Discriminative Substring Decoding for Transliteration

Colin Cherry and **Hisami Suzuki**
Microsoft Research
One Microsoft Way
Redmond, WA, 98052
{colinc,hisamis}@microsoft.com

Abstract

We present a discriminative substring decoder for transliteration. This decoder extends recent approaches for discriminative character transduction by allowing for a list of known target-language words, an important resource for transliteration. Our approach improves upon Sherif and Kondrak's (2007b) state-of-the-art decoder, creating a 28.5% relative improvement in transliteration accuracy on a Japanese katakana-to-English task. We also conduct a controlled comparison of two feature paradigms for discriminative training: indicators and hybrid generative features. Surprisingly, the generative hybrid outperforms its purely discriminative counterpart, despite losing access to rich source-context features. Finally, we show that machine transliterations have a positive impact on machine translation quality, improving human judgments by 0.5 on a 4-point scale.

1 Introduction

Transliteration occurs when a word is borrowed into a language with a different character set. The word is transcribed into the new character set in such a way as to maintain rough phonetic correspondence; for example, the English word *hip-hop* becomes ヒップホップ [*hippuhoppu*], when transliterated into Japanese. A task frequently of interest to the NLP community is back-transliteration, where one seeks the original word, given the borrowed form.

We investigate machine transliteration as a method to handle out-of-vocabulary items in a Japanese-to-English translation system. More often than not, this will correspond to back-transliteration. Our goal is to prevent the copying or deletion of Japanese words when they are missing from our statistical machine translation (SMT) system's translation tables. This can have a substantial impact on the quality of SMT output, transforming translations of questionable usefulness, such as:

Avoid using a フリーメール account.[1]

into the far more informative:

Avoid using a Freemail account.

Though the techniques we present here are language-independent, we focus this study on the task of Japanese katakana-to-English back-transliteration. Katakana is one of the four character types used in the Japanese writing system (along with hiragana, kanji and Roman alphabet), consisting of about 50 syllabic characters. It is used primarily to spell foreign loanwords (e.g., チョコレート [*chokoreeto*] — *chocolate*), and names (e.g., クリントン [*kurinton*] — *Clinton*). Therefore, katakana is a strong indicator that a Japanese word can be back-transliterated. However, katakana can also be used to spell scientific names of animals and plants (e.g., カモ [*kamo*] — *duck*), onomatopoeic expressions (e.g., バシャバシャ [*bashabasha*] — *splash*) and foreign origin words that are not transliterations (e.g., ホチキス [*hochikisu*] — *stapler*). These untransliterable cases constitute about 10% of the katakana words in our data.

We employ a discriminative substring decoder for machine transliteration. Following Sherif and Kondrak (2007b), the decoder operates on short source substrings, with each operation producing one or more target characters, as shown in Figure 1. However, where previous approaches employ generative modeling, we use structured perceptron training to discriminatively tune parameters according to 0-1 transliteration accuracy. This

[1] フリーメール is romanized as [*furiimeeru*]

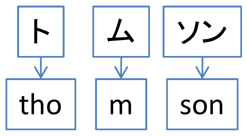

Figure 1: Example substring derivation

allows us to test novel methods for the use of target lexicons in discriminative character transduction, allowing our decoder to benefit from a list of known target words. Perhaps more significantly, our framework allows us to test two competing styles of features:

- sparse **indicators**, designed to capture the same channel and language modeling data collected by previous generative models, and
- components of existing generative models, used as real-valued features in a discriminatively weighted, **generative hybrid**.

Note that generative hybrids are the norm in SMT, where translation scores are provided by a discriminative combination of generative models (Och, 2003). Substring-based transliteration with a generative hybrid model is very similar to existing solutions for phrasal SMT (Koehn et al., 2003), operating on characters rather than words. Unlike out-of-the-box phrasal SMT solutions, our generative hybrid benefits from a target a lexicon. As we will show, this is the difference between a weak baseline and a strong competitor.

We demonstrate that despite recent successes in discriminative character transduction using indicator features (Jiampojamarn et al., 2008; Dreyer et al., 2008), our generative hybrid performs surprisingly well, producing our highest transliteration accuracies. Researchers frequently compare against a phrasal SMT baseline when evaluating a new transduction technique (Freitag and Khadivi, 2007; Dreyer et al., 2008); however, we are careful to vary only the features in our comparison. Confounding variables, such as alignment, decoder and training method, are held constant.

