate model can be constructed using a much smaller sample. Without an alignment, there is very little we can do to determine the relevance of a translation instance. As a result, we simply sample the matches uniformly. The exception is that all complete matches (i.e. those whose prefix is the start-of-sentence marker and whose suffix is the end-of-sentence marker) are selected first. When there are fewer matches in the corpus than the desired sample size, then we select all of them. Conveniently, this means only high-frequency words and phrases will be sampled. Typically we will

[1] By storing an additional data structure for the longest common prefix between neighboring rows in the suffix array, it is possible to reduce the search time to $O(m + \log_2 n)$ (Manber and Myers, 1990), but this is not currently implemented in our system.

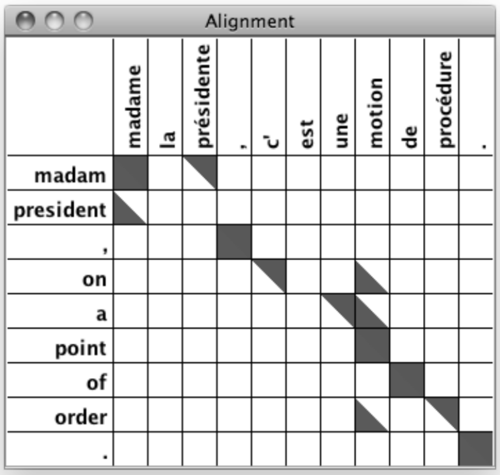


Fig. 2 Word Alignment Visualization. Alignment in a single direction is represented by a triangle; when an entire cell is shaded then both directions of the GIZA++ alignments agree.

extract a few hundred matches (this is configurable by the user) and the reduced set is then passed on for alignment and scoring.

4.3 Identifying Aligned Phrase(s)

After a match is found on the source-side of the corpus, Cunei determines possible target phrase alignments. Ideally, the complete alignment process would be computed on-demand at run-time, but this is prohibitively expensive even with a simple IBM Model-1. Instead, we run a word-aligner once while building the system and store the word alignments as part of the indexed corpus. Phrase alignment is then performed at *run-time* and is incorporated as part of the translation instance's log-linear model. The saved word alignments are used to generate the features for phrase alignment. While the calculations are not exactly the same, conceptually this work is modeled after Vogel (2005).

For each source-side match in the corpus, we load the alignment matrix for the complete sentence in which the match resides. The alignment matrix, as visually depicted in Figure 2, contains scores for all possible word correspondences in the sentence-pair. Each word alignment link maintains two scores: α_s and α_t. When using GIZA++ (Och and Ney, 2003) to generate the initial word alignments, $P(s_i|t_j)$ will be stored as $\alpha_s(i,j)$ and $P(t_j|s_i)$ as $\alpha_t(i,j)$. Cunei also supports initializing the alignment matrix using the Berkeley aligner (Liang et al, 2006) which symmetrizes the probability model. In this case, $\alpha_s(i,j)$ and $\alpha_t(i,j)$ will both be set to $P(s_i,t_j)$. While both GIZA++ and Berkeley model probability distributions, the α_s and α_t scores need not be normalized for our calculations.

Let $\alpha_s(i,j)$ and $\alpha_t(i,j)$ be the alignment score between the source word at position i and target word at position j (from the external word aligner).

Outside Probability

Let the set of positions in the source phrase and target phrase that are outside the phrase alignment be, respectively, s_{out} and t_{out}.

`Alignment.Weights.Outside.Source.Probability`	$\sum_{i \in s_{out}} \log \frac{\epsilon + \sum_{j \in t_{out}} \alpha_t(i,j)}{\epsilon + \sum_j \alpha_t(i,j)}$
`Alignment.Weights.Outside.Target.Probability`	$\sum_{j \in t_{out}} \log \frac{\epsilon + \sum_{i \in s_{out}} \alpha_s(i,j)}{\epsilon + \sum_i \alpha_s(i,j)}$

Inside Probability

Let the set of positions in the source phrase and target phrase that are inside the phrase alignment be, respectively, s_{in} and t_{in}.

