Lightly Supervised Transliteration for Machine Translation

Amit Kirschenbaum

March, 2009

Contents

Abstract

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 orthographic systems and phonetic inventories. In the context of a Machine Translation system, one has to first identify which terms should be transliterated rather than translated, and then produce a proper transliteration for these terms.

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 from a corpus of articles from Hebrew press and web-forums. Our classifier reduces more than 38% of the errors made by a baseline method. The identified terms are then transliterated based on a Statistical Machine Translation technique. The transliteration model was trained with a parallel corpus extracted from Wikipedia using a fairly simple method which requires minimal linguistic knowledge. The correct result is produced as the Top-1 result in more than 76% of the cases, and in 95% of the instances it is one of the Top-5 results. We also demonstrate an improvement in the performance of a Hebrew-to-English MT system that uses our transliteration module.

List of Tables

Chapter 1

Introduction

1.1 Machine Translation

Contemporary approaches to machine translation follow for the most part the paradigm of Statistical Machine Translation (SMT) (Brown et al., 1993). SMT is based on ideas used in Information Theory and in particular Shannon's noisy-channel model. The purpose of this model is to identify a message which is transmitted through a communication channel and is hence prone to errors due to the channel's quality. Adapting this model, SMT systems treat the source text as a version of the target text which has gone through a noisy channel. The translation task is therefore reduced to recovering the original message.

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:

$$\hat{e} = \arg\max_{e} \{Pr(e|f)\} = \arg\max_{e} \{Pr(f|e) \cdot Pr(e)\}$$

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 $\bar{f}_1 \dots \bar{f}_I$, and each source phrase $\bar{f}_i$ is translated into a target phrase $\bar{e}_i$ according to a phrase translation model. Target phrases may then be reordered using a distortion model.

In order for the above formula to hold, SMT systems require terms in the source language to have translation equivalents in the target language. However, language is subject to changes, as new expressions are introduced in speech and media frequently. As a result, lexicons might be lacking, in terms of adequately representing speakers' word inventories. This results in zero probability when trying to find the translation of such terms. Such words, which are in actual use but do not appear in published vocabularies, are called *Out Of Vocabulary* (OOV) terms, and are often handled by *transliteration* rather than translation.

1.2 Transliteration

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 writing systems and phonetic inventories. This work addresses transliteration from Hebrew to English in the context of a machine translation system.

Transliteration has acquired a growing interest recently, in particular in the field of machine translation. 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 MT systems, 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). Named Entities, such as names of persons, locations or organizations, are often transliterated. However, this cannot be a general method for processing them. On the one hand, many NEs should be translated rather than transliterated, for example:[1]

[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 document:

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

משרד המשפטים		הים התיכון		צרפת
m$rd	hm$p@im	him	htikwn	crpt
[misrad]	[hamiʃpatim]	[hayam]	[hatiχon]	[tsarfat]
ministry-of	the-sentences	the-sea	the-central	
‘Ministry of Justice’		‘the Mediterranean Sea’		‘France’

On the other hand, there are terms that are not NEs that should be transliterated rather than translated. These include both borrowed words (or borrowings) and culturally specific terms.

Borrowings are words that are adopted by speakers of one language from another language, with which they are in contact, in cases where no proper equivalent would suffice (Nir, 1979). The vocabulary of Modern Hebrew is enriched through a variety of terms borrowed mainly from European languages and American English, as shown by the following example:

אקזיסטנציאליזם

aqzis@ncializm

[ɛkzistɛntsializəm]

‘Existentialism’

This example illustrates what is treated here as the default case of borrowed words, that is, a case in which the borrowed word has no equivalent term in Hebrew. This entails that MT systems should approach such cases not as candidates for translation, but inherently as strings that should be transliterated.

Borrowings in general can be linguistically distinguished from native words. Particularly relevant for the present work is the fact that they reflect both phonological and morphological deviations from the canonic structure of native words (Schwarzwald, 2002). For example, borrowed words in Hebrew are marked by the use of non-native consonants as well as by specific prefixes and suffixes used for inflection and derivation (see, also, Bolozky (1978)).Consider the following examples:

ג'יפ	אנרכיסט
g'ip	anrkis@
[ʤip]	[anarχist]
'Jeep'	'Anarchist'

The word **ג'יפ** in this example both contains the non-Hebrew consonant [ʤ] and displays a distinct distribution of consonants which is atypical of Hebrew – the pronunciation of [p] in word-final position. The word **אנרכיסט** illustrates that not only bases but also inflectional morphemes can be borrowed into Hebrew and are perceived by native Hebrew speakers as foreign strings that require transliteration. Even foreign derivational suffixes, as **יה-** ,in the example below, only *resemble* Hebrew suffixes (Rosner, 2007), and cannot be considered candidates for translation, since no transfer of meaning is involved.

טרנסליטרציה

trnsli@rcih

[transliteratsia]

'Transliteration'

Such features will assist us in the task of identifying TTTs (section 4). There are other properties that may distinguish foreign words from native words, such as location of word stress, and consonant-clusters in the word. These features, however, cannot be revealed from Hebrew texts in general, since the standard Hebrew orthography does not indicate stress, and leaves many of the vowels implicit.

In contrast, culturally specific terms cannot be identified based on surface features as in the case of borrowings. Consider the following example:

טלית

@lit

[talit]

'Tallit'

The word **טלית** is of Hebrew origin, and cannot be distinguished from other native words

based on orthography alone. As will be suggested below (section 4), words of this type are assumed to appear in particular *syntactic contexts* that can be used to identify them.

Tranlisteration 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 more than one way. For example, Hebrew ב can be pronounced either as [b] or as [v], פ can be pronounced either as [p] or as [f] and ש can be pronounced as either [ʃ] or as [s]. 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.

The following examples present the ASCII representation of Hebrew letters aligned with the phonetic form. Note that words that have seemingly very similar surface forms may have distinctively different pronunciations.

ברק	ברך	דור	דור
b r q	b r k	d w r	d w r
[b a r a k]	[b e r e χ]	[d a v a r]	[d o r]
'lightning'	'knee'	'postman'	'generation'

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 more 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].