We also include a human evaluation of transliteration-augmented SMT output. Though human evaluations are too expensive to allow a comparison between transliteration systems, we are able to show that adding our transliterations to a production-level SMT engine results in a substantial improvement in translation quality.

2 Background

This work draws inspiration from previous work in transliteration, which we divide into similarity and transduction-based approaches. We also discuss recent successes in discriminative character transduction that have influenced this work.

2.1 Similarity-based transliteration

In similarity-based transliteration, a character-based, cross-lingual similarity metric is calculated (or bootstrapped) from known transliteration pairs. Given a source word s, its transliteration is the target word t most similar to s, where t is drawn from some pool of candidates. This approach may also be referred to as **transliteration discovery**.

Brill et al. (2001) describe a katakana-to-English approach with an EM-learned edit distance, which bootstraps from a small number of examples to learn transliteration pairs from query logs. Bilac and Tanaka (2005) harvest transliteration candidates from comparable bilingual corpora (conference abstracts in English and Japanese), and use distributional as well as phonetic similarity to choose among them. Sherif and Kondrak (2007a) also bootstrap a learned edit distance for Arabic named entities, with candidate pairs drawn from sentence or document-aligned parallel text. Klementiev and Roth (2006) bootstrap an SVM classifier trained to detect true transliteration-pairs. They draw candidates from comparable news text, using date information to provide further clues as to aligned named entities. Bergsma and Kondrak (2007) extend the classification approach with features derived from a character alignment. They train from bilingual dictionaries and word-aligned parallel text, selecting negative examples to target false-friends.

The work of Hermjakob et al. (2008) is particularly relevant to this paper, as they incorporate a similarity-based transliteration system into an Arabic-to-English SMT engine. They employ a hand-crafted cross-lingual similarity metric, and use capitalized n-grams from the Google n-gram corpus as candidates. With such a huge candidate list, a cross-lingual indexing scheme is designed for fast candidate look-up. Their work also addresses the question of when to transliterate (as opposed to translate), a realistic concern when deploying a transliteration component in SMT. This, however, is not of so much concern for katakana, as it is used primarily for loanwords.

2.2 Transduction-based transliteration

The approach presented in this paper is an instance of transduction-based transliteration, where the source word is transformed into a target word using a sequence of character-level operations. The parameters of the transduction process are learned from a collection of transliteration pairs. These systems do not require a list of candidates, but many incorporate a target lexicon, favoring target words that occur in the lexicon. This approach is also known as **transliteration generation**.

The majority of transliteration generation approaches are based on the noisy channel model, where a target t is generated according to $P(t|s) \propto P(s|t)P(t)$. This approach is typified by finite-state transliteration, where the various stages of the channel model are represented by finite state transducers and automata. Early systems employed a complex channel, passing through multiple phonetic representations (Knight and Graehl, 1998; Bilac and Tanaka, 2004), but later versions replaced characters directly (Al-Onaizan and Knight, 2002). Sherif and Kondrak (2007b) extend this approach with substring operations in the style of phrasal SMT, and show that doing so improves both accuracy as well as space and time efficiency. Note that it is possible to incorporate a target lexicon by making $P(t)$ a word unigram model with a character-based back-off.

Li et al. (2004) present an alternative to the noisy channel with their joint n-gram model, which calculates $P(s,t)$. This formulation allows operations to be conditioned on both source and target context. However, the inclusion of a candidate list is more difficult in this setting, as $P(t)$ is not given its own model.

Zelenko and Aone (2006) investigate a purely discriminative, alignment-free approach to transliteration generation. The target word is constructed one character at a time, with each new character triggering a suite of features, including indicators for near-by source and target characters, as well a generative target language model. Freitag and Khadivi (2007) propose a discriminative, latent edit distance for transliteration. In this case, training data need not be aligned in advance, but a latent alignment is produced during decoding. Again, the target word is constructed one character at a time, using edit operations that are scored according to source and target context features. Both approaches train using a structured perceptron, as we do here. However, these models represent a dramatic departure from the existing literature, while ours has clear analogs to the well-known noisy-channel paradigm, which allows for useful comparisons and insights into the advantages of discriminative training.

