Identification of Transliterated Foreign Words in Hebrew Script

Yoav Goldberg and Michael Elhadad

Computer Science Department
Ben Gurion University of the Negev
P.O.B 653 Be'er Sheva 84105, Israel
{yoavg,elhadad}@cs.bgu.ac.il

Abstract. We present a loosely-supervised method for context-free identification of transliterated foreign names and borrowed words in Hebrew text. The method is purely statistical and does not require the use of any lexicons or linguistic analysis tool for the source languages (Hebrew, in our case). It also does not require any manually annotated data for training – we learn from noisy data acquired by over-generation. We report precision/recall results of 80/82 for a corpus of 4044 unique words, containing 368 foreign words.

1 Introduction

Increasingly, native speakers tend to use borrowed foreign terms and foreign names in written texts. In sample data, we found genres with as many as 5% of the word instances borrowed from foreign languages. Such borrowed words appear in a transliterated version. Transliteration is the process of writing the phonetic equivalent of a word of language A in the alphabet of language B. Borrowed words can be either foreign loan words with no equivalent in language B, or words from language A used as slang in language B. Identifying foreign words is not a problem in languages with very similar alphabets and sound systems, as the words just stay the same. But this is not the case in words borrowed from languages that have different writing and sound systems, such as English words in Japanese, Hebrew and Arabic texts.

Transliterated words require special treatment in NLP and IR systems. For example, in IR, query expansion requires special treatment for foreign words; when tagging text for parts of speech, foreign words appear as unknown words and the capability to identify them is critical for high-precision PoS tagging; in Machine Translation, back transliteration from the borrowed language to the source language requires the capability to perform the inverse operation of transliteration; in Named Entity Recognition and Information Extraction, the fact that a word is transliterated from a foreign language is an important feature to identify proper names.

We focus in this paper on the task of identifying whether a word is a transliteration from a foreign language – and not on the task of mapping back the

transliteration to its source. In some languages, such as Japanese, identifying borrowed foreign words is easy – they are written in a special script (Katakana) used just for such purposes. Other languages don't have such luxury. In this paper we describe a method for identifying transliterated words in Hebrew, written in the Hebrew script. The method is unsupervised, uses easily acquired resources, and is not specific to Hebrew.

The Modern Hebrew writing system has properties that make it different from most Indo-european writing systems, and which make transliteration identification difficult (similar properties are shared with all Semitic languages):

- Vowels are not written in most cases, but in some cases, vowels are indicated by inserting specific letters (for the sounds *i* and *o*). The same letters can be used either as vowels or as consonants (*e.g.*, yod and vav).
- The sounds *p* and *f* are encoded by the same letter, similarly for the pairs *b* and *v* and *s* and *sh*. The sounds *th* and *j* do not exist in Hebrew.
- The letters *ayin, het, khaf* in Hebrew have no corresponding sounds in most European languages.
- In the written form, words are agglutinated with a sequence of prefixes and suffixes (corresponding to prepositions and pronouns).

In the rest of the paper, we first review previous work related to the task of transliteration identification. We then present our approach: we identify transliteration by comparing the letter structure of words with models trained in a way that captures the sound structure of the language – one in Hebrew and one in English, as written in the Hebrew writing system. The model for Transliterated English is trained on data automatically generated from an English corpora, using the CMU pronunciation dictionary and very simple phoneme to letter rules.

2 Related Work

There is a large body of work on transliteration [1,2,3] which mainly focuses on back-transliteration.

There is a growing body of research on automatic extraction of transliterated pairs [4,5]. Sherif and Kondrak [5] use seed examples and a sentence aligned English/Arabic text to jointly learn a bilingual string distance function and extract transliterated pairs. While this work aims at complete alignment, our task is only the identification of transliterated candidates. Identification of transliteration candidates can help full alignment by relaxing the need for aligned text.

The task of identifying transliterated words has been less studied. Stalls and Knight [1] identified the problem – "...in Arabic, there are no obvious clues, and it's difficult to determine even whether to attempt a back-transliteration, to say nothing of computing and accurate one" – but don't deal with it directly.