`Alignment.Weights.Inside.Source.Probability`	$\sum_{i \in s_{in}} \log \frac{\epsilon + \sum_{j \in t_{in}} \alpha_t(i,j)}{\epsilon + \sum_j \alpha_t(i,j)}$
`Alignment.Weights.Inside.Target.Probability`	$\sum_{j \in t_{in}} \log \frac{\epsilon + \sum_{i \in s_{in}} \alpha_s(i,j)}{\epsilon + \sum_i \alpha_s(i,j)}$

Inside Unknown

The user-defined threshold θ identifies the value below which an an alignment score is considered uncertain.

`Alignment.Weights.Inside.Source.Unknown`	$\sum_{i \in s_{in}} \max(0, \frac{\theta - \left(\epsilon + \sum_j \alpha_t(i,j)\right)}{\theta})$
`Alignment.Weights.Inside.Target.Unknown`	$\sum_{j \in t_{in}} \max(0, \frac{\theta - \left(\epsilon + \sum_i \alpha_s(i,j)\right)}{\theta})$

Table 1 Description of Phrase Alignment Features

Using this alignment matrix, each source phrase is modeled as having some probability of aligning to every possible target phrase within a given sentence. When a source phrase is aligned to a target phrase, it not only implies that words within the source phrase are aligned to words within the target phrase, but also that the remainder of the source sentence not specified by the source phrase is aligned to the remainder of the target sentence not specified by the target phrase. As detailed in Table 1, we define separate features to model the probability that the alignment links for tokens within the phrase are concentrated within the phrase boundaries and that the alignment links for tokens outside the phrase are concentrated outside the phrase boundaries. In addition, tokens within the phrase that are not aligned or have weak alignments demonstrate uncertainty in modeling. To capture this effect, we incorporate two more features that measure the number of uncertain alignment links included within the phrase alignment boundaries.

Phrase Coverage

Let s and t represent the source and target for this translation instance and S represent the entire input sentence.

`Translation.Weights.Spans`	1
`Translation.Weights.Coverage`	$\log \frac{\lvert s\rvert}{\lvert S\rvert}$
`Translation.Weights.Null`	$\begin{cases} 1 & \text{if } \lvert t\rvert = 0 \\ 0 & \text{otherwise} \end{cases}$

Phrase Frequency

The function $c(x)$ returns the number of occurrences in the corpus of the phrase x.

`Translation.Weights.Frequency.Correlation`	$\frac{(c(s)-c(t))^2}{(c(s)+c(t)+1)^2}$
`Translation.Weights.Frequency.Source`	$-\log(c(s))$
`Translation.Weights.Frequency.Target`	$-\log(c(t))$

Lexical Probability

Let x_i denote the word at position i in phrase x. $P(s_i|t_j)$ and $P(t_i|s_j)$ are relative frequency counts over all word alignments in the corpus.

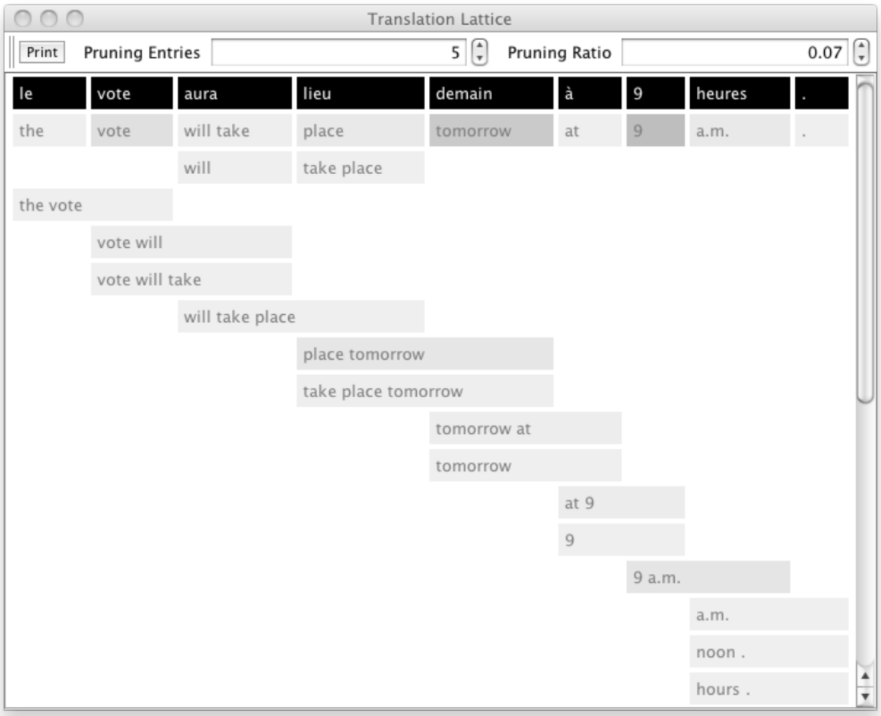
`Lexicon.Weights.Lexical.Source`	$\sum_{i=0}^{\lvert s\rvert} \max_{j=0}^{\lvert t\rvert} \log P(s_i\lvert t_j)$
`Lexicon.Weights.Lexical.Target`	$\sum_{i=0}^{\lvert t\rvert} \max_{j=0}^{\lvert s\rvert} \log P(t_i\lvert s_j)$

Table 2 Description of Static, SMT-like Features

For each match in the corpus, Cunei uses these feature functions to extract a scored n-best list of possible phrase alignments. The size of the n-best list is controlled by two user-defined pruning parameters: a maximum number of elements and a maximum ratio between the best and worst score. In practice, 3 to 6 phrase alignments are typically selected. Each possible alignment forms a new instance of translation between the source and target phrase.

4.4 Additional Static Scores

Once all translation instances are retrieved from the corpus, we can also apply more general, static, SMT-like features. This is possible because our distance function for each translation instance takes the same form as a standard SMT log-linear model. We simply perform the standard SMT feature calculations over the set of retrieved translation instances and then apply those feature uniformly to all the instances. Cunei currently computes the features detailed in Table 2 which model a translation's overall


Translation Lattice

Print Pruning Entries 5 Pruning Ratio 0.07

le	vote	aura	lieu	demain	à	9	heures	.
the	vote	will take	place	tomorrow	at	9	a.m.	.
		will	take place					
the vote								
	vote will							
	vote will take							
		will take place						
			place tomorrow					
			take place tomorrow					
				tomorrow at				
				tomorrow				
					at 9			
					9			
						9 a.m.		
							a.m.	
							noon .	
							hours .	

Fig. 3 A lattice of translations

frequency in the corpus, lexical probability, and coverage. These particular features are static in the sense that they do not change from instance to instance if they share the same source and target phrases. We include these to ensure that our model has no less discriminative power than a standard SMT system. While the emphasis is placed on Cunei's ability to use per-instance features, in this manner, Cunei can also take advantage of features computed over sets of instances or loaded from external models.

4.5 Combining Partial Translations

Cunei synthetically combines a lattice of partial translations into a complete sentence using a statistical decoder. Recall that the set of matches retrieved from the corpus particular to the input sentence are stored in a lattice. These matches are aligned to form partial translations which are in turn stored in another lattice as illustrated in Figure 3. The decoder then searches this latter lattice for a set of partial translations with the minimum score that completely cover the input.

The decision of which partial translations to combine is informed by the scores of the partial translations and the decoding features detailed in Table 3. In order to compensate for divergences between the source and target language, Cunei may need to reorder the partial translations. Reordering is modeled by counting the total number of re-orderings and by keeping track of the total distance that words have been

Reordering

Let the first position of the source span for the current partial translation be i and the last position of the source span for the previous partial translation be j.

`Hypothesis.Weights.Reorder.Count`	$\begin{cases} 1 & \text{if } i - j \neq 1 \\ 0 & \text{otherwise} \end{cases}$
`Hypothesis.Weights.Reorder.Distance`	$\lvert i - j - 1 \rvert$

Language Model

Multiple language models can be used; these refer to the model identified as `Default`*. Let the order of the language model be denoted by n and the target sequence be represented as $w_0 w_1 w_2 ... w_n$.*

`LanguageModel.Default.Weights.Probability`	$\sum_{i=0}^{n} \log P(w_i \mid w_{i-i} w_{i-2} ... w_{i-n+1})$
`LanguageModel.Default.Weights.Unknown`	$\sum_{i=0}^{n} \begin{cases} 1 & \text{if } w_i \text{ is unknown} \\ 0 & \text{otherwise} \end{cases}$

Sentence Length

Let the phrase x contain $\lvert x \rvert_{word}$ words and $\lvert x \rvert_{char}$ characters. The mean, μ, and variance, σ^2, of both word and character lengths are calculated over the corpus.