2.3 Discriminative character transduction

While our chosen application is transliteration, our decoder is influenced by recent successes in general-purpose discriminative transduction. Jiampojamarn et al. (2008) describe a discriminative letter-to-phoneme substring transducer, while Dreyer et al. (2008) describe a discriminative character transducer with a latent derivation structure for morphological transformations. Both models are extremely effective, but both rely exclusively on indicator features; they do not explore the use of knowledge-rich generative models. Our indicator system uses an extended version of the Jiampojamarn et al. (2008) feature set.

3 Methods

We adopt a discriminative substring decoder for our transliteration task. A structured perceptron (Collins, 2002) learns weights for our transliteration features, which are drawn from two broad classes: indicator and hybrid generative features.

3.1 Structured perceptron

The decoder's discriminative parameters are learned with structured perceptron training. Let a derivation d describe a substring operation sequence that transliterates a source word into a target word. Given an input training corpus of such derivations $D = d_1 \dots d_n$, a vector feature function on derivations $\vec{F}(d)$, and an initial weight vector $\vec{w}$, the perceptron performs two steps for each training example $d_i \in D$:

- Decode: $\bar{d} = \text{argmax}_{d \in D(\text{src}(d_i))} \left(\vec{w} \cdot \vec{F}(d)\right)$
- Update: $\vec{w} = \vec{w} + \vec{F}(d_i) - \vec{F}(\bar{d})$

where $D(\text{src}(d))$ enumerates all possible derivations with the same source side as d. To improve generalization, the final feature vector is the average of all vectors found during learning (Collins, 2002). Accuracy on the development set is used to select the number of times we pass through all $d_i \in D$.

Given the above framework, we require training derivations D, feature vectors $\vec{F}$, and a decoder to carry out the argmax over all d reachable from a particular source word. We describe each of these components in turn below.

3.2 Training derivations

Note that the above framework describes a max-derivation decoder trained on a corpus of gold-standard derivations, as opposed to a max-transliteration decoder trained directly on source-target pairs. By building the entire system on the derivation level, we side-step issues that can occur when perceptron training with hidden derivations (Liang et al., 2006), but we also introduce the need to transform our training source-target pairs into training derivations.

Training derivations can be learned unsupervised from source-target pairs using character alignment techniques. Previously, this has been done using an EM-learned edit distance (Ristad and Yianilos, 1998), or generalizations thereof (Brill and Moore, 2000; Jiampojamarn et al., 2007). We opt for an alternative alignment technique, similar to the word-aligner described by Zhang et al. (2008). This approach employs variational EM with sparse priors, along with hard length limits, to reduce the length of substrings operated upon. By doing so, we hope to learn only non-compositional transliteration units.

Our aligner produces only monotonic alignments, and does not allow either the source or target side of an operation to be empty. The same restrictions are imposed during decoding. In this way, each alignment found by variational EM is also an unambiguous derivation. We align our training corpus with a maximum substring length of three characters. The same derivations are used to train all of the transliteration systems tested in this paper.

3.3 Features

We employ two main types of features: indicators and hybrid generative models. Indicators detect binary events in a derivation, such as the presence of a particular operation. Hybrid generative features assign a real-valued probability to a derivation, based on statistics collected from training derivations. There are few generative features and each carries a substantial amount of information, while indicators are sparse and knowledge-poor.

We treat these two classes of features as distinct. We do so because researchers often use either one approach or the other.[2] Furthermore, it is not clear how to optimally employ training derivations when combining generative models and sparse indicators: generative models need large amounts of data to collect statistics and relatively little for perceptron training,[3] while sparse indicators require only a large perceptron training set.

We can further divide feature space according to the information required to calculate each feature. Both feature sets can be partitioned into the following subtypes:

- **Emission**: How accurate are the operations used by this derivation?
- **Transition**: Does the target string produced by this derivation look like a well-formed target character sequence?
- **Lexicon**: Does the target string contain known words from a target lexicon?

Indicator Features

Previous approaches to discriminative character transduction tend to employ only sparse indicators (Jiampojamarn et al., 2008; Dreyer et al., 2008). This is because sparsity is not a major concern in character-based domains, and sparse indicators are extremely flexible.

Our emission and transition indicator features follow Jiampojamarn et al. (2008). Emission indicators are centered around an operation, such as [ト $\rightarrow$ *tho*]. Minimally, an indicator exists for each operation. Many more **source context** features can be generated by conjoining an operation with source n-grams found within a fixed window of C characters to either side of the operation. These source context features have minimal computational cost, and they allow each operator to account for large, overlapping portions of the source, even when the substrings being operated upon are small. Meanwhile, transition indicators stand in for a character-based target language model. Indicators are built for each possible target n-gram, for $n = 1 \dots K$, allowing the perceptron to construct a discriminative back-off model. Development experiments lead us to select $C = 3$ and $K = 5$.

[2] Generative hybrids are often accompanied by a small number of unsparse indicators, such as operation count.

[3] Perceptron training on the same data used for model construction can lead to overconfidence in model quality. One can address this problem by using a large number of modeling-training folds (Collins et al., 2005), but we do not do so here.

Indicator lexicon features are novel to this work. Given access to a target lexicon with type frequencies, we opt to create features that indicate the frequencies of generated target words according to coarse bins. Experiments on our development set lead to the selection of 5 frequency bins: [< 2,000], [< 200], [< 20], [< 2], [< 1]. To keep the model linear, these features are cumulative; thus, generating a word with frequency 126 will result in both the $[< 2,000]$ and $[< 200]$ features firing. Note that a single transliteration can potentially generate multiple target words, and doing so can have a major impact on how often the lexicon features fire. Thus, we employ another feature that indicates the introduction of a new word. We expect these frequency indicators to be superior to a word-level unigram model, as they allow the designer to select notable frequencies. In particular, the bins we have selected do not give any advantage to extremely common words, as these are generally less likely to be transliterated.

Hybrid Generative Features

We begin with the three components of the generative noisy channel employed by Sherif and Kondrak (2007b). Their transliteration probability is:

$$P(t|s) \propto P_E(s|t) \cdot \max\left[P_T(t), P_L(t)\right] \qquad (1)$$

Inspired by the linear models used in SMT (Och, 2003), we can discriminatively weight the components of this generative model, producing:

$$w_E \log P_E(s|t) + w_T \log P_T(t) + w_L \log P_L(t)$$

with weights w learned by perceptron training.

These three models conveniently align with our three feature subtypes. Emission information is provided by $P_E(s|t)$, which is estimated by maximum likelihood on the operations observed in our training derivations. Including source context is difficult in such a model. To compensate for this, all systems using $P_E(s|t)$ also use composed operations, which are constructed from operation sequences observed in the training set. This removes the length limit on substring operations.[4] $P_T(t)$ provides transition information through a character language model, estimated on the target side of the training derivations. In our implementation, we employ a KN-smoothed 7-gram model (Kneser and Ney, 1995). Finally, $P_L(t)$ is a unigram target word model, estimated from the same type frequencies used to build our lexicon indicators.

Since we have adopted a linear model, we are no longer constrained by the original generative story. Therefore, we are free to incorporate other SMT-inspired features: $P_{E'}(t|s)$, target character count, and operation count.[5]

Feature summary

The indicator and hybrid-generative feature sets each provide a discriminative version of the noisy channel model. In the case of transition and lexicon features, both systems have access to the exact same information, but encode that information differently. The lexicon encoding is the most dramatic difference, with the indicators using a small number of frequency bins, and the generative unigram model providing a single, real-valued feature that is proportional to frequency.

In the case of their emission features, the two systems actually encode different information. Both have access to the same training derivations, but the indicator system provides source context through n-gram indicators, while the generative system does so using composed operations.

3.4 Decoder

Our decoder builds upon machine translation's monotone phrasal decoding (Zens and Ney, 2004), or equivalently, the sequence tagging algorithm used in semi-Markov CRFs (Sarawagi and Cohen, 2004). This dynamic programming (DP) decoder extends the Viterbi algorithm for HMMs by operating on one or more source characters (a substring) at each step. A DP block stores the best scoring solution for a particular prefix. Each block is subdivided into cells, which maintain the context necessary to calculate target-side features. We employ a beam, keeping only the 40 highest-scoring cells for each block, which speeds up inference at the expense of optimality. We found that the beam had no major effect on perceptron training, nor on the system's final accuracy.