Oh and Choi [6] studied identification of transliterated foreign words in Korean text, using an HMM on the word syllable structure. They used a corpus of about 1,900 documents in which each syllable was manually tagged as being either Korean or Foreign, and achieved impressive results. However, beside

requiring a large amount of human labor, their results are not applicable to Hebrew (or Arabic) as these languages' syllables structure is not clearly marked in writing, and even the vowels are not available in most cases.

Nwesri *et al.* [7] dealt with the identification of transliterated foreign words in Arabic text in the setting of an information retrieval system. They tried several approaches: using an Arabic lexicon (everything which is not in the lexicon is considered foreign), relying on the pattern system of Arabic morphology, and two statistical ngram models, the better of which was based on Cavnar and Trenkle's rank order statistics [8], traditionally used for language identification. For the statistical methods, training was done on manually constructed lists of a few thousands Arabic and foreign words written in Arabic script. They also augmented each of the approaches with hand written heuristic rules.

They achieved mediocre results on their training set, somewhat lower results for their test set, and concluded that the best approach for identification of transliterated foreign words is the lexicon-based method enhanced by hand written heuristics, by which they achieved a precision of 68.4% and recall of 71.1% on their training set, and precision of 47.4% and recall of 57.2% on their test set.

Another related field of research is that of language identification, in which documents are classified according to their language. The problem of finding transliterated foreign words can be regarded as performing language identification on individual words instead of documents or sentences. Algorithms that rely on letter-ngram statistics can be relevant to the foreign words identification task. Two notable works are [8] and [9], both based on letter-level ngram statistics. Canvar and Trenkle use rank order statistics, and Dunning use Naive-Bayes classification with a trigram language model, and add-one smoothing.

Language identification can be considered a 'solved problem', with success rates of above 99.8%. However, such systems usually require a minimum of about 50 bytes of text in order to perform well. This data is not available when working on a single word. Indeed, Nwesri *et al.* [7] report low success rates using a modification of Cavnar's method for Arabic foreign words identification. Qu and Grefenstette [10] used Cavnar's method for the more limited task of language identification of names, to distinguish English, Japanese and Chinese names in Latin script. Training on 88k English names, 83k Japanese names and 11k Chinese names, they achieved accuracies of 92% (Japanese), 87% (Chinese) and 70% (English).

3 Our Approach

We concentrate on identifying borrowed and transliterated words in Hebrew text. As most borrowed words come from English or European sources, we concentrate on finding borrowed words from such origins. We also focus on a context-free approach, which works on words alone, without requiring their context. Our intuition is that Hebrew/Arabic words sound different than English/Indo-European words, and as letters are closely related to sounds, this should be reflected in

their writing. Therefore, we believe that letter level ngram approaches should be applicable to this problem, given sufficient data and suitable language models, even at the word level.

Naive-Bayes Classification At its core, our method is a Naive-Bayes classifier for choosing the best generative language model for a given word: Assuming two generative language models, M_n for generating native (Hebrew) words and M_f for generating foreign words, and an observed word w, we would like to find the model M_i that maximizes $P(M_i|w)$. As we don't know $P(M_i|w)$, we use Bayes Formula, and get:

$$P(M_i|w) = P(w|M_i)P(M_i)/P(w)$$

$P(w) = 1$ (the word is given), and we assume both models are equally probable ($P(M_n) = P(M_f)$), so we are left with the problem of evaluating $P(w|M_i)$. By normalizing the results:

$$P'(M_f|w) = \frac{P(M_f|w)}{P(M_f|w) + P(M_n|w)}$$

$$P'(M_n|w) = \frac{P(M_n|w)}{P(M_f|w) + P(M_n|w)}$$

we can get not only a decision, but a probability for each category. This is a standard construction (see [9,11]). We differ from earlier work in the model parameters estimation, and by the data from which we learn.

Unsupervised Approach Our model is unsupervised, in that it does not require any manually tagged data for training. Our approach needs data to learn and estimate the parameters $P(w|M_i)$. Unfortunately, there is no large corpus of transliterated foreign words in Hebrew script available, and manually creating one is labor intensive. For training their language model of Foreign words in Arabic script, [7] created a list of 3046 foreign words, and achieved rather low results. We identify two problems with this approach:

1. They learn from a set of words, without taking into account the frequencies of these words in the language, which results in a poor approximation of the language 'sound-patterns'.
2. The list of words constructed manually is just too small to be useful – most machine learning approaches will not get enough data to reach reliable results from such a quantity.

Instead of using a small amount of quality data, we opted for a large quantity of noisy data. For estimating the native language model we use a collection of Modern Hebrew texts from a period of about 100 years ago, which are assumed to have a relatively low frequency of foreign words. For estimating the foreign language model, we use over-generation, and create a large and noisy corpus. This noisy data performs much better than using just a small amount of hand-made clean data.

The Over-Generation Process At the heart of the over-generation mechanism is a procedure $pron- > nat$ which, given a foreign word in foreign script, outputs many possible transliterations of that word in the native script. These transliterations are meant for machines to learn from, and need not be 'correct' by human standards. Indeed, most of them will not be accepted as proper transliterations of the word by humans. They do, however, seem to capture the 'essence' of the foreign language in native writing.

The algorithm we use is as follows:

1. find a phonemic representation of the word
2. list all common ways of writing each phoneme in the native script
3. output all the combinations of writing produced in stage 2

For step 1, we look up the word in the CMU pronunciation dictionary (http://www.speech.cs.cmu.edu/cgi-bin/cmudict).[1] We ignore stress information, and distinguish phonemes at the start of words from the rest of the phonemes[2]. We also convert the 'T S' phoneme sequence into a new 'TS' phoneme, because this sequence can be represented in Hebrew by the single letter '**צ**'. We also change all 'K' phonemes into a new 'KH' phoneme if the word contains the 'ch' letter sequence and the pronunciation does not have a 'CH' phoneme.

Step 2 is achieved by a simple table lookup. The table is easy to construct, and presented below (Table 1).

As a concrete example, in Hebrew, the English word 'county' has two pronunciations in the cmudict: 'K AW1 N T IY0' and 'K AW1 N IY0'. These are converted to: '*K AW N T IY' and '*K AW N IY' ('*' marks a phone at the beginning of a word). The translation table indicates that:
'*K' : '**ק**'
'AW' : '**או**', '**אוו**', '**ואו**'
'N' : '**נ**'
'T' : '**ת**', '**ט**'
'IY' : '**י**', '**אי**', ϵ

and so the resulting 26 transliterations are (only 3 of them, underlined, would be accepted by humans):

קאונתי קאונתאי קאונת קאונטי קאונטאי קאונט קאוונתי קאוונתאי קאוונת קאוונטי
קאוונטאי קאוונט קואונתי קואונתאי קואונת קואונטי קואונטאי קואונט קואוני קאונאי
קאוונ קאווני קאוונאי קאוונ קואוני קואונאי קואונ

[1] Our approach does not require any new and task specific lexicons, and crucially no lexicon is needed for the source (*e.g.*, Hebrew) language. Furthermore, for target languages with simpler letter-to-sounds rules, such as Spanish or Italian, a more direct, lexicon free approach could be taken.

[2] Because the Hebrew writing system does not include vowels, vocalized phonemes are transliterated differently when they are word-initial: word initial vowels are always represented in the written Hebrew form, while non word-initial vowels might be dropped and not appear in the written form.

Table 1: Pronunciation to Hebrew Translation Table

Phoneme	Possible Hebrew Rendering	Phoneme	Possible Hebrew Rendering
AA	או , א , ו	*AA	או , א
AH	ϵ , א	*AH	א
AW	או , אוו , ואו	*AW	או , אוו , אאו
AE	ϵ , א	*AE	א
AO	או , אוו , ו	*AO	או , אוו
AY	אי , יי	*AY	אי
B	ב	CH	צ׳
D	ד	DH	ד , ת , ת׳
EH	ϵ , א	*EH	א
ER	אר , ר	*ER	אר
EY	אי , יי , י	*EY	אי
F	פ	G	ג
HH	ה	IH	י , אי
*IH	אי	IY	ϵ , אי , י
*IY	אי	JH	ג׳
K	ק	L	ל
M	מ	N	נ
NG	נג	OW	או , אוו , ו , וו
*OW	או , אוו	OY	אוי , וי
*OY	אוי	P	פ
R	ר	S	ס
SH	ש	T	ת , ט
TH	ת׳ , ד	UH	ו
*UH	או	UW	ו
*UW	או	V	ב , ו , וו
W	וו	Y	י
Z	ז	ZH	ז׳ , ז׳
TS	צ , טס	KH	ק , כ
N*	ן	M*	ם
H*	ך	F*	ף

We then apply this technique to a large body of foreign (English) text, to produce a large corpus for training (6MB of the Brown Corpus provided by the NLTK toolkit [12] resulted in 126MB of 'foreign Hebrew' data).

Probabilistic Model Now that we have a fair amount of training data for both languages, we should decide on the specific probabilistic model. As a baseline, we used the estimation Dunning [9] used for language identification. The setting is a generative Markov Model, in which every letter is assumed to be dependent only on the n previous letters[3]. Dunning used a model of depth 2 (a trigram model), we tried models of depth 2 and 3.

[3] Word boundaries are also considered as letters, and we treat each of the Hebrew ז׳ צ׳ ג׳ as one letter as well (צ׳ , ז׳ and ג׳ are used in Modern-Hebrew script to represent the sounds tch, dj, and j, found in many foreign words. These sounds are not available in traditional Hebrew, and so don't have unique characters.)

In this model, the probability of a word w (made of letter $w_1 \dots w_N$) is:

$$\prod_{n \le i \le N} P(w_i | w_{i-1}, \dots, w_{i-n})$$

The most natural estimation for $P(w_i | w_{i-1}, \dots, w_{i-n})$ is:

$$P(w_i | w_{i-1}, \dots, w_{i-n}) = \frac{C(w_i w_{i-1} \cdots w_{i-n})}{C(w_{i-1} \cdots w_{i-n})}$$

Where $C(.)$ are the counts collected from the training data, and we set $C(.) = 0$ for every value that appears less than K times (we set $K = 5$), as these counts are considered to be unreliable, esp. when dealing with such a small alphabet. For smoothing, Dunning used Laplace's add-1 method, which seems reasonable for small alphabets (and performs very well in practice for language identification).

Table 3 lists the results of using trigram and 4-gram models with Laplace smoothing (Dun3 and Dun4), on the data described in Section 4. The trigram model performs better, achieving a precision of 58.7% and recall of 64.9%.

The results are fine (much better than was previously achieved for the Arabic case[4] [7] who reported precision/recall of 47%/57% on the test data using a lexicon), but still far from perfect. We suggest below several modifications to improve these results.

Non-traditional Smoothing Laplace's add-one smoothing seems too simplistic for this task. We note that using a higher-order model yields somewhat worse results in terms of precision (58.7% for the trigram model vs. 55.6% for the 4gram model) while only slightly improving recall (from 64.9% to 65.7%). Using more advanced smoothing methods, such as these used by [13] didn't improve the results. An interesting observation is that our small alphabet together with our practically unlimited training set data means that smoothing will hardly be needed for unigram or bigram models. However, we would like to have the extra discrimination power we can get from 3- and 4-gram models, when available. To achieve this, we create 4 different models for each language ($M_{i1} \dots M_{i4}$, which are unigram, bigram, trigram and 4gram models) and linearly combine them:

$$P(w|M_i) = \lambda_1 P(w|M_{i1}) + \lambda_2 P(w|M_{i2}) + \lambda_3 P(w|M_{i3}) + \lambda_4 P(w|M_{i4})$$

If we keep $\sum_{i \in 1 \dots 4} \lambda_i = 1$, $P(w|M_i)$ stays a proper probability. One can view this combination as a weighted voting scheme.

Note that this differs from traditional back-off estimation based smoothing, which is also linear combination of lesser degree ngrams, but on the ngram level, whereas we work on the model level.

Setting the best values of λ_is is yet to be decided, for the current experiment we set them all to an equal value of 1/4, which appears to work well empirically, as can be seen in Table 3 (Vot).

[4] The results are not directly comparable, but we believe the differences in performance are big enough to be indicative of the improved performance of our method.

Note that this scheme does leave a structural 0 in the model: for example the overgeneration procedure for Hebrew does not produce the letters ח or ע, as these sounds rarely occur in foreign words, if at all. As a result, even after our 'smoothing' words with these letters will have a 0 probability of being foreign, even in the unigram model. But this is fine, as it results in the right decision. In some respects, this is a good feature of this smoothing method.

Backward Model When working with such a short words (the average word in our corpus is about 5.5 letters long), we cannot afford to miss clues that can be ignored by methods working on larger samples. In addition to the forward-moving Markov process, in which every letter is dependent on the n previous ones, we hypothesise a backward moving process in which every letter is dependent on the n following ones, $P(w_1, \ldots, w_N) = \prod p(w_i|w_{i+1}, \ldots, w_{i+n})$. These probabilities are in many ways similar to the forward moving ones, but there are slight variations in some cases. We add 3 such models (bigram, trigram, 4gram) to our voting scheme:

$$P(w|M_i) = \lambda_1 P(w|M_{i1}) + \lambda_2 P(w|M_{i2}) + \lambda_3 P(w|M_{i3}) + \lambda_4 P(w|M_{i4}) + \lambda_5 P(w|M_{i2back}) + \lambda_6 P(w|M_{i3back}) + \lambda_7 P(w|M_{i4back}) \quad (1)$$

The addition of the backward models to the vote (Vot+Back in Table 3) results in improved precision (from 76.3 to 80.4), and slightly hurt recall (from 71.7 to 71.4).

All Foreign Words are Not Created Equal Examining foreign words in Hebrew text reveals that a very big proportion of them are proper names. Although many English words function also as proper names, the sound patterns of proper names are somewhat different than those of regular English words. As they represent much of the data we want to identify, they deserve a special treatment. This special treatment is achieved by adding another 'language' to our identification scheme – the language of foreign proper names, and building a model M_{fn} for that language.

The foreign names corpus is brutally constructed by creating a sub-corpus of the English one, containing all words that appear only in capitalized form, and then over-generating the corresponding native version. This is noisy in several respects: some of the capitalized words in the Brown corpus are not really proper names but rather plain nouns or modifiers[5], and the representation of proper names in cmudict is not very complete. For this reason, the resulting model can not reliably decide between 'English' and 'English name', but it does succeed in approximating some of the properties of names which get lost in the mass of all English words.

We then apply our NaiveBayes classifier on the word, and get 1 of 3 possible decisions: $native$, $foreign$, $foreign\_name$. We treat either $foreign\_name$ or

[5] Note that we consider only words for which **all** occurrences are capitalized, thus we rule out most of this kind of noise.

$foreign$ as $foreign$. This results in our best model so far, in terms of recall (82 vs. 71), even if somewhat hurting precision (80.1 vs. 80.4).

Adding a Lexicon Having achieved reasonable precision and recall by using statistical information alone and no explicitly annotated data, we turn to assess the effect of adding knowledge from a manually created lexicon of Hebrew words. For our lexicon, we use the word-list of the Hebrew spell checker HSPELL [14]. This lexicon contains 404,640 unique words.

As a baseline (HSPELL in Table 4), we consider all the words appearing in the lexicon as Hebrew, and all the other words as Foreign. This results in our best recall so far (85%), coupled with very poor precision (13%). Next, we combine the lexicon with the statistical models by considering every word appearing in the lexicon as Hebrew, and using the statistical model for deciding the rest of the words. This is done for the voted forward and backward model (Vot+Back+HSPELL) as well as for the model considering proper-names (Vot+Back+Names+HSPELL).

The lexicon increases the precision of each model by about 11%, while dropping the recall by about 10%. We note that the effects of incorporating the proper-names model are orthogonal to the addition of lexicon knowledge. The decrease in recall is due to very common borrowed words (*e.g.,* "Blues", "Internet", "Single"), and some common foreign names (*e.g.,* "Madonna", "Paul", "Glenn") which are included in the lexicon.

Comparison with a Supervised Method In Section 3, we claim that our method of learning with a lot of noisy data is better than learning with a small amount of quality hand annotated data. To support this claim, we perform a 5-fold cross validation experiment, in which we use some of our hand annotated data to train the trigram, voted, and voted+backward models, and test on the rest of it.

The results are summarized in Table 2. As can be seen, the models trained on the automatically generated noisy data (Table 3) consistently outperform the supervised models trained on the small amount of data. Contrary to the case with the generated noisy data, there is no real difference between the voted and voted+back models for the small supervised data.

4 Evaluation

Experiment Settings For training the Hebrew model, we use the prose and mass sections of the 'Ben Yehuda Project' (the Hebrew equivalent of 'Project Gutenberg', containing copyright-free modern-Hebrew writings (prose, poetry and essays) of 26 authors, all of which died more than 70 years ago), overall 28MB of text.

For the foreign models we use text over-generated from 6MB portion of the Brown Corpus distibuted with the NLTK toolkit [12]. For testing on the He-

brew case, we use words from 50 articles from the 'gossip' section of Ynet[6], a total of 9618 words, 4044 of them unique. We manually removed the prefixes (prepositions) from all words, and tagged them as either H(ebrew), F(oreign) or FN(foreign name). Overall, there are 3608 Hebrew words, 368 foreign words of which 251 are foreign proper names, and another 68 words which are ambiguous (these words either have a clear foreign origin but became so common that they sound Hebrew, or they have two readings, one Hebrew and one borrowed). The data is available at (http://www.cs.bgu.ac.il/∼yoavg/ForeignWordsId).

For the cross validation experiments described in Section 3, we took for each fold a different 20% of the Hebrew words and 20% of the Foreign words for testing, and the rest for training.

Results The following tables summarize the results of all the experiments described above.

Table 2: Results on Ynet Gossip Section, Supervised via 5-fold Cross Validation

Experiment	Precision (%)	Recall (%)
Dun3(CV)	17.75	60.42
Vot(CV)	59.72	60.81
Vot+Back(CV)	59.70	60.76

Table 3: Results on Ynet Gossip Section, using Purely Statistical Models

Experiment	Precision (%)	Recall (%)
Dun3	58.7	64.9
Dun4	55.6	65.7
Vot	76.3	71.7
Vot+Back	80.4	71.4
Vot+Back+Names	80.1	82

Table 4: Results on Ynet Gossip Section, using Statistical Models together with a Lexicon

Experiment	Precision (%)	Recall (%)
HSPELL	13	85
Vot+Back+HSPELL	91.86	61.41
Vot+Back+Names+HSPELL	91.19	70.38

[6] http://www.ynet.co.il – a popular Israeli news portal

Precision is the ratio between the number of correctly identified foreign words and the total number of words identified as foreign. Recall is the ratio between the number of identified foreign words, and the real number of foreign words. The 68 ambiguous words were excluded from the calculations.

5 Open Issues and Future Work

Segmentation In both Hebrew and Arabic, some prepositions are appended as prefixes to words. All the results presented here ignored that fact and were on words without prefixes. This is suitable for IR settings, but not good enough for the general NLP application, in which text appear unsegmented. This prefixation can make otherwise 'foreign' words to get tagged as 'Hebrew'. Although recent morphological disambiguators for Hebrew [15,16] perform such segmentation with reasonable accuracy (above 98%), they all rely on a Hebrew lexicon. By its very nature, such lexicon is bