Lightly Supervised Transliteration for Machine Translation

Amit Kirschenbaum
Department of Computer Science
University of Haifa
31905 Haifa, Israel
akirsche@cs.haifa.ac.il

Shuly Wintner
Department of Computer Science
University of Haifa
31905 Haifa, Israel
shuly@cs.haifa.ac.il

Abstract

We present a Hebrew to English transliteration method in the context of a machine translation system. Our method uses machine learning to determine which terms are to be transliterated rather than translated. The training corpus for this purpose includes only positive examples, acquired semi-automatically. Our classifier reduces more than 38% of the errors made by a baseline method. The identified terms are then transliterated. We present an SMT-based transliteration model trained with a parallel corpus extracted from Wikipedia using a fairly simple method which requires minimal knowledge. The correct result is produced in more than 76% of the cases, and in 92% of the instances it is one of the top-5 results. We also demonstrate a small improvement in the performance of a Hebrew-to-English MT system that uses our transliteration module.

1 Introduction

Transliteration is the process of converting terms written in one language into their approximate spelling or phonetic equivalents in another language. Transliteration is defined for a pair of languages, a *source* language and a *target* language. The two languages may differ in their script systems and phonetic inventories. This paper addresses transliteration from Hebrew to English as part of a machine translation system.

Transliteration of terms from Hebrew into English is a hard task, for the most part because of the differences in the phonological and orthographic systems of the two languages. On the one hand, there are cases where a Hebrew letter can be pronounced in multiple ways. For example, Hebrew ב can be pronounced either as [b] or as [v]. On the other hand, two different Hebrew sounds can be mapped into the same English letter. For example, both ת and ט are in most cases mapped to [t]. A major difficulty stems from the fact that in the Hebrew orthography (like Arabic), words are represented as sequences of consonants where vowels are only partially and very inconsistently represented. Even letters that are considered as representing vowels may sometimes represent consonants, specifically ו [v]/[o]/[u] and י [y]/[i]. As a result, the mapping between Hebrew orthography and phonology is highly ambiguous.

Transliteration has acquired a growing interest recently, particularly in the field of Machine Translation (MT). It handles those terms where no translation would suffice or even exist. Failing to recognize such terms would result in poor performance of the translation system. In the context of an MT system, one has to first identify which terms should be transliterated rather than translated, and then produce a proper transliteration for these terms. We address both tasks in this work.

Identification of Terms To-be Transliterated (TTT) must not be confused with recognition of Named Entities (NE) (Hermjakob et al., 2008). On the one hand, many NEs should be translated rather than transliterated, for example:[1]

m$rd	hm$p@im
misrad	*hamishpatim*
ministry-of	the-sentences

'Ministry of Justice'

[1]To facilitate readability, examples are presented with interlinear gloss, including an ASCII representation of Hebrew orthography followed by a broad phonemic transcription, a word-for-word gloss in English where relevant, and the corresponding free text in English. The following table presents the ASCII encoding of Hebrew used in this paper:

א	ב	ג	ד	ה	ו	ז	ח	ט	י	כ
a	b	g	d	h	w	z	x	@	i	k
ל	מ	נ	ס	ע	פ	צ	ק	ר	ש	ת
l	m	n	s	&	p	c	q	r	$	t

Proceedings of the 12th Conference of the European Chapter of the ACL, pages 433–441,
Athens, Greece, 30 March – 3 April 2009. ©2009 Association for Computational Linguistics

him	htikwn
hayam	*hatichon*
the-sea	the-central
'the Mediterranean Sea'	

On the other hand, there are terms that are not NEs, such as borrowed words or culturally specific terms that are transliterated rather than translated, as shown by the following examples:

aqzis@ncializm	@lit
eqzistentzializm	*talit*
'Existentialism'	'Tallit'

As these examples show, transliteration cannot be considered the default strategy to handle NEs in MT and translation does not necessarily apply for all other cases.