Previously, target lexicons have been used primarily in finite-state transliteration, as they are easily encoded as finite-state-acceptors (Al-Onaizan and Knight, 2002; Sherif and Kondrak,

[4]Derivations built by our character aligner use operations on substrings of maximum length 3. To enable perceptron training with composed operations, once $P_E(s|t)$ has been estimated by counting composed operations in the initial alignments, we re-align our training examples with those composed operations to maximize $P_E(s|t)$, creating new training derivations.

[5]Character and operation counts also fit in the indicator system, but did not improve performance in development.

2007b). It is possible to extend the DP decoder to also use a target lexicon. By encoding the lexicon as a trie, and adding the trie index to the context tracked by the DP cells, we can provide access to frequency estimates for words and word prefixes. This has the side-effect of creating a new cell for each target prefix; however, in the character domain, this remains computationally tractable.

4 Data

4.1 Wikipedia training and test data

Our katakana-to-English training data is derived from bilingually-linked Wikipedia titles. Any Japanese Wikipedia article with an entirely katakana title and a linked English article results in training pair. This results in 60K transliteration pairs; we removed 2K pairs for development, and 2K for held-out testing.

The remaining 56K training pairs are quite noisy. As mentioned earlier, roughly 10% of our examples are simply not transliterable, but approximate Wikipedia title translations are an even more substantial source of noise. For example, コンピュータゲーム [*konpyuutageemu*] — *computer game* is aligned with the English article *Computer and video games*. We found it beneficial, in terms of both speed and accuracy, to do some coarse alignment-based pruning. After alignment, the operations used by all derivations are counted. Any operation that is used fewer than three times is eliminated, along with any derivation using that operation. The goal is to eliminate loose transliteration pairs from our data, where a word or initial is included in one language but not the other. This results in 40K training pairs. Despite the noise in the Wikipedia data, there are clear advantages in using it for training transliteration models: it is available for any language pair, it reflects recent trends and events, and the amount of data increases daily. As we will see below, the model trained on this data performs well on a test set from a very different domain.

All systems use development set accuracy to select their meta-parameters, such as the number of perceptron iterations, the size of the source-context window, and the n-gram length used in character language modeling. The hybrid generative system further splits the training set, using 38K derivations for the calculation of its emission and transition models, and 2K derivations for perceptron training its model weights.

4.2 Machine translation test data

In order to see how effective our transliterator is on out-of-domain test data, we also created test data from a log of translation requests to a web-based, Japanese-to-English translation service.[6] Out of 5,000 randomly selected translation requests, there are 312 cases where katakana source words are out-of-vocabulary for the MT system, and therefore remain untranslated. We created a reference translation (not necessarily a transliteration) for these katakana words by manually selecting the corresponding English word(s) in the sentence-level reference translation, which was produced independently from this experiment. This test set is quite divergent from the Wikipedia titles: only 17 (5.5%) of its katakana words are found in the Wikipedia training data, and six of these did not agree on the English translation.

4.3 English lexicon

Our English lexicon is derived from two overlapping data sources: the English gigaword corpus (LDC2003T05; GW) and the language model training data for our SMT system, which contains selections from Europarl, gigaword, and web-harvested text. Both are lowercased. We combine the unigram frequency counts from the two sources by taking the max when they overlap. The resulting lexicon has 5M types, 2.5M of which have frequency 1.

5 Experiments

In this section, we summarize development experiments, and then conduct a comparison on our two transliteration test sets. We report 0-1 accuracy: a transliteration is only correct if it exactly matches the reference. For the comparison experiments, we also report 10-best accuracy, where a system is correct if it includes the correct transliteration somewhere in its 10-best list.

5.1 Baselines

We compare our systems against a re-implementation of Sherif and Kondrak's (2007b) noisy-channel substring decoder. This uses the same P_E, P_T and P_L models as our hybrid generative system, but employs a two-pass decoding scheme to find the max transliteration according to Equation 1. It represents a purely generative solution using otherwise identical architecture.

[6]`http://www.microsofttranslator.com`

Since our hybrid generative system implements a model that is very similar to those used in phrasal SMT, we also compare against a state-of-the-art phrasal SMT system (Moore and Quirk, 2007). This system is trained by applying the standard SMT pipeline to our Wikipedia title pairs, treating characters as words, using a 7-gram character-level language model, and disabling re-ordering. Unfortunately, the decoder's architecture does not allow the use of a word-level unigram model, reducing the usefulness of this baseline. Instead, we include the target lexicon as a second character-level language model. This baseline indicates the level of performance one can expect by applying phrasal SMT straight out of the box.

Comparing the two baselines qualitatively, both use a combination of generative models inspired by the noisy channel. Sherif and Kondrak employ a word-level unigram model without discriminatively weighting the models, while the Phrasal SMT approach uses weights derived from max-BLEU training without word-level unigrams. The obvious question of what happens when one does both will be answered by our hybrid generative system.

5.2 Development experiments

Table 1 shows development set accuracy for a number of systems and feature types, along with the model size of the corresponding systems, where size is measured in terms of the number of non-zero discriminatively-trained parameters. The accuracy of the Sherif and Kondrak baseline is shown as *SK07*. Despite its lack of discriminative training, word-level unigrams allow the *SK07* baseline to outperform *Phrasal SMT*. In future experiments, we compare only against *SK07*.

The indicator system was tested using only operation indicators, with source context, transition and lexicon indicators added incrementally. All feature types have a substantial impact, with the lexicon providing the boost needed to surpass the baseline. Note that the inclusion of the five frequency bins is sufficient to decrease the overall feature count of the system by 600K, as much fewer mistakes are made during training.

Development of the hybrid generative system used the *SK07* baseline as a starting point. The result of combining its three components into a flat linear model, with all weights set to 1, is shown in Table 1 as *Linear SK07*. This violation of conditional independence assumptions results in a drop in accuracy. However, the $+$ *perceptron* line shows that setting the three weights with perceptron training results in a huge boost in accuracy, nearly matching our indicator system. Adding features inspired by SMT, such as $P_{E'}(t|s)$, eliminates the gap between the two.

Table 1: Development accuracy and model size

System		Acc.	Size
Baseline	Phrasal SMT	30.7	8
	SK07	33.5	–
Indicator	Operations only	3.6	6.8K
	+ source context	23.9	2.8M
	+ transition	28.6	3.1M
	+ lexicon	**44.2**	2.5M
	+ gen. lexicon	44.1	3.0M
Generative	Linear SK07	31.7	–
	+ perceptron	42.4	3
	+ SMT features	44.1	6
	+ ind. lexicon	**44.3**	12

5.3 Development discussion

Considering their differences, the two systems' proximity in score is quite surprising. Given the character domain's lack of sparsity, and the large amount of available training data, we had expected the hybrid generative system to behave only as a strong baseline; instead, it matched the performance of the indicator system. However, this is not unprecedented: discriminatively weighted generative models have been shown to outperform purely discriminative competitors in various NLP classification tasks (Raina et al., 2004; Toutanova, 2006), and remain the standard approach in statistical translation modeling (Och, 2003).

Examining the development results on an example-by-example basis, we see that the two systems make mostly the same mistakes: for 87% of examples, either both systems are right, or both are wrong. The remainder represents a (relatively small) opportunity to improve through system or feature combination: an oracle that perfectly selects between the two scores 50.6.

One opportunity for straight-forward combination is the target lexicon. Because lexicon frequencies are drawn from an independent word list, and not the transliteration training derivations, there is no reason why both systems cannot use both lexicon representations. Unfortunately, doing so has

Table 2: Test set comparisons

	Wikipedia		MT	
System	Acc.	Top 10	Acc.	Top 10
SK07	33.5	57.9	38.8	57.0
Generative	**43.0**	**65.6**	42.9	**58.3**
Indicator	42.5	63.5	**43.6**	57.7

little impact, as is shown in each system's final row in Table 1. Adding the word unigram model to the indicator system results in slightly lower performance, and a much larger model. Adding the frequency bins to the generative system does improve performance slightly, but attempts to completely replace the generative system's word unigram model with frequency bins resulted in a substantial drop in accuracy.[7]

5.4 Test set comparisons

Table 2 shows the accuracies of the systems selected during development on our testing data. On the held-out Wikipedia examples, the trends observed during development remain the same, with the generative system expanding its lead. Moving to 10-best accuracies changes little, except for slightly narrowing the gap between SK07 and the discriminative systems.

The second column of Table 2 compares the systems on our MT test set. As discussed earlier, this data is quite different from the Wikipedia training set, and as a result, the systems' differences are less pronounced. 1-best accuracy still shows the discriminative systems having a definite advantage, but at the 10-best level, those distinctions are muted.

Compared with the previous work on katakana-to-English transliteration, these accuracies do not look particularly high: both Knight and Graehl (1998) and Bilac and Tanaka (2004) report accuracies above 60% for 1-best transliteration. We should emphasize that this is due to the difficulty of our test data, and that we have tested against a baseline that has been shown to outperform Knight and Graehl (1998). The test data was not filtered for noise, leaving untransliterable cases and loose translations intact. The accuracies reported above are under-estimates of real performance: many transliterations not matching the reference may still be useful to a human reader, such as differences in inflection (e.g., レチノイド [*rechinoido*] — *retinoids*, transliterated as *retinoid*), and spacing (e.g. シエラレオネ [*shierareone*]— *Sierra Leone*, transliterated as *sierraleone*).

[7]Lexicon replacement experiment is not shown in Table 1.

6 Integration with machine translation

We used the transliterations from our indicator system to augment a Japanese-to-English MT system.[8] This treelet-based SMT system (Quirk et al., 2005) is trained on about 4.6M parallel sentence pairs from diverse sources including bilingual books, dictionaries and web publications. Our goal is to measure the impact of machine transliterations on end-to-end translation quality.

6.1 Evaluation method

We use the MT-log translation pairs described in Section 4.2 as a sentence-level translation test set. For each katakana word left untranslated by the baseline SMT engine, we generated 10-best transliteration candidates and added the katakana-English pairs to the SMT system's translation table. Perceptron scores were exponentiated, then normalized, to create probabilities, which were given to the SMT system as $P(source|target)$;[9] all other translation features were set to $\log 1$.

We translated the test set with and without the augmented translation table. 120 sentences were randomly selected from the cases where the translations output by the two SMT systems differed, and were submitted for two types of human evaluation. In the **absolute evaluation**, each SMT output was assigned a score between 1 and 4 (1 = completely useless; 4 = perfect translation); in the **relative evaluation**, the evaluators were presented with a pair of SMT outputs, with and without the transliteration table, and were asked to judge if they preferred one translation over the other. In both evaluation settings, the machine-translated sentences were evaluated by two native speakers of English who have no knowledge of Japanese, with access to a reference translation.

6.2 Results

The evaluation results show that our transliterator does improve the quality of SMT. The BLEU

[8]The human evaluation was carried out before we discovered the effectiveness of the hybrid generative system, but recall that the performance of the two is similar.

[9]The perceptron scores are more naturally interpreted as $P(target|source)$, but the opposite direction is generally the highest-weighted feature in the SMT system's linear model.

Table 3: Relative translation evaluation

	evaluator 1 preference			
eval2pref	+translit	equal	baseline	sum
+translit	**95**	0	2	97
equal	19	**1**	2	22
baseline	1	0	**0**	1
sum	115	1	4	120

score on the entire test set improved only slightly, from 21.8 to 22.0. However, in the absolute human evaluation, the transliteration table increased the average human judgement from 1.5 to 2 out of a maximum score of 4. Table 3 shows the results of the relative evaluation along with the judges' sentence-level agreement. In 95 out of 120 cases, both annotators agreed that the augmented table produced a better translation than the baseline.

One might expect that any replacement of katakana would improve the perception of MT quality. This is not necessarily the case: it can be more confusing to have a drastically incorrect transliteration, such as transliterating アップローダ [*appurooda*] — *uploader* incorrectly as *applaud*. Fortunately, Table 3 shows that we make very few of these sorts of mistakes: the baseline is preferred only rarely. Also note that, according the MT 10-best accuracies in Table 2, we would have expected to improve at most 60% of cases, however, the human judgements indicate that our actual rate of improvement is closer to 80%, which demonstrates that even an imperfect transliteration is often useful.

7 Conclusion

We have presented a discriminative substring decoder for transliteration. Our decoder is based on recent approaches for discriminative character transduction, extended to provide access to a target lexicon. We have presented a comparison of indicator and hybrid generative features in a controlled setting, demonstrating that generative models perform surprisingly well when discriminatively weighted. We have also shown our discriminative models to be superior to a state-of-the-art generative system. Finally, we have demonstrated that machine transliteration is immediately useful to end-to-end SMT.

As mentioned earlier, by focusing on katakana, we bypass the problem of deciding when to transliterate rather than translate; next, we plan to combine our models with a classifier that makes such a decision, allowing us to integrate transliteration into SMT for other language pairs.

References

Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in Arabic text. In *ACL Workshop on Comp. Approaches to Semitic Languages*.

Shane Bergsma and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *ACL*, pages 656–663, Prague, Czech Republic, June.

Slaven Bilac and Hozumi Tanaka. 2004. A hybrid back-transliteration system for Japanese. In *COLING*, pages 597–603, Geneva, Switzerland.

Slaven Bilac and Hozumi Tanaka. 2005. Extracting transliteration pairs from comparable corpora. In *Proceedings of the Annual Meeting of the Natural Language Processing Society*, Japan.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *ACL*, pages 286–293, Morristown, NJ.

Eric Brill, Gary Kacmarcik, and Chris Brockett. 2001. Automatically harvesting katakana-english term pairs from search engine query logs. In *Asia Federation of Natural Language Processing*.

Michael Collins, Brian Roark, and Murat Saraçlar. 2005. Discriminative syntactic language modeling for speech recognition. In *ACL*, pages 507–514, Ann Arbor, USA, June.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *EMNLP*, pages 1080–1089, Honolulu, Hawaii, October.

Dayne Freitag and Shahram Khadivi. 2007. A sequence alignment model based on the averaged perceptron. In *EMNLP*, pages 238–247, Prague, Czech Republic, June.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *ACL*, pages 389–397, Columbus, Ohio, June.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *HLT-NAACL*, pages 372–379, Rochester, New York, April.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *ACL*, pages 905–913, Columbus, Ohio, June.

Alexandre Klementiev and Dan Roth. 2006. Named entity transliteration and discovery from multilingual comparable corpora. In *HLT-NAACL*, pages 82–88, New York City, USA, June.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, pages 181–184.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT-NAACL*.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source-channel model for machine transliteration. In *ACL*, pages 159–166, Barcelona, Spain, July.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *COLING-ACL*, pages 761–768, Sydney, Australia, July.

Robert Moore and Chris Quirk. 2007. Faster beam-search decoding for phrasal statistical machine translation. In *MT Summit XI*.

Franz J. Och. 2003. Minimum error rate training for statistical machine translation. In *ACL*, pages 160–167.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *ACL*, pages 271–279, Ann Arbor, USA, June.

Rajat Raina, Yirong Shen, Andrew Y. Ng, and Andrew McCallum. 2004. Classification with hybrid generative/discriminative models. In *Advances in Neural Information Processing Systems 16*.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Sunita Sarawagi and William Cohen. 2004. Semi-markov conditional random fields for information extraction. In *ICML*.

Tarek Sherif and Grzegorz Kondrak. 2007a. Bootstrapping a stochastic transducer for Arabic-English transliteration extraction. In *ACL*, pages 864–871, Prague, Czech Republic, June.

Tarek Sherif and Grzegorz Kondrak. 2007b. Substring-based transliteration. In *ACL*, pages 944–951, Prague, Czech Republic, June.

Kristina Toutanova. 2006. Competitive generative models with structure learning for nlp classification tasks. In *EMNLP*, pages 576–584, Sydney, Australia, July.

Dmitry Zelenko and Chinatsu Aone. 2006. Discriminative methods for transliteration. In *EMNLP*, pages 612–617, Sydney, Australia, July.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT-NAACL*, pages 257–264, Boston, USA, May.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *ACL*, pages 97–105, Columbus, Ohio, June.