`Sentence.Weights.Length.Words`	$\lvert t \rvert_{word}$
`Sentence.Weights.Ratio.Word`	$-\frac{(\lvert s \rvert_{word} * \mu_{word} - \lvert t \rvert_{word})^2}{\sigma^2 (\lvert s \rvert_{word} * \mu_{word} + \lvert t \rvert)}$
`Sentence.Weights.Ratio.Character`	$-\frac{(\lvert s \rvert_{char} * \mu_{char} - \lvert t \rvert_{char})^2}{\sigma^2 (\lvert s \rvert_{char} * \mu_{char} + \lvert t \rvert)}$

Table 3 Description of Decoder Features

moved. Additionally, the probability of the complete target sequence is estimated with a statistical language model. Any number of language models can be used simultaneously during decoding and each will generate its own set of features. In order to offset the tendency of the language model(s) to prefer short output, we balance the overall score by including a feature that simply counts the number of words present in the target. The complete sentence length is also modeled based on the ratio of source words to target words. When combined, these features instruct the decoding process which composition of partial translations to select.

5 Optimization

Because Cunei uses a log-linear model, we can optimize it using the same approaches developed for SMT. The most common technique, Minimum Error Rate Training

(MERT), iteratively generates an n-best list of translations from one set of weights and finds a new set of weights that maximize an objective function (typically BLEU) (Och, 2003). Due to pruning and the beam search within the decoder, the new weights may yield different translations in the n-best list. This is traditionally remedied by merging the n-best lists after each iteration to obtain a larger representation of the search space. In Cunei, new weights can also change what translations are modeled at run-time and the translation's estimated feature scores. Recall that Cunei's model is only an approximation once the weights are modified. In order to manage this variability, we discount models that were estimated far from the current set of weights.

Because Cunei must also manage a large feature space, MERT is not actually an ideal choice. Instead, Cunei's optimization code closely follows the approach of Smith and Eisner (2006). Conceptually, this approach is similar to MERT except it uses an objective function that minimizes the expected loss over the distribution of translations present in the entire n-best list. In its default configuration, Cunei will optimize toward BLEU using the following objective function that in log-space sums the expected value of BLEU's brevity penalty and precision score:

$$(1 + e^{\mu(h)-\mu(r)})(\frac{\mu(|r|)}{\mu(h)} e^{\frac{\sigma(h)}{2\mu(h)^2} - \frac{\sigma(r)}{2\mu(r)^2}} - 1) + \frac{\sum_{n=1}^{4} \log(\mu(t_n)) - \frac{\sigma(t_n)}{2\mu(t_n)^2} - \log(\mu(c_n)) + \frac{\sigma(c_n)}{2\mu(c_n)^2}}{4}$$

$$p_i = \frac{e^{\gamma m_i}}{\sum_k e^{\gamma m_k}} \qquad \mu(x) = \sum_i p_i x_i \qquad \sigma(x) = \sum_i p_i (x_i - \mu(x))^2$$

m_i	Log-score of hypothesis i in the n-best list
γ	Gamma (used for annealing)
h	Length of the hypothesis
r	Length of the selected (shortest or closest) reference
c_n	"Modified count" of matching n-grams according to BLEU
t_n	Total number of n-grams present in the hypothesis

Following Smith and Eisner (2006), the distribution of the n-best list is slowly annealed in order to avoid local minima. This is carried out by multiplying the log-score of each translation by a γ parameter; the default schedule initializes γ to 0.25 and doubles it after convergence. Hence, we begin with a mostly flat distribution and mildly peak the distribution each time the objective function converges. Eventually this process reaches a distribution where, for each sentence, nearly all of the probability mass resides on one translation. In the early stages, sharpening the distribution is often the quickest way to minimize the expected loss. While γ is fixed until convergence, the same effect can be achieved by uniformly increasing the magnitude of all the other weights. To address this weight creep, Cunei augments the objective function with an L2 normalization term. In addition, Cunei supports the ability during decoding to select hypotheses that are most similar to a particular set of references. Often we will initialize the n-best list with these translations to guide the optimization process toward high-scoring, obtainable translations.

	Korean	English
Size of Vocabulary	34,635	10,584
Number of Words	172,641	252,982
Number of Sentences	26,719	

Table 4 Korean-English Corpus Statistics

6 Experiments

Our evaluation was performed in two scenarios that offer unique challenges–translating from Korean to English and from Czech to English. In addition, we compare Cunei with three alternative open-source machine translation systems: Moses, CMU-EBMT, and Marclator. Moses[2] is a widely-used SMT system with many extensions (Koehn et al, 2007). We built Moses using its default configuration which includes a phrase-based model and bidirectional lexicalized reordering. CMU-EBMT[3] is a shallow EBMT system from Carnegie Mellon University (Brown, 1996, 2004). This system does not require examples to be linguistic constituents and weights each example based on document similarity and local context. Marclator is another EBMT system developed by Dublin City University and available as part of OpenMaTrEx (Stroppa and Way, 2006)[4]. It is more limited than the other systems in that it can only generate monotonic output (no reordering). However, Marclator is perhaps the most traditional EBMT system in this mix as it identifies translation chunks based on the Marker Hypothesis (Green, 1979). Unfortunately, the Marclator distribution does not include marker information for Korean, so we could only include this system in the Czech to English experiments.

In order to build comparable systems, we managed the training resources and provided the same files to each system. All the training resources were processed with our own simple normalization and tokenization routines. Using the parallel resources and GIZA++ (Och and Ney, 2003), we trained IBM Model-4 word alignments in both directions. Using the monolingual resources and the SRILM toolkit (Stolcke, 2002), we trained a 5-gram English language model with Kneser-Ney smoothing. Each MT system was provided the tokenized corpus, word alignments, and language model. The systems then used their respective phrase extraction, model estimation routines, and tuning methods. Evaluation scores were computed with BLEU (Papineni et al, 2002) and NIST (Doddington, 2002).

Our Korean-English parallel text is quite small, weighing in at just over 25,000 sentences. This particular scenario challenges an MT systems to work with very little information. The corpus consists of conversational sentences related to business and travel. From the same domain we withheld 966 sentences for development and 1170 sentences for testing. Lacking additional in-domain text, we trained the English language model solely on the English half of the parallel corpus. Statistics describing the training corpus are shown in Table 4 and the results of our experiments are in Table 5.

The scores for all three systems are low due to the extremely limited training data, but Moses is clearly the weakest contender. The occurrences of phrase-pairs are far too sparse for accurate relative frequency estimation (the basis of Moses' feature functions). In addition, Cunei's model has allowed it to select longer translations. Cunei's output closely matches the length of the references, whereas CMU-EBMT uses

[2] `http://www.statmt.org/moses/`

[3] `http://sourceforge.net/projects/cmu-ebmt/`

[4] `http://www.openmatrex.org/`

	Development		**Test**	
	BLEU	NIST	BLEU	NIST
Moses	0.1431	3.7208	0.1278	3.5973
CMU-EBMT	0.1888	4.2893	0.1923	4.3560
Cunei	0.2095	4.5291	0.2200	4.6919

Table 5 Korean-English Evaluation

approximately 5% fewer words and Moses has 10% fewer words. This situation results in Cunei producing the best output and, more generally, EBMT systems outperforming SMT.

Next we trained each system for Czech to English translation. We created a parallel corpus by combining text from version 6 of the Europarl (Koehn, 2005), the 2011 edition of parallel news commentary released by the 2011 Workshop on Statistical Machine Translation[5] (WMT'11), and CzEng 0.9 (Bojar and Žabokrtský, 2009). CzEng is a large collection of many different texts including works of fiction, websites, subtitles, and technical documentation. The motivation for this scenario is that it offers the MT systems multiple genres to translate. We created a 763 sentence development set and 1506 sentence test set by uniformly sampling each genre from a held-out portion of CzEng. As shown in Table 6, we also sampled from the parallel text to create two sizes of parallel training data. With both sizes of training data, we used the same language model trained on over 500 million English words. The monolingual text included the English half of the parallel corpora and years 2010 and 2011 of web-crawled news text released by WMT'11.

Our experiments are presented in Table 7 and examples of translations from all four systems can be found in Tables 8, 9, and 10. Marclator is at a disadvantage because it does not allow chunk re-ordering, but it also tends to select chunks that contain extraneous words. Now with a reasonable quantity of training data, Moses performs quite well and bests CMU-EBMT, but not Cunei.