Candidacy for either transliteration or translation is not necessarily determined by orthographic features. In contrast to English (and many other languages), proper names in Hebrew are not capitalized. As a result, the following homographs may be interpreted as either a proper name, a noun, or a verb:

alwn	alwn	alwn
alon	*alun*	*alon*
'oak'	'I will sleep'	'Alon' (name)

One usually distinguishes between two types of transliteration (Knight and Graehl, 1997): *Forward transliteration*, where an originally Hebrew term is to be transliterated to English; and *Backward transliteration*, in which a foreign term that has already been transliterated into Hebrew is to be recovered. Forward transliteration may result in several acceptable alternatives. This is mainly due to phonetic gaps between the languages and lack of standards for expressing Hebrew phonemes in English. For example, the Hebrew term *cdiq* may be transliterated as *Tzadik*, *Tsadik*, *Tsaddiq*, etc. On the other hand, backward transliteration is restrictive. There is usually only one acceptable way to express the transliterated term. So, for example, the name *wiliam* can be transliterated only to *William* and not, for example, to *Viliem*, even though the Hebrew character *w* may stand for the consonant [v] and the character *a* may be vowelized as [e].

We approach the task of transliteration in the context of Machine Translation in two phases. First, we describe a lightly-supervised classifier that can identify TTTs in the text (section 4). The identified terms are then transliterated (section 5) using a transliteration model based on Statistical Machine Translation (SMT). The two modules are combined and integrated in a Hebrew to English MT system (section 6).

The main contribution of this work is the actual transliteration module, which has already been integrated in a Hebrew to English MT system. The accuracy of the transliteration is comparable with state-of-the-art results for other language pairs, where much more training material is available. More generally, we believe that the method we describe here can be easily adapted to other language pairs, especially those for which few resources are available. Specifically, we did not have access to a significant parallel corpus, and most of the resources we used are readily available for many other languages.

2 Previous Work

In this section we sketch some related works, focusing on transliteration from Hebrew and Arabic, and on the context of machine translation.

Arbabi et al. (1994) present a hybrid algorithm for romanization of Arabic names using neural networks and a knowledge based system. The program applies vowelization rules, based on Arabic morphology and stemming from the knowledge base, to unvowelized names. This stage, termed the *broad* approach, exhaustively yields all valid vowelizations of the input. To solve this overgeneration, the *narrow* approach is then used. In this approach, the program uses a neural network to filter unreliable names, that is, names whose vowelizations are not in actual use. The vowelized names are converted into a standard phonetic representation which in turn is used to produce various spellings in languages which use Roman alphabet. The broad approach covers close to 80% of the names given to it, though with some extraneous vowelization. The narrow approach covers over 45% of the names presented to it with higher precision than the broad approach.

This approach requires a vast linguistic knowledge in order to create the knowledge base of vowelization rules. In addition, these rules are applicable only to names that adhere to the Arabic morphology.

Stalls and Knight (1998) propose a method for back transliteration of names that originate in English and occur in Arabic texts. The method uses a sequence of probabilistic models to convert names written in Arabic into the English script. First,

an Arabic name is passed through a phonemic model producing a network of possible English sound sequences, where the probability of each sound is location dependent. Next, phonetic sequences are transformed into English phrases. Finally, each possible result is scored according to a unigram word model. This method translates correctly about 32% of the tested names. Those not translated are frequently not foreign names.

This method uses a pronunciation dictionary and is therefore restricted to transliterating only words of known pronunciation. Both of the above methods perform only unidirectional transliteration, that is, either forward- or backward- transliteration, while our work handles both.

Al-Onaizan and Knight (2002) describe a system which combines a phonetic based model with a spelling model for transliteration. The spelling based model directly maps sequences of English letters into sequences of Arabic letters without the need of English pronunciation. The method uses a translation model based on IBM Model 1 (Brown et al., 1993), in which translation candidates of a phrase are generated by combining translations and transliterations of the phrase components, and matching the result against a large corpus. The system's overall accuracy is about 72% for top-1 results and 84% for top-20 results.

This method is restricted to transliterating NEs, and performs best for person names. As noted above, the TTT problem is not identical to the NER problem. In addition, the method requires a list of transliteration pairs from which the transliteration model could be learned.

Yoon et al. (2007) use phonetic distinctive features and phonology-based pseudo features to learn both language-specific and language-universal transliteration characteristics. Distinctive features are the characteristics that define the set of phonemic segments (consonants, vowels) in a given language. Pseudo features capture sound change patterns that involve the position in the syllable. Distinctive features and pseudo features are extracted from source- and target-language training data to train a linear classifier. The classifier computes compatibility scores between English source words and target-language words. When several target-language strings are transliteration candidates for a source word, the one with the highest score is selected as the transliteration. The method was evaluated using parallel corpora of English with each of four target languages. NEs were extracted from the English side and were compared with all the words in the target language to find proper transliterations. The baseline presented for the case of transliteration from English to Arabic achieves Mean Reciprocal Rank (MRR) of 0.66 and this method improves its results by 7%. This technique involves knowledge about phonological characteristics, such as elision of consonants based on their position in the word, which requires expert knowledge of the language. In addition, conversion of terms into a phonemic representation poses hurdles in representing short vowels in Arabic and will have similar behavior in Hebrew. Moreover, English to Arabic transliteration is easier than Arabic to English, because in the former, vowels should be deleted whereas in the latter they should be generated.

Matthews (2007) presents a model for transliteration from Arabic to English based on SMT. The parallel corpus from which the translation model is acquired contains approximately 2500 pairs, which are part of a bilingual person names corpus (LDC2005G02). This biases the model toward transliterating person names. The language model presented for that method consisted of 10K entries of names which is, again, not complete. This model also uses different settings for maximum phrase length in the translation model and different n-gram order for the language model. It achieves an accuracy of 43% when transliterating from Arabic to English.

Goldwasser and Roth (2008) introduce a discriminative method for identifying NE transliteration pairs in English-Hebrew. Given a word pair (w_s, w_t), where w_s is an English NE, the system determines whether w_t, a string in Hebrew, is its transliteration. The classification is based on pairwise features: sets of substrings are extracted from each of the words, and substrings from the two sets are then coupled to form the features. The accuracy of correctly identifying transliteration pairs in top-1 and top-5 was 52% and 88%, respectively. Whereas this approach *selects* most suitable transliteration out of a list of candidates, our approach *generates* a list of possible transliterations ranked by their accuracy.

Despite the importance of identifying TTTs, this task has only been addressed recently. Goldberg and Elhadad (2008) present a loosely supervised method for non contextual identification of

transliterated foreign words in Hebrew texts. The method is a Naive-Bayes classifier which learns from noisy data. Such data are acquired by over-generation of transliterations for a set of words in a foreign script, using mappings from the phonemic representation of words to the Hebrew script. Precision and recall obtained are 80% and 82%, respectively. However, although foreign words are indeed often TTTs, many originally Hebrew words should sometimes be transliterated. As explained in section 4, there are words in Hebrew that may be subject to either translation or transliteration, depending on the context. A non-contextual approach would not suffice for our task.

Hermjakob et al. (2008) describe a method for identifying NEs that should be transliterated in Arabic texts. The method first tries to find a matching English word for each Arabic word in a parallel corpus, and tag the Arabic words as either names or non-names based on a matching algorithm. This algorithm uses a scoring model which assigns manually-crafted costs to pairs of Arabic and English substrings, allowing for context restrictions. A number of language specific heuristics, such as considering only capitalized words as candidates and using lists of stop words, are used to enhance the algorithm's accuracy. The tagged Arabic corpus is then divided: One part is used to collect statistics about the distribution of name/non-name patterns among tokens, bigrams and trigrams. The rest of the tagged corpus is used for training using an averaged perceptron. The precision of the identification task is 92.1% and its recall is 95.9%. This work also presents a novel transliteration model, which is integrated into a machine translation system. Its accuracy, measured by the percentage of correctly translated names, is 89.7%.

Our work is very similar in its goals and the overall framework, but in contrast to Hermjakob et al. (2008) we use much less supervision, and in particular, we do not use a parallel corpus. We also do not use manually-crafted weights for (hundreds of) bilingual pairs of strings. More generally, our transliteration model is much more language-pair neutral.

3 Resources and Methodology

Our work consists of of two sub-tasks: Identifying TTTs and then transliterating them. Specifically, we use the following resources for this work: For the identification task we use a large un-annotated corpus of articles from Hebrew press and web-forums (Itai and Wintner, 2008) consisting of 16 million tokens. The corpus is POS-tagged (Bar-Haim et al., 2008). We bootstrap a training corpus for one-class SVM (section 4.2) using a list of rare Hebrew character n-grams (section 4.1) to generate a set of positive, high-precision examples for TTTs in the tagged corpus. POS tags for the positive examples and their surrounding tokens are used as features for the one-class SVM (section 4.2).

For the transliteration itself we use a list that maps Hebrew consonants to their English counterparts to extract a list of Hebrew-English translation pairs from Wikipedia (section 5.2). To learn the transliteration model we utilize Moses (section 5) which is also used for decoding. Decoding also relies on a target language model, which is trained by applying SRILM to Web 1T corpus (section 5.1).

Importantly, the resources we use for this work are readily available for a large number of languages and can be easily obtained. None of these require any special expertise in linguistics. Crucially, no parallel corpus was used.

4 What to transliterate

The task in this phase, then, is to determine for each token in a given text whether it should be translated or transliterated. We developed a set of guidelines to determine which words are to be transliterated. For example, person names are always transliterated, although many of them have homographs that can be translated. Foreign words, which retain the sound patterns of their original language with no semantic translation involved, are also (back-)transliterated. On the other hand, names of countries may be subject to translation or transliteration, as demonstrated in the following examples:

crpt	sprd	qwngw
tsarfat	*sfarad*	*kongo*
'France'	'Spain'	'Congo'

We use information obtained from POS tagging (Bar-Haim et al., 2008) to address the problem of identifying TTTs. Each token is assigned a POS and is additionally marked if it was not found in a lexicon (Itai et al., 2006). As a baseline, we tag for transliteration Out Of Vocabulary (OOV) tokens.

Our evaluation metric is tagging accuracy, that is, the percentage of correctly tagged tokens.

4.1 Rule-based tagging

Many of the TTTs do appear in the lexicon, though, and their number will grow with the availability of more language resources. As noted above, some TTTs can be identified based on their surface forms; these words are mainly loan words. For example, the word *brwdqsting* (broadcasting) contains several sequences of graphemes that are not frequent in Hebrew (e.g., *ng* in a word-final position).

We manually generated a list of such features to serve as tagging rules. To create this list we used a few dozens of character bigrams, about a dozen trigrams and a couple of unigrams and four-grams, that are highly unlikely to occur in words of Hebrew origin. Rules associate n-grams with scores and these scores are summed when applying the rules to tokens. A typical rule is of the form: if $\sigma_1\sigma_2$ are the final characters of w, add c to the score of w, where w is a word in Hebrew, σ_1 and σ_2 are Hebrew characters, and c is an positive integer. A word is tagged for transliteration if the sum of the scores associated with its substrings is higher than a predefined threshold.

We apply these rules to a large Hebrew corpus and create an initial set of instances of terms that, with high probability, should be be transliterated rather than translated. Of course, many TTTs, especially those whose surface forms are typical of Hebrew, will be missed when using this tagging technique. Our solution is to learn the *contexts* in which TTTs tend to occur, and contrast these contexts with those for translated terms. The underlying assumption is that the former contexts are *syntactically* determined, and are independent of the actual surface form of the term (and of whether or not it occurs in the lexicon). Since the result of the rule-based tagging is considered as examples of TTTs, this automatically-annotated corpus can be used to extract such contexts.

4.2 Training with one class classifier

The above process provides us with 40279 examples of TTTs out of a total of more than 16 million tokens. These examples, however, are only positive examples. In order to learn from the incomplete data we utilized a One Class Classifier. Classification problems generally involve two or more classes of objects. A function separating these classes is to be learned and used by the classifier. *One class* classification utilizes only target class objects to learn a function that distinguishes them from any other objects.

SVM (Support Vector Machine) (Vapnik, 1995) is a classification technique which finds a linear separating hyperplane with maximal margins between data instances of two classes. The separating hyperplane is found for a mapping of data instances into a higher dimension, using a kernel function. Schölkopf et al. (2000) introduce an adaptation of the SVM methodology to the problem of one-class classification. We used one-class SVM as implemented in LIBSVM (Chang and Lin, 2001). The features selected to represent each TTT were its POS and the POS of the token preceding it in the sentence. The kernel function which yielded the best results on this problem was a sigmoid with standard parameters.

4.3 Results

To evaluate the TTT identification model we created a gold standard, tagged according to the guidelines described above, by a single lexicographer. The testing corpus consists of 25 sentences from the same sources as the training corpus and contains 518 tokens, of which 98 are TTTs. We experimented with two different baselines: the naïve baseline always decides to translate; a slightly better baseline consults the lexicon, and tags as TTT any token that does not occur in the lexicon. We measure our performance in error rate reduction of tagging accuracy, compared with the latter baseline.

Our initial approach consisted of consulting only the decision of the one-class SVM. However, since there are TTTs that can be easily identified using features obtained from their surface form, our method also examines each token using surface-form features, as described in section 4.1. If a token has no surface features that identify it as a TTT, we take the decision of the one-class SVM. Table 1 presents different configurations we experimented with, and their results. The first two columns present the two baselines we used, as explained above. The third column (OCS) shows the results based only on decisions made by the One Class SVM. The penultimate column shows the results obtained by our method combining the SVM with surface-based features. The final column presents the Error Rate Reduction (ERR) achieved

when using our method, compared to the baseline of transliterating OOV words. As can be observed, our method increases classification accuracy: more than 38% of the errors over the baseline are reduced.

Naïve	Baseline	OCS	Our	ERR
79.9	84.23	88.04	90.26	38.24

Table 1: TTT identification results (% of the instances identified correctly)

The importance of the recognition process is demonstrated in the following example. The underlined phrase was recognized correctly by our method.

kbwdw	habwd	$l	bn	ari
kvodo	*heavud*	*shel*	*ben*	*ari*
His-honor	the-lost	of	Ben	Ari

'Ben Ari's lost honor '

Both the word *ben* and the word *ari* have literal meanings in Hebrew (*son* and *lion*, respectively), and their combination might be interpreted as a phrase since it is formed as a Hebrew noun construct. Recognizing them as transliteration candidates is crucial for improving the performance of MT systems.

5 How to transliterate

Once a token is classified as a TTT, it is sent to the transliteration module. Our approach handles the transliteration task as a case of phrase-based SMT, based on the noisy channel model. According to this model, when translating a string f in the source language into the target language, a string $\hat{e}$ is chosen out of all target language strings e if it has the maximal probability given f (Brown et al., 1993):

$$\begin{aligned}\hat{e} &= \arg\max_e \{Pr(e|f)\} \\ &= \arg\max_e \{Pr(f|e) \cdot Pr(e)\}\end{aligned}$$

where $Pr(f|e)$ is the translation model and $Pr(e)$ is the target language model. In phrase-based translation, f is divided into phrases $\bar{f}_1 \ldots \bar{f}_I$, and each source phrase $\bar{f}_i$ is translated into target phrase $\bar{e}_i$ according to a phrase translation model. Target phrases may then be reordered using a distortion model.

We use SMT for transliteration; this approach views transliteration pairs as aligned sentences and characters are viewed as words. In the case of phrase-based SMT, phrases are sequences of characters. We used Moses (Koehn et al., 2007), a phrase-based SMT toolkit, for training the translation model (and later for decoding). In order to extract phrases, bidirectional word level alignments are first created, both source to target and target to source. Alignments are merged heuristically if they are consistent, in order to extract phrases.

5.1 Target language model

We created an English target language model from unigrams of Web 1T (Brants and Franz, 2006). The unigrams are viewed as character n-grams to fit into the SMT system. We used SRILM (Stolcke, 2002) with a modified Kneser-Ney smoothing, to generate a language model of order 5.

5.2 Hebrew-English translation model

No parallel corpus of Hebrew-English transliteration pairs is available, and compiling one manually is time-consuming and labor-intensive. Instead, we extracted a parallel list of Hebrew and English terms from Wikipedia and automatically generated such a corpus. The terms are parallel titles of Wikipedia articles and thus can safely be assumed to denote the same entity. In many cases these titles are transliterations of one another. From this list we extracted transliteration pairs according to similarity of consonants in parallel English and Hebrew entries.

The similarity measure is based only on consonants since vowels are often not represented at all in Hebrew. We constructed a table relating Hebrew and English consonants, based on common knowledge patterns that relate sound to spelling in both languages. Sound patterns that are not part of the phoneme inventory of Hebrew but are nonetheless represented in Hebrew orthography were also included in the table. Every entry in the mapping table consists of a Hebrew letter and a possible Latin letter or letter sequences that might match it. A typical entry is the following:

$:SH|S|CH

such that SH, S or CH are possible candidates for matching the Hebrew letter $.