1.3 Research Objectives

The objective of this work is to improve the quality of Hebrew to English MT by implementing a transliteration module for this pair of languages. We divide this task into two sub-tasks, which we address separately: First, we classify which words in the text are to be transliterated rather than translated, using machine learning techniques (Chapter 4). Then (Chapter 5), we describe a transliteration model based on Statistical Machine Translation (SMT). Each module is evaluated separately, and both perform as well as (or better than) state-of-the-art methods for similar language pairs. The two modules are then combined and integrated in a Hebrew to English MT system (Chapter 6), and we use the MT system as an external evaluation framework for the combined method. The performance of the MT system is shown to improve when the transliteration module is added.

Chapter 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 over-generation, 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 vast linguistic knowledge in order to create the knowledge base of vowelization rules. In addition, these rules are applicable only for names that adhere to the Arabic morphology .

While Arbabi et al. (1994) focuse on transliterating Arabic names, Stalls and Knight (1998) propose a method for *back* transliteration of names that originate in English and appear in Arabic texts. Their method is based on a phonemic model, and 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 that produces 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 transliteration in the context of machine translation should handle both.

Al-Onaizan and Knight (2002) describe a system which combines a phonetic-based model with a spelling-based model for transliteration. This method is restricted to transliterating NEs, and performs best for person names. The spelling based-model directly maps sequences of English letters into sequences of Arabic letters without the need of English pronunciation, as in a phonemic model. Transforming into a phonetic representation requires an additional phase which introduces additional imprecision due to the probabilistic model. Therefore, a transliteration method which does not include such a phase may have an improved accuracy. 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 was tested using a blind test set, consisting of 20 newspaper articles which have translated by professionals from Arabic to English. The system's overall accuracy is about 72% for Top-1 results and 84% for the Top-20 results.

Studies that use more sophisticated conversion rules are mostly suited for source languages which encode phonetic information explicitly. For example, Oh and Choi (2002) introduce a correspondence-based model for transliteration of English to Korean. This model uses contexts of source language graphemes and their corresponding source language phonemes when producing target language graphemes. Oh and Choi (2002) present the notion of

pronunciation-units: a chunk of graphemes that can be mapped to a phoneme. First, English pronunciation units are aligned with their corresponding English phonemes. Then, context sensitive rewrite rules, incorporating this correspondence, are used to generate Korean graphemes. Evaluation was performed using two evaluation metrics: word accuracy, which measures the ratio of words generated correctly to all generated words, and character accuracy, which measures the edit distance between the generated word and the reference. This method achieves character accuracy of 93.49% and word accuracy of 67.83% for words appearing in a pronunciation dictioanry. For words which are not registerd in the dictionary, the respective accuracies are 91.47% and 52.40%.

Oh et al. (2006) provide a comprehensive comparison between different models of transliteration for English-Korean and English-Japanese: grapheme-based, phoneme-based, hybrid (combining both) and correspondence-based. They offer convincing arguments for preferring hybrid and correspondence-based models over the other models. However, in Arabic and Hebrew vowels are often not represented in the script. Phonetic representation of source-language strings in these languages cannot be implied by their surface forms, as demonstrated in the introduction, and therefore a different approach is required.

Alignment methods have been successfully applied to languages such as Arabic and Persian. AbdulJaleel and Larkey (2003) present a technique for English to Arabic transliteration using a list of English proper nouns and their Arabic translations. The method consists of two alignment stages; the first stage aligns characters of the word pairs with Arabic as the source language and English as target language. Cases where a sequence of English characters is aligned to a single Arabic characters are tallied to find the most frequent among these character sequences. The second stage consists of aligning these word pairs, where English is the source language and Arabic is the target language, such that the English character-sequences mentioned above are grouped to act as a single character. Both stages use GIZA++ (Och and Ney, 2003), a tool which was originally designed for word alignment in parallel corpora. The transliteration model is then based on counting alignments generated by GIZA++, to calculate conditional probabilities. A letter-bigram model of Arabic is used as a language

model. This technique achieves accuracy of 69.3% for the Top-1 results and 88.3% for the Top-20 results.

Applying a word alignment tool to align characters is indeed good. Using such a tool assumes no prior knowledge related to a specific language. However, the task discussed here is simpler than full transliteration. First, it handles only forward transliteration, where multiple results are accepted. In addition, observe that English-to-Arabic (or English-to-Hebrew) transliteration is easier than Arabic-to-English (or Hebrew-to-English), because in the former vowels should be deleted whereas in the latter they should be generated. When Hebrew is the source language, sequences of consonants are to be vowelized, and a vast range of options exists for vowelization, but usually only one is valid.

Karimi et al. (2007) extend the notion of alignment to *meta*-characters. They propose an algorithm for transliterating from English to Persian, using a two step alignment method. The first step represents parallel words in a bilingual corpus as *reduced consonant-vowel* sequences, that is, sequences where each meta-character denotes either a cluster of consonants (C) or a sequence of vowels (V). The resulting parallel meta-character sequences are compared, and if they are equal (e.g., a CVC sequence is generated for both English and Persian due to consonant-vowel reduction), the aligned consonant clusters and vowel sequences are added to the alignments repository. As for transliteration from Persian to English, a back-transliteration method is used. It utilizes *reverse segmentation*, where reduced consonant-vowel sequences that were generated during the process of English-to-Persian alignment are used for the segmentation of words in Persian that are typically lacking in the representation of vowels. Aligned data are then used to derive probabilistic transformation rules. To generate transliterations, a source word is first segmented following the process used for training. Then, target words are generated according to the conditional probabilities of each segment's transliteration, and are ranked based on their probabilities. For English-Persian transliteration, this method achieves mean word accuracy of 72.2% in Top-1 and 93.5% in Top-10 results. Persian-English transliteration achieves mean word accuracy of 48.2% in Top-1 results and 75.7%, when using the proposed alignment method including the reverse-

segmentation technique.

This method requires language knowledge for segmenting words to consonant-vowel sequences. The collapsed-vowel model is based on particular characteristics of Persian such as the tendency of Persian speakers to change diphthongs to monophtongs in loan words. It also handles issues that are specific to Persian-English transliteration, such as epenthesis and elision. The low accuracy of Persian to English transliteration stresses the complexity of transliterating from languages which do not encode vowels explicitly in their script.

Matthews (2007) presents a model for transliteration from Arabic to English based on SMT. The translation model is acquired from a parallel corpus which includes approximately 2500 pairs, a part of a bilingual person name corpus (Intel, 2005). The language model used in this work consists of 10K entries of names. The system achieves accuracy of 43% when transliterating from Arabic to English. Like Matthews (2007), we rely on phrase-based SMT for transliteration (see Chapter 5). Matthews (2007) relies on domain specific resources, biasing the model towards transliterating person names. In contrast, our method is general. We also address the question of which terms to transliterate. Finally, our choice of resources allows for much more accurate transliteration.

Huang (2005) proposes a framework for cluster-specific transliteration of names, and applies it to Chinese-to-English transliteration. Starting with a list of transliteration pairs of names, whose original language is labeled, language and transliteration models are constructed for each original language. Models from different original languages are recursively merged to form clusters, and transliteration and language models are then trained for each cluster. Given a name in the source language, it is first classified into the most likely cluster. Then, it is transliterated according to the corresponding models of this cluster using SMT technique. The overall accuracy of this method is 56% for Top-1 results and 62.6% for Top-5 results.

This method requires a list of names labeled with their original languages. Labeling such a list manually is labor intensive, and may be ambiguous in Hebrew, where homographs may correspond to names from different origins. In addition, this method was trained with a list

of names and is therefore biased to transliterating person names,

Several studies employ a *discriminative*, rather than generative, approach to the transliteration task. Methods using a discriminative approach determine for two strings occurring in a parallel corpus (w_s, w_t) whether w_t, a target language string, is a transliteration of w_s, a source language string.

Yoon et al. (2007) suggest to solve the transliteration problem by constructing a classifier that determines whether a word in the target language is a transliteration of a name in the source language. The candidates for the classifier are NEs extracted from the source language and the set of all words in the target language, and terms in both sets are converted to a phonemic representation. The classifier is based on the Winnow algorithm (Littlestone, 1987) and use features such as place and manner of articulation, vowel length, as well as patterns of insertion/deletion/substitution of such features, based on the position in the syllable. The baseline for this method is described in Tao et al. (2006) and uses purely linguistic knowledge. This baseline achieves Mean Reciprocal Rank (MRR) (see Chapter 3 for a definition) of 0.66 for the case of transliteration from English to Arabic. The method presented in Yoon et al. (2007) improves this result 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. Conversion of terms into a phonemic representation poses hurdles in representing short vowels in Arabic and will have similar behavior in Hebrew. As mentioned, English to Arabic transliteration is easier than Arabic to English. In addition, transliteration of terms that do not appear in the target-language side of a parallel corpus cannot be evaluated properly by this method. The latter is true also for names that do not have a standard transliteration, and for which multiple transliterations are acceptable, which is often the case in forward transliteration.

Very few studies concern Hebrew to English transliteration. Goldwasser and Roth (2008a) propose a discriminative approach to transliteration, and demonstrate it for English-Hebrew (as well as English-Russian). In this setup, a bilingual, comparable corpus is given, in which

named entities are annotated on the source side. In addition, a reference set consisting of 50,000 NEs in the source language (English) is given (this set was automatically extracted from Wikipedia). The method further uses an "oracle", which is capable of matching a named entity in the source language with its counterpart in the target; the oracle was acquired from Wikipedia, by identifying the topic of two topic-aligned documents in the Biography section. This implementation relies on the fact that topics of biography articles are person names, and are hence transliterations of one another. The core of the system is a process that identifies a small yet informative subset of the NEs in the reference set, along with their transliterations as provided by the oracle. This set is used to train a linear classifier (SNoW (Carlson et al., 1999), with perceptron as the update rule); the features for the classifier are pairs of character substrings, taken from the source and target words. The classifier is evaluated on 300 English-Hebrew and English-Russian pairs; each English term is paired with all the possible terms in the target language, and the percentage of the original 300 pairs identified correctly is reported. In the case of Hebrew, 52% of the pairs were identified correctly. When the top-five candidates are considered, the correct transliteration is identified in 88% of the cases.

In a subsequent work, Goldwasser and Roth (2008b) acknowledge the difficulties of the discriminative model, and attribute them to the inability of pairwise character n-gram features to capture contextual constraints that affect the transliteration. They therefore suggest to view the transliteration task as a constrained optimization problem with more global constraints. The features for the classification task remain pairs of character n-grams, each associated with a weight. This representation is refined by solving a constrained optimization problem that maximizes the score of a candidate pair subject to a set of constraints that account for interdependencies among features. This method is evaluated in the same way, but the measure here is MRR (see Chapter 3). The best result is MRR of 0.894.

The main drawback of the discriminative approach is that it can be used to *identify* transliteration terms, but not to *generate* them. That is, the correct transliteration cannot be generated by the system if it is not included in the bilingual corpus. In addition, the two

methods use parallel corpora of person names only. This may bias the transliteration model, and patterns which are infrequent in person names but nevertheless occur in other TTTs, would therefore be under-represented.

Despite the importance of identifying TTTs, this task has been addressed only recently. Goldberg and Elhadad (2008) present a loosely supervised method for identification of words that are phonologically close to English pronunciation, in the Hebrew script. Their approach focuses on words in isolation, ignoring their context, and they assume that these words can be distinguished from Hebrew ones based on character n-grams. The method is a Naive-Bayes classifier which learns from noisy data. The native language model is estimated using a corpus consisting mainly of words of Hebrew origin. That is, they categorize words as native ones according to the degree to which they conform with words occurring in texts in Modern Hebrew. The foreign language model is acquired by over-generation of transliterations. This is accomplished using mappings from the phonemic representation of words in the foreign script to the Hebrew script. Precision and recall for the task of identifying such words are 80% and 82%, respectively.

This system does not identify TTTs in the broad sense of the word, as defined here (Chapter 4). Rather, they identify only those strings whose phonemic representation in Hebrew correspond exactly with that in English. For example, the word **טרנדי** (@rndi) is identified by this system since it conforms with the orthographic representation of the word in English - *trendy*, but the word **אלכוהולי** (alkwhwli) is not identified by this system, since the word is written differently in English - *alcoholic*. This means that foreign words that do not have the same surface form as Hebrew words are not identified. Words that can be read as either Hebrew or foreign are not recognized as well, since context is required for their correct identification.

Hermjakob et al. (2008) describe a method for identifying NEs that should be transliterated in Arabic texts. Given an English-Arabic parallel corpus, the method first tries to find a matching English word for each Arabic word in this corpus. Matching is based on a scoring model which assigns manually-crafted costs to pairs of Arabic and English substrings, allow-

ing for context restrictions. Arabic names are then marked as either names (i.e., candidates for transliteration) or non-names based on this matching algorithm. 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.

This work also presents a transliteration model from Arabic to English. English names are mapped to *consonant skeletons* based on Arabic consonant classes, to serve as reference data. Given an Arabic word to transliterate, English name candidates that match its consonant skeleton are retrieved. These candidates are then ranked according to the scoring model described above. The model is integrated into a machine translation system, and its accuracy, measured by the percentage of correctly translated names, is 89.7%.

Our work is very similar in its goals, but in contrast to Hermjakob et al. (2008) we use minimal amount of 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.

Chapter 3

Methodolgy

Our work consists of two sub-tasks: Identifying TTTs and then transliterating them. We apply machine learning techniques to address both. Specifically, we use the following resources and methods 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 morphologically analyzed (Yona and Wintner, 2008) and POS-tagged (Bar-Haim et al., 2008). We assume that TTTs appear in similar morpho-syntactic contexts and learn these contexts. We acquire training data automatically by marking words that with high probability are to be transliterated, namely, borrowed words. As noted in the introduction, such words have distinctive phonetic and orthographic attributes that assist in identifying them without any context. We use orthographic features consisting of rare Hebrew character n-grams as tagging rules (Chapter 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 training a one-class SVM to identify TTTs (Chapter 4.2).

Transliteration itself is viewed as a simple form of statistical machine translation, where characters are viewed as words, and words are viewed as sentences. We apply Moses (Koehn et al., 2007), a phrase-based SMT toolkit, for training and later for decoding. SMT algorithms make use of a parallel corpus of translation pairs to train a translation model, and a large monolingual corpus of the target language to train a language model (see Chapter 1). We

extract transliteration pairs automatically from titles of Wikipedia articles in Hebrew and English. Retrieving the appropriate pairs relies on a simple mapping of Hebrew consonants to their English counterparts (Chapter 5.2). We construct an English language model by applying SRILM (Stolcke, 2002) to the unigram section of Web 1T corpus (Brants and Franz, 2006).

Previous studies relied on language specific knowledge which needs to be encoded by experts. Hermjakob et al. (2008) use manually-crafted rules for (hundreds of) bilingual pairs of strings to match transliteration pairs in a parallel corpus. They also use heuristics such as considering only capitalized words as transliteration candidates, to enhance the accuracy of their method. Yoon et al. (2007) model sound change patterns for consonants depending on their position in the syllable, and Karimi et al. (2007) explicitly and directly handle specific issues in English-Persian and Persian-English transliteration such as elision, epenthesis and monophtongization. In contrast, our method employs knowledge which can be easily acquired, and requires only very basic knowledge in language specific features. In particular, we do not use a pre-compiled parallel corpus of transliteration pairs, but rather extract one from publicly available resources; nor do we use a parallel bilingual corpus of sentences. Our approach is more language-pair neutral and can therefore be easily adapted to other language pairs with similar resources.

Our evaluation method for the TTT identification task is based on Error Rate Reduction (ERR) of the tagging accuracy. Accuracy is defined as $Accuracy = \frac{C}{N}$, where C is the number of tokens which have been correctly tagged for either transliteration or translation, and N is the total number of tokens. Accuracy error is defined as $\epsilon = 1 - Accuracy$. We compare the error rate of our method, ϵ, with that of a baseline ϵ_0 to obtain the error rate reduction (section 4.3):

$$ERR = \frac{\epsilon - \epsilon_0}{\epsilon_0}$$

.

We evaluate transliteration by measuring its accuracy: the proportion of the tokens with correct transliteration that occur among a list of top-n possible transliterations ranked by

their accuracy ($n = 1, 2, 5, 10$). We also experiment with two methods where the list of possible transliteration is of variable length, depending on the accuracy scores of the transliteration options (section 5.3).

An alternative evaluation method for transliteration is Mean Reciprocal Rank (MRR) (Voorhees, 1999), a measure borrowed from the field of information retrieval. This measure is used when there is a single correct answer among several options and these options are ranked by their probability of correctness, as calculated by the information retrieval system. For a set of queries Q, MRR is the mean of the inverse ranks of the *correct* answers for the queries $q_i \in Q$.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank(q_i)}$$

In the context of transliteration the correct answer is the option that matches the reference in the parallel corpus. This measure, however, requires a parallel corpus from which possible transliterations are extracted. Particularly, this method is not suitable when transliterating terms that do not occur in the parallel corpus or if such corpus does not exist.

We integrate our system as a module in an existing MT system (Lavie et al., 2004a) and provide an external evaluation for the MT system with and without the transliteration module using BLEU and METEOR scores (Chapter 6).

Chapter 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. These rules are used for manually tagging the evaluation corpus. The following guidelines define which terms are to be transliterated rather then translated:

- Persons' first and last names
- Names of streets
- Names of geographical sites such as mountains or rivers e.g., Himalaya - הימליה
- Names of cities and other settlements e.g., London - לונדון, unless a different substitute is commonly used (as in e.g., Jerusalem - ירושלים)
- Names of countries e.g., Congo - קונגו, unless translation or other substitute exists (as in e.g., France - צרפת, Côte d'Ivoire - חוף השנהב)
- Names of regions within a country e.g Galicia - גליציה
- Names of brands and firms, e.g., Chevrolet - שברולט, Elite - עלית
- Names of bands and sport groups, e.g., HaPoel Tel-Aviv - הפועל תל־אביב
- Names of vessels, e.g., Queen Elizabeth - קווין אליזבת, Voyager - וויאג'ר

- Loanwords which retain the sound patterns of their original language with no semantic translation involved, e.g., Internet - **אינטרנט**, Compilation - **קומפילציה**

- Culturally specific terms, with no equivalent in the target language, e.g., Shofar - **שופר**.

We use information obtained from morphological analysis (Itai and Wintner, 2008) and POS tagging (Bar-Haim et al., 2008) to address the problem of identifying TTTs. Each token is assigned a POS and possible morphological analyses given this POS. Tokens are additionally marked if they are not found in a lexicon (Itai et al., 2006), in which case they are considered Out Of Vocabulary (OOV) terms. Our evaluation metric is tagging accuracy, that is, the ratio of correctly tagged tokens, for either translation or transliteration.

4.1 Rule based tagging

Our basic assumption is that OOVs should be transliterated since there is no way to handle them more accurately. However, not all TTTs are OOVs, since many of the TTTs do appear in the lexicon.

Some TTTs can be identified by their surface forms. These words are mainly loan words. For example, the word **ברודקסטינג** (broadcasting) contains a sequence of graphemes that are not frequent in Hebrew: **נג** (*ng*) in a word-final position (Schwarzwald, 2002), and its length is much longer than the average length of Hebrew words (which is approximately 4). We take advantage of these properties and create a simple heuristic system which can recognize some of the TTTs based on their surface form, favoring precision over recall.

We manually generated a list of such features to serve as tagging rules. To construct this list we used 38 character bigrams, 14 trigrams, 8 fourgrams and 6 unigrmas, that are highly unlikely to appear in words of Hebrew origin. Rules associate n-grams with scores, depending on their position (word-initial, word-medial or word-final) 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 integer in the range $[0, 10]$.

The value of c is determined by corpus statistics, where higher values reflect less frequent n-grams in the specific position. A word is tagged for transliteration if the sum of the scores associated with its substrings is higher than a predefined threshold of 10. The two examples below illustrate these rules:

- if קג (qg) occurs in w then add 7 to the score of w.

  The relatively high score associated with this bigram is due to the fact that it is rare in Hebrew.

- if קט (q@) occur in word-final position in w then add 4 to the score of w.

  This bigram is infrequent only at the ends of Hebrew words, and therefore its position is specified in the rule.

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 in which translated terms tend to occur. The underlying assumption is that these 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, we use this automatically-annotated corpus to extract such contexts and bootstrap a TTT classifier, similarly to Collins and Singer (1999).

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 utilize 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).

We trained the classifier using feature vectors generated for the positive examples extracted as described in Section 4.1. Features were based on information obtained from morphological analysis (Itai and Wintner, 2008) and POS tagging (Bar-Haim et al., 2008). The features we used are: Part of Speech (23 values); Definiteness (4 values); Status (5 values); Tense (9 values); Person (6 values); Number (7 values); Gender (3 values);

We experimented with the choice of morpho-syntactic features as well as window size and different kernel functions for the TTTs identification task. The minimal window contains only the the positive example (the target word). Wider windows include also features of adjacent words, with a maximal window of two words to the left and two words to the right of the target word. As for the choice of features, we compared a representation using POS only with one utilizing *all* the features produced by the morphological analyzer.

To evaluate the tagging accuracy of the One-Class SVM classifier, we used an evaluation corpus consisting of 163 sentences of the same genre and domain as the training corpus, containing 1793 tokens. This corpus was tagged according to the guidelines described earlier in this chapter. Table 4.1 presents the identification accuracy obtained for different configurations by features and kernel functions. Each row presents a configuration for a given window, and each cell compares the accuracy obtained for the two choices of features we experimented with: POS only, and all-features obtained from the morphological analysis.

The best identification accuracy was obtained using sigmoid as a kernel function, where each TTT is represented by its POS and the POS of the token preceding it in the sentence.

The results obtained from the classifier show that no single model is substantially better than the others for this task. Moreover, using additional features other than the POS does not necessarily contribute to improving tagging accuracy. This is especially true when taking into consideration tokens occurring after the target word.

T-2	T-1	T	T+1	T+2	Linear	Polynomial	Radial	Sigmoid
		+			90.13/ 91.34	90.13/ 91.34	90.13/ 91.34	90.13/ 93.42
	+	+			91.21/ 92.25	92.70/ 92.02	90.40/ 92.06	**94.05**/ 92.20
+	+	+			92.47/ 92.11	92.11/ 92.06	89.67/ 91.93	91.84/ 92.11
		+	+		90.13/ 92.60	93.78/ 92.11	93.82/ 92.60	92.67/ 92.65
		+	+	+	92.40/ 89.0	93.33/ 89.90	84.72/ 91.70	92.74/ 88.95
	+	+	+		92.43/ 92.38	92.92/ 91.84	90.94/ 91.02	93.01/ 92.38
+	+	+	+		92.29/ 91.12	92.06/ 91.03	89.31/ 90.78	92.29/ 91.12
+	+	+	+	+	92.65/ 86.16	92.29/ 86.11	87.24/ 89.27	92.65/ 86.16

Table 4.1: TTT identification results for different configurations (% of the instances identified correctly, using POS-only/all features)

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 test corpus consists of 25 sentences, from the same genre and domain as the training and evaluation corpora, and contains 98 TTTs. We compare with two baseline models. The naïve baseline always decides to translate; a slightly better baseline consults the lexicon, and tags as TTT any token that the morphological analyzer fails to analyze. We measure our performance in error rate reduction of tagging accuracy compared with our baseline, as described in section 3.

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, as described in section 4.1, our revised method uses a combination of surface-based *and* contextual features. We first examine each token to determine whether it can be identified using surface-form features. If a token has no surface features that identify it as a TTT we consult the one-class SVM and take its decision.

Table 4.2 presents the different configurations we experimented with and their results for

the evaluation set (Dev.) and for the test set (Test). 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 as described above. The rightmost column presents the Error Rate Reduction (ERR) achieved using our method, compared with the second baseline. As can be observed, our method increases classification accuracy both for the evaluation set and for the test set: more than 38% of the errors (in the test set) are reduced.

	Naïve	Baseline	OCS	Our method	ERR
Dev.	89.3	93.2	94.04	**94.4**	17.65
Test	79.9	84.23	88.04	**90.26**	38.24

Table 4.2: 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	<u>bn</u>	<u>ari</u>
kvodo	*heavud*	*shel*	*ben*	*ari*
His-honor	the-lost	of	Ben	Ari

'Ben Ari's lost honor '

Both words *ben* and *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 such terms is crucial for the performance of translation systems.

Chapter 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.

It is appealing to use SMT techniques in order to perform the transliteration task. 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. Alignment models are then used to learn the transliteration from source to target. Furthermore, some problems that are present in SMT are not encountered when using SMT techniques for transliteration:

- Transliteration is monotonic, and therefore reordering of the terms (distortion) is unnecessary.
- Language models do not suffer from sparsity caused by OOV "words".

Several works use this approach for name transliteration (AbdulJaleel and Larkey, 2003; Virga and Khudanpur, 2003; Zhao et al., 2007). AbdulJaleel and Larkey (2003) and Virga and Khudanpur (2003) use it to transliterate names from English, where phonetic information is more apparent than in Hebrew; Zhao et al. (2007) use an alignment method specifically suited for forward-transliteration from Arabic to English. Our work focuses on the hard direction of Hebrew to English transliteration. We do not limit the task to person names or named entities in general. Furthermore, our method works for both forward and backward

transliteration. The key to our high accuracy is the choice of resources, which we detail below.

We use 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 (English) language model

We created a target language model from unigrams of Web 1T corpus (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 n-gram of order 5.

5.2 Hebrew-English translation model

No parallel corpus of Hebrew-English transliteration pairs is available. 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 (see Table 5.1).

From this list we extracted transliteration pairs according to the 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 mapping 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 the Hebrew orthography were also included in the table. For that purpose, the Hebrew alphabet was extended with ג' [ʤ], ז' [ʒ] and צ' [ʧ]. This is true as well for cases where Hebrew encodes

sounds which are not part of the set of English phonemes (such as צ [ts] or ח [ħ]). In such cases, we used a sequence of graphemes that approximates the Hebrew sound (based, for example, on sound patterns of languages such as German, where ח = CH). Every entry in the mapping table consists of a Hebrew letter and a possible Latin letter or letter sequence that might match it. A typical entry is the following:

ש:SH|S|CH |SCH

such that SH, CH, SCH or S are possible candidates for matching the Hebrew letter ש. This table contains 35 entries, of which 10 are mappings of digits to themselves. Constructing such table is trivial and can be done easily by every native Hebrew speaker.

Since we check for similarity of consonants, English words were stripped of their vowels. Hebrew words were stripped of letters that might indicate the presence of vowels (א,ו,י,ע) and these letters also do not appear in the table. Both Hebrew and English titles in Wikipedia may be composed of several words, and words composing the entries in each of the languages may be ordered differently. Therefore, every word in Hebrew has to be matched against every word in English, assuming that titles are short enough. The example in Table 5.1 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 הקבוצה הריקה (*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	
g	r	i	i	@		p	w	l	

d	e	a	d
d			d

Table 5.1: 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 has been done on the development set to optimize translation

performance obtained by the training phase.[1] 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 obtain a list of top-n transliteration options for each entry in the set. The results attained by Moses were further re-ranked to take into account their frequency as reflected in the unigrams of Web 1T corpus (Brants and Franz, 2006). The re-ranking method first normalizes the scores of Moses' results to the range $[0, 1]$. The respective frequencies of these results in Web1T are normalized as well to this range. The score s for 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. We experimented with different values for α, of the range $[0.5, \ldots, 1]$. The value for α which turned out to be optimal in this case is 0.75.

Re-ranking may dramatically affect the transliteration results. Table 5.2 demonstrates a few examples of transliteration results obtained after re-ranking. The top line presents the Hebrew source and the correct transliteration. Each of the following lines is a transliteration option accompanied by its original Moses score (normalized), and the score after re-ranking.

As can be observed, the correct transliteration option, for each of the above examples, does not have the highest Moses score. Nevertheless, re-ranking promotes the correct option for transliteration, in these examples, from the third, fifth and penultimate positions to the first position.

Table 5.3 summarizes the proportion of the terms transliterated correctly across top-n results as achieved by Moses, and the improved results after re-ranking. We experimented

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

ברטי : bertie

translit option	Moses score	score
bertie	0.1456	0.2975
berti	0.1611	0.1452
berty	0.1639	0.1330
brati	0.1383	0.1039
barti	0.1135	0.0865
varti	0.1040	0.0781
brti	0.1036	0.0780
brty	0.0678	0.0512
barty	0.0	0.0181
berte	0.0021	0.0081

שון : sean

translit option	Moses score	score
sean	0.1001	0.229
shon	0.2376	0.179
shawn	0.1517	0.169
shaun	0.1302	0.125
shun	0.1270	0.100
shōn	0.0705	0.053
ssoon	0.0673	0.050
chon	0.0621	0.048
schun	0.0525	0.040
schon	0.0	0.006

שופנהאואר : schopenhauer

translit option	Moses score	score
schopenhauer	0.0047	0.250
chopinhauer	0.2864	0.215
chopinhower	0.2822	0.211
shopenhauer	0.1765	0.134
shopnhower	0.1069	0.080
chopinhousar	0.0491	0.036
shofenhauer	0.0363	0.027
shopaneauer	0.0319	0.024
shupenhauer	0.0259	0.019
shopnhauer	0.0	1.466E-6

Table 5.2: Transliteration example after re-ranking

with two sizes of the candidate list to which re-ranking was applied: 10 and 20, and the rows in the table are marked accordingly.

Since Web1T corpus covers millions of unigrams in English and their frequencies as observed by Google Inc., re-ranking the transliteration options using only these frequencies may seem as a sufficient criterion. However, experiments show that the accuracy obtained when using this method is lower than our re-ranking method. This may be the result of several factors. In some cases all transliteration options produced by Moses do not occur in Web1T corpus, as demonstrated in Table 5.4a. In this case all the options are equally likely. This is equivalent to not assigning weights at all. In other cases, some transliteration options, which do not match the reference transliteration, are extremely common words in English. Thus,

Results	Top-1	Top-2	Top-5	Top-10	Top-20
Moses	68.4	81.6	90.2	93.6	96.6
Re-ranked-10	**76.6**	**86.6**	92.6	93.6	-
Re-ranked-20	69.4	**86.6**	**95.2**	**96.0**	96.6

Table 5.3: Transliteration results (% of the instances transliterated correctly)

each of these words gets a very high score which leaves the other options behind, including the one that matches the reference, as shown in Table 5.4b

זנגרייה : zangariyye	
translit option	score
zangarier	0.1
zangariia	0.1
zangariyye	0.1
zangrier	0.1
zengrier	0.1
zangreyh	0.1
zangarilleux	0.1
zangariah	0.1
sengreyh	0.1
sengariyye	0.1

(a)

אלל : allele	
translit option	score
all	0.9958
alle	0.0016
alla	0.0012
allele	0.0008
ell	4.1311E-4
allel	3.3706E-5
alal	4.9253E-6
a'all	8.4628E-10
aalal	8.4628E-10
élél	8.4628E-10

(b)

Table 5.4: Transliteration results (% of the instances transliterated correctly), re-ranking according to Web1T frequencies only

We further experimented with two methods for reducing the number of transliteration options by taking a *variable* number of candidates, depending on different measures. This is important for limiting the search space of MT systems. In both methods we start from the re-ranked list and select dome of its top candidates according to different criteria. The first method (var1) measures the ratio between each two consecutive options and returns the candidate that scored lower only if this ratio exceeds a predefined threshold. We experimented with threshold values in the range $[0.4, \ldots, 0.8]$, where the lower bound is approximately the average ratio between the first and the second options, and the upper bound is approximately the ratio between other pairs of consecutive options of the top-5. 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 returns all the results whose combined probabilities are at most p. We experimented with p values in the range $[0.4, \ldots, 0.75]$ so that most of the probability mass would be included. The best value in this case was 0.5 , resulting in accuracy of 0.874, and 1.92 results per token on average. Both methods outperform the top-2 accuracy.

Comparing the transliteration quality across different pairs of languages is usually not very meaningful, due to the nature of the attributes associated with different language pairs. Nevertheless, we note that our success rate is similar to or even higher than those reported for languages which share orthographic features with Hebrew, i.e., Arabic-English (AbdulJaleel and Larkey, 2003; Al-Onaizan and Knight, 2002; Matthews, 2007) or Persian-English (Karimi et al., 2007). Our method uses relatively little knowledge about the two specific languages. We do not rely on any pre-compiled list of name pairs, but rather extract one from a more general resource which is publicly available, and we do not use a parallel corpus.

Table 5.5 presents a few examples from the test set that were correctly transliterated by our method. Incorrect transliterations are demonstrated in Table 5.6.

Source		Transliteration
נפש	np$	nefesh
הלמסברגר	hlmsbrgr	hellmesberger
סמבטיון	smb@iwn	sambation
היפרבולה	hiprbwlh	hyperbola
שפרד	$prd	shepard
בשה	ba$h	bachet
חתשפסות	xt$pswt	hatshepsut
ברגנצה	brgnch	berganza
אלישר	ali$r	elissar
ג'ובאני	g'wbani	giovanni

Table 5.5: Transliteration examples generated correctly from the test set

Source		Transliteration	Target
רבינדרנת	rbindrnt	rbindrant	rabindranath
אסוירה	aswirh	asuira	essaouira
כמפיט	kmpi@	champit	chamaephyte
בודלר	bwdlr	bodler	baudelaire
לורה	lwrh	laura	lorre
הוליס	hwlis	ollies	hollies
ונום	wnwm	onom	venom

Table 5.6: Incorrect transliteration examples

Chapter 6

Integration with machine translation

We evaluated both parts of the model independently; we now provide an external evaluation in the context of a Hebrew-to-English MT system. 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, the transliteration procedure described in section 5 is applied to the token to produce the transliteration results. The evaluation measures we use are BLUE (Papineni et al., 2001) and Meteor (Lavie et al., 2004b); we compare the performance of the MT system with and without the transliteration module.

When integrating our method in the MT system we use the best-n transliteration options produced by the transliteration module. The testing corpus consists of 19 Hebrew sentences out a parallel Hebrew-English corpus designed to examine the performance of the MT system. The sentences were chosen such that each of them contains at least one TTT which is transliterated in the parallel English sentence. We implemented an oracle which predicts precisely which tokens are TTTs and transliterates them accurately. This oracle provides an upper bound for the MT system with respect to influence of the transliteration module.

We have experimented with several language models for English, trained on various text sources. The translation evaluation results are presented in Table 6.1, compared to the basic MT system where no transliteration takes place, for each language model used.

The configuration that yields the best translation results uses a language model trained on

news-parl-430MW corpus, as demonstrated in Table 6.1e. Using the transliteration module with this language model shows statistically significant improvement in METEOR scores ($p < 0.05$). METEOR scores are a better evaluation measure here since they reflect improvement in recall, but note that the BLEU scores improve as well.

System	BLEU	METEOR
Base	8.23	33.94823
Top-1	8.06	43.87316
Top-10	8.67	35.60303
upper bound	12.19	43.87316

(a) nyt-heb

System	BLEU	METEOR
Base	9.02	34.54124
Top-1	8.85	36.70513
Top-10	9.35	35.60303
upper bound	12.63	36.61574

(b) nyt-heb-300M

System	BLEU	METEOR
Base	8.62	35.31502
Top-1	8.57	38.13971
Top-10	8.96	37.59647
upper bound	12.51	44.42637

(c) ArabicPlusXinhuaNewsPlusAENewsBi

System	BLEU	METEOR
Base	10.64	36.68081
Top-1	10.86	38.99727
Top-10	9.76	37.70229
upper bound	14.62	46.36910

(d) OneGigaWordPlusHebNyt

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
upper bound	12.58	44.89328

(e) news-parl-430MW

Table 6.1: Integration of transliteration module in MT system using English language models trained on different text sources

We further experimented with a method which uses Minimum Error Rate Training to learn the contribution of the transliteration module to the translation process. The results of learning the weight this module are presented in Table 6.2 compared to the MT system where it was not used. Ideally, the scores of the transliteration options should also be taken into account when considering the translation. Unfortunately, the current system cannot exploit this information and we therefore provide all the results without their scores.

System	BLEU	METEOR
Base	8.715	35.37779
Translit	10.35	39.15054

Table 6.2: Integration of transliteration module in MT system using MERT

Chapter 7

Conclusions

We presented a 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. It reduces over 38% of the errors 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, easy to obtain resources. The correct transliteration is generated in more than 76% of the cases, and in more than 95% of the instances it is one of the top-5 results. We have integrated our method with an existing MT system, to provide an external evaluation, and demonstrated a small yet significant improvement.

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.

References

Nasreen AbdulJaleel and Leah S. Larkey. Statistical transliteration for English-Arabic cross language information retrieval. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 139–146, New York, NY, USA, 2003. ACM. ISBN 1-58113-723-0.

Yaser Al-Onaizan and Kevin Knight. 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, 2002. Association for Computational Linguistics.

M. Arbabi, S. M. Fischthal, V. C. Cheng, and E. Bart. Algorithms for Arabic name transliteration. *IBM J. Res. Dev.*, 38(2):183–194, 1994. ISSN 0018-8646.

Roy Bar-Haim, Khalil Sima'an, and Yoad Winter. Part-of-speech tagging of modern hebrew text. *Nat. Lang. Eng.*, 14(2):223–251, 2008. ISSN 1351-3249.

Shaul Bolozky. Some aspects of Modern Hebrew phonology. In Ruth A.Berman, editor, *Modern Hebrew Structure*, pages 11–67. University Publishing Project, Tel Aviv, 1978.

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

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

Andrew J. Carlson, Chad M. Cumby, Jeff L. Rosen, and Dan Roth. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May 1999.

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

Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the 1999 Joint SIGDAT Conference on EMNLP and VLC*, 1999. URL `http://citeseer.ist.psu.edu/196394.html`.

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

Dan Goldwasser and Dan Roth. Active sample selection for named entity transliteration. In *Proceedings of ACL-08: HLT, Short Papers*, pages 53–56, Columbus, Ohio, June 2008a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-2014`.

Dan Goldwasser and Dan Roth. Transliteration as constrained optimization. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 353–362, Honolulu, Hawaii, October 2008b. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D08-1037`.

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

Fei Huang. *Multilingual Named Entity Extraction and Translation From Text and Speech*. PhD thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 2005.

Intel. Gale kickoff release - Intel transliteration standard v1.0, LDC2005G02. Linguistic Data Consortium, Philadelphia, 2005.

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

Alon Itai, Shuly Wintner, and Shlomo Yona. 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, 2006.

Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. Collapsed consonant and vowel models: New approaches for english-persian transliteration and back-transliteration. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 648–655, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Kevin Knight and Jonathan Graehl. Machine transliteration. In Philip R. Cohen and Wolfgang Wahlster, editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 128–135, Somerset, New Jersey, 1997. 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. 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 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P07/P07-2045`.

Alon Lavie, Erik Peterson, Katharina Probst, Shuly Wintner, and Yaniv Eytani. 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 2004a.

Alon Lavie, Kenji Sagae, and Shyamsundar Jayaraman. 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, 2004b. ISBN 3-540-23300-8.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1987.

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

Refael Nir. *Musagim beValshanut Shimushit (Concepts in Applied Linguistics)*. Ministry of Education and Culture, Jerusalem, Israel, 1979. In Hebrew.

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

Jong-Hoon Oh and Key-Sun Choi. An English-Korean transliteration model using pronunciation and contextual rules. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

Jong-Hoon Oh, Key-Sun Choi, and Hitoshi Isahara. A comparison of different machine transliteration models. *J. Artif. Intell. Res. (JAIR)*, 27:119–151, 2006.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 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, 2001. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073083.1073135.

Rachel Rosner. The formation of foreign words in Hebrew – research and language textbooks. *BaMichlala - The College Voice: Research, Essays and Literary Works*, 19:89–104, 2007. In Hebrew.

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

Ora R. Schwarzwald. *Studies in Hebrew Morphology*. The Open University, Tel-Aviv, Israel, 2002.

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

Andreas Stolcke. Srilm – an extensible language modeling toolkit. In *Proc. Int. Conf. Spoken Language Processing (ICSLP 2002)*, pages 901–904, 2002.

Tao Tao, Su-Youn Yoon, Andrew Fister, Richard Sproat, and ChengXiang Zhai. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 250–257, Sydney, Australia, July 2006. Association for Computational Linguistics.

Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

Paola Virga and Sanjeev Khudanpur. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition*, pages 57–64, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

Ellen M. Voorhees. The trec-8 question answering track report. In *TREC*, 1999.

Shlomo Yona and Shuly Wintner. A finite-state morphological grammar of Hebrew. *Natural Language Engineering*, 14(2):173–190, April 2008.

Su-Youn Yoon, Kyoung-Young Kim, and Richard Sproat. 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 2007.

Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P07/P07-1015.

Bing Zhao, Nguyen Bach, Ian Lane, and Stephan Vogel. A log-linear block transliteration model based on bi-stream HMMs. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 364–371, Rochester, New York, April 2007. Association for Computational Linguistics.