All of these systems incorporate a language model, but Moses and Cunei seem to be slightly more biased toward fluency. For example, the CMU-EBMT translation in Table 10 contains more key concepts, but the overall translation is stilted by the frequent insertion of determiners. Moses and Cunei provide translations that lose a little in adequacy, but yield greater fluency. The right balance between adequacy and fluency is debatable, but the behavior of Moses and Cunei is likely due to these two systems more closely matching BLEU during training.

Table 9 illustrates an obvious discrepancy between the three EBMT systems and Moses (the one SMT system). Moses is attempting to cobble together a (mostly) word-by-word translation, while the EBMT systems have either matched the entire sentence or are using much longer phrases. Matching extremely long phrases is the exception,

[5] `http://www.statmt.org/wmt11/`

	Small		**Large**	
	Czech	English	Czech	English
Size of Vocabulary	252,982	162,199	601,286	343,262
Number of Words	9,310,976	10,579,280	37,230,252	42,321,891
Number of Sentences	829,403		3,317,055	

Table 6 Czech-English Corpus Statistics

	Small				Large			
	Development		**Test**		**Development**		**Test**	
	BLEU	NIST	BLEU	NIST	BLEU	NIST	BLEU	NIST
Marclator	0.1463	5.1988	0.1412	5.2789	0.1809	5.5969	0.1822	5.7488
CMU-EBMT	0.2753	6.0908	0.2457	6.2206	0.3149	6.5265	0.2890	6.7069
Moses	0.2805	6.4915	0.2580	6.6857	0.3377	7.0828	0.3146	7.3120
Cunei	0.3073	6.7194	0.2943	7.0738	0.3752	7.4062	0.3624	7.7838

Table 7 Czech-English Evaluation

but a more general improvement in phrase selection and matching is still present in the other examples. In Table 8, Cunei selects the phrase "raised his eyes from the instrument panel". Cunei's output is the most fluent, but the same idea is also present in Marclator and CMU-EBMT. On the other hand, Moses uses "glanced up from the the instrument panel" which is also fluent, but does not match the reference. Perhaps more clearly, in Table 10 Cunei generates "when i leave dance floor" while Moses simply has "to leave dance floor" which is neither more fluent or more adequate.

We found that Cunei and Moses often render similar output that differs in the lexical selection of one or two words per sentence. Unlike Moses, Cunei's model enables it to selectively weight instances of translation based on their relevance. Overall, we find Cunei's translations to form a balance between the strengths of traditional SMT and EBMT systems.

7 Summary

As illustrated in both Table 5 and Table 7, Cunei is state-of-the-art across several language-pairs and bests its competition. In recent years SMT has dominated the field of machine translation research. Even though SMT in many respects grew from the data-driven focus of EBMT, the concept of modeling each translation instance indi-

Reference	`slowly , he raised his eyes from the instrument board and stared out the window .`
Marclator	`slowly up eyes the pilot oswaldo zileri and looked out the window`
CMU-EBMT	`slowly his eyes from the instrument board and , from windows .`
Moses	`slowly he glanced up from the the instrument panel he looked out the window .`
Cunei	`slowly raised his eyes from the instrument panel and looked out of the window .`

Table 8 Czech-English (Large) Translation Example 1

Reference	`rich is the guy behind the cream team , is that correct ?`
Marclator	`rich is the guy behind the cream team is that correct ?`
CMU-EBMT	`rich is the guy behind the cream team , is that correct ?`
Moses	`so cream team put together rich ?`
Cunei	`rich is the guy behind the cream team , is that correct ?`

Table 9 Czech-English (Large) Translation Example 2

Reference	`when i leave the dance floor , go upstairs with carina to distract the security guards .`
Marclator	`to i floor go upstairs with carinou away guardian`
CMU-EBMT	`to leave a dance floor , go upward with carinou a distraction a guardian .`
Moses	`to leave dance floor , go up with carinou away the guard .`
Cunei	`when i leave dance floor , go up with carinou away bodyguard .`

Table 10 Czech-English (Large) Translation Example 3

vidually has been lost. Cunei's translation model that incorporates instance-specific features outperforms comparable SMT and EBMT systems. We are encouraged by these results as our approach opens the door for new sources of knowledge that describe each instance of translation individually. Ultimately, Cunei should be able to use any available information, be it lexical, syntactic, semantic, grammatical, pragmatic, contextual, etc., to make a case-by-case selection of the best possible instance of translation in its corpus–fulfilling the goal of true 'data-driven' machine translation.