Both Hebrew and English titles in Wikipedia may be composed of several words. However, words composing the entries in each of the languages may be ordered differently. Therefore, every word in Hebrew is compared with every word

in English, assuming that titles are short enough. The example in Table 2 presents an aligned pair of multi-lingual Wikipedia entries with high similarity of consonants. This is therefore considered as a transliteration pair. In contrast, the title *empty set* which is translated to *hqbwch hriqh* shows a low similarity of consonants. This pair is not selected for the training corpus.

g	r	a		t	e	f	u	l		d	e	a	d
g	r	i	i	@		p	w	l		d		d

Table 2: Titles of Wikipedia entries

Out of 41914 Hebrew and English terms retrieved from Wikipedia, more than 20000 were determined as transliteration pairs. Out of this set, 500 were randomly chosen to serve as a test set, 500 others were chosen to serve as a development set, and the rest are the training set. Minimum error rate training was done on the development set to optimize translation performance obtained by the training phase.[2] For decoding, we prohibited Moses form performing character reordering (distortion). While reordering may be needed for translation, we want to ensure the monotone nature of transliteration.

5.3 Results

We applied Moses to the test set to get a list of top-n transliteration options for each entry in the set. The results obtained by Moses were further re-ranked to take into account their frequency as reflected in the unigrams of Web 1T (Brants and Franz, 2006). The re-ranking method first normalizes the scores of Moses' results to the range of $[0, 1]$. The respective frequencies of these results in Web1T corpus are also normalized to this range. The score s of each transliteration option is a linear combination of these two elements: $s = \alpha s_M + (1 - \alpha)s_W$, where s_M is the normalized score obtained for the transliteration option by Moses, and s_W is its normalized frequency. α is empirically set to 0.75. Table 3 summarizes the proportion of the terms transliterated correctly across top-n results as achieved by Moses, and their improvement after re-ranking.

We further experimented with two methods for reducing the list of transliteration options to the most prominent ones by taking a *variable* number of candidates rather than a fixed number. This is important for limiting the search space of MT systems. The first method (var1) measures the ratio between the scores of each two consecutive options and generates the option that scored lower only if this ratio exceeds a predefined threshold. We found that the best setting for the threshold is 0.75, resulting in an accuracy of 88.6% and an average of 2.32 results per token. Our second method (var2) views the score as a probability mass, and generates all the results whose combined probabilities are at most p. We found that the best value for p is 0.5, resulting in an accuracy of 87.4% and 1.92 results per token on average. Both methods outperform the top-2 accuracy.

[2]We used `moses-mert.pl` in the Moses package.

Results	Top-1	Top-2	Top-5	Top-10
Moses	68.4	81.6	90.2	93.6
Re-ranked	76.6	86.6	92.6	93.6

Table 3: Transliteration results (% of the instances transliterated correctly)

Table 4 presents a few examples from the test set that were correctly transliterated by our method. Some incorrect transliterations are demonstrated in Table 5.

Source	Transliteration
np$	nefesh
hlmsbrgr	hellmesberger
smb@iwn	sambation
hiprbwlh	hyperbola
$prd	shepard
ba$h	bachet
xt$pswt	hatshepsut
brgnch	berganza
ali$r	elissar
g'wbani	giovanni

Table 4: Transliteration examples generated correctly from the test set

6 Integration with machine translation

We have integrated our system as a module in a Machine Translation system, based on Lavie et al. (2004a). The system consults the TTT classifier described in section 4 for each token, before translating it. If the classifier determines that the token should be transliterated, then the transliteration procedure described in section 5 is applied to the token to produce the transliteration results.

Source	Transliteration	Target
rbindrnt	rbindrant	rabindranath
aswirh	asuira	essaouira
kmpi@	champit	chamaephyte
bwdlr	bodler	baudelaire
lwrh	laura	lorre
hwlis	ollies	hollies
wnwm	onom	venom

Table 5: Incorrect transliteration examples

We provide an external evaluation in the form of BLEU (Papineni et al., 2001) and Meteor (Lavie et al., 2004b) scores for SMT with and without the transliteration module.

When integrating our method in the MT system we use the best transliteration options as obtained when using the re-ranking procedure described in section 5.3. The translation results for all conditions are presented in Table 6, compared to the basic MT system where no transliteration takes place. Using the transliteration module yields a statistically significant improvement in METEOR scores ($p < 0.05$). METEOR scores are most relevant since they reflect improvement in recall. The MT system cannot yet take into consideration the weights of the transliteration options. Translation results are expected to improve once these weights are taken into account.

System	BLEU	METEOR
Base	9.35	35.33127
Top-1	9.85	38.37584
Top-10	9.18	37.95336
var1	8.72	37.28186
var2	8.71	37.11948

Table 6: Integration of transliteration module in MT system

7 Conclusions

We presented a new method for transliteration in the context of Machine Translation. This method identifies, for a given text, tokens that should be transliterated rather than translated, and applies a transliteration procedure to the identified words. The method uses only positive examples for learning which words to transliterate and achieves over 38% error rate reduction when compared to the baseline. In contrast to previous studies this method does not use any parallel corpora for learning the features which define the transliterated terms. The simple transliteration scheme is accurate and requires minimal resources which are general and easy to obtain. The correct transliteration is generated in more than 76% of the cases, and in 92% of the instances it is one of the top-5 results.

We believe that some simple extensions could further improve the accuracy of the transliteration module, and these are the focus of current and future research. First, we would like to use available gazetteers, such as lists of place and person names available from the US census bureau, `http://world-gazetteer.com/` or `http://geonames.org`. Then, we consider utilizing the bigram and trigram parts of Web 1T (Brants and Franz, 2006), to improve the TTT identifier with respect to identifying multi-token expressions which should be transliterated. In addition, we would like to take into account the weights of the different transliteration options when deciding which to select in the translation. Finally, we are interested in applying this module to different language pairs, especially ones with limited resources.

Acknowledgments

We wish to thank Gennadi Lembersky for his help in integrating our work into the MT system, as well as to Erik Peterson and Alon Lavie for providing the code for extracting bilingual article titles from Wikipedia. We thank Google Inc. and the LDC for making the Web 1T corpus available to us. Dan Roth provided good advice in early stages of this work. This research was supported by THE ISRAEL SCIENCE FOUNDATION (grant No. 137/06); by the Israel Internet Association; by the Knowledge Center for Processing Hebrew; and by the Caesarea Rothschild Institute for Interdisciplinary Application of Computer Science at the University of Haifa.

References

Yaser Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 400–408, Morristown, NJ, USA. Association for Computational Linguistics.

Mansur Arbabi, Scott M. Fischthal, Vincent C. Cheng,

and Elizabeth Bart. 1994. Algorithms for arabic name transliteration. *IBM Journal of Research and Development*, 38(2):183–194.

Roy Bar-Haim, Khalil Sima'an, and Yoad Winter. 2008. Part-of-speech tagging of Modern Hebrew text. *Natural Language Engineering*, 14(2):223–251.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram version 1.1. Technical report, Google Reseach.

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

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Yoav Goldberg and Michael Elhadad. 2008. Identification of transliterated foreign words in hebrew script. In *CICLing*, pages 466–477.

Dan Goldwasser and Dan Roth. 2008. Active sample selection for named entity transliteration. In *Proceedings of ACL-08: HLT, Short Papers*, pages 53–56, Columbus, Ohio, June. Association for Computational Linguistics.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *Proceedings of ACL-08: HLT*, pages 389–397, Columbus, Ohio, June. Association for Computational Linguistics.

Alon Itai and Shuly Wintner. 2008. Language resources for Hebrew. *Language Resources and Evaluation*, 42(1):75–98, March.

Alon Itai, Shuly Wintner, and Shlomo Yona. 2006. A computational lexicon of contemporary hebrew. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-2006)*, pages 19–22, Genoa, Italy.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Madrid, Spain. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Alon Lavie, Erik Peterson, Katharina Probst, Shuly Wintner, and Yaniv Eytani. 2004a. Rapid prototyping of a transfer-based Hebrew-to-English machine translation system. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 1–10, Baltimore, MD, October.

Alon Lavie, Kenji Sagae, and Shyamsundar Jayaraman. 2004b. The significance of recall in automatic metrics for mt evaluation. In Robert E. Frederking and Kathryn Taylor, editors, *AMTA*, volume 3265 of *Lecture Notes in Computer Science*, pages 134–143. Springer.

David Matthews. 2007. Machine transliteration of proper names. Master's thesis, School of Informatics, University of Edinburgh.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Bernhard Schölkopf, Alex J. Smola, Robert Williamson, and Peter Bartlett. 2000. New support vector algorithms. *Neural Computation*, 12:1207–1245.

Bonnie Glover Stalls and Kevin Knight. 1998. Translating names and technical terms in Arabic text. In *Proceedings of the COLING/ACL Workshop on Computational Approaches to Semitic Languages*, pages 34–41.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing (ICSLP 2002)*, pages 901–904.

Vladimir N. Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Su-Youn Yoon, Kyoung-Young Kim, and Richard Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 112–119, Prague, Czech Republic, June. Association for Computational Linguistics.