References

Bojar O, Žabokrtský Z (2009) CzEng 0.9: Large parallel treebank with rich annotation. Prague Bulletin of Mathematical Linguistics 92

Brown RD (1996) Example-based machine translation in the pangloss system. In: Proceedings of the 16th International Conference on Computational Linguistics, Copenhagen, Denmark, pp 169–174

Brown RD (2004) A modified burrows-wheeler transform for highly scalable example-based translation. In: Frederking RE, Taylor K (eds) Machine Translation: From Real Users to Research, 6th Conference of the Association for Machine Translation in the Americas, Springer, Washington, DC, USA, pp 27–36

Callison-Burch C, Bannard C, Schroeder J (2005) Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, Ann Arbor, USA, pp 255–262

Chiang D, Marton Y, Resnik P (2008) Online large-margin training of syntactic and structural translation features. In: 2008 Conference on Empirical Methods in Natural Language Processing, Honolulu, USA, pp 224–233

Doddington G (2002) Automatic evaluation of machine translation quality using n-gram cooccurrence statistics. In: Proceedings of the Human Language Technology Conference, San Diego, CA, pp 128–132

Green T (1979) The necessity of syntax markers: Two experiments with artificial languages. Journal of Verbal Learning and Behavior 18:481–496

Koehn P (2005) Europarl: A parallel corpus for statistical machine translation. In: Machine Translation Summit X Proceedings, Phuket, Thailand, pp 79–86

Koehn P, Hoang H, Birch A, Callison-Burch C, Federico M, Bertoldi N, Cowan B, Shen W, Moran C, Zens R, Dyer C, Bojar O, Constantin A, Herbst E (2007) Moses: Open source toolkit for statistical machine translation. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Prague, Czech Republic, pp 177–180

Liang P, Taskar B, Klein D (2006) Alignment by agreement. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, New York City, USA, pp 104–111

Lopez A (2008) Tera-scale translation models via pattern matching. In: Proceedings of the 22nd International Conference on Computational Linguistics, Manchester, United Kingdom, pp 505–512

Manber U, Myers G (1990) Suffix arrays: a new method for on-line string searches. In: SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp 319–327

Och FJ (2003) Minimum error rate training in statistical machine translation. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Sapporo, Japan, pp 160–167

Och FJ, Ney H (2003) A systematic comparison of various statistical alignment models. Computational Linguistics 29(1):19–51

Och FJ, Gildea D, Khudanpur S, Sarkar A, Yamada K, Fraser A, Kumar S, Shen L, Smith D, Eng K, Jain V, Jin Z, Radev D (2004) Final report of johns hopkins 2003 summer workshop on syntax for statistical machine translation. Tech. rep., Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD

Papineni K, Roukos S, Ward T, Zhu WJ (2002) BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, USA, pp 311–318

Shen L, Xu J, Zhang B, Matsoukas S, Weischedel R (2009) Effective use of linguistic and contextual information for statistical machine translation. In: 2009 Conference on Empirical Methods in Natural Language Processing, Suntec, Singapore, pp 72–80

Smith DA, Eisner J (2006) Minimum risk annealing for training log-linear models. In: Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia, pp 787–794

Stolcke A (2002) SRILM - an extensible language modeling toolkit. In: 7th International Conference on Spoken Language Processing, Denver, USA, pp 901–904

Stroppa N, Way A (2006) MaTrEx: DCU machine translation system for IWSLT 2006. In: Proceedings of the International Workshop on Spoken Language Translation, Kyoto, Japan, pp 31–36

Vogel S (2005) PESA: Phrase pair extraction as sentence splitting. In: Machine Translation Summit X Proceedings, Phuket, Thailand, pp 251–258

Yamamoto M, Church KW (2001) Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. Computational Linguistics 27(1):1–30, DOI http://dx.doi.org/10.1162/089120101300346787

Zhang Y, Vogel S (2005) An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In: Proceedings of the Tenth Annual Conference of the European Assocation for Machine Translation, Budapest, Hungary, pp 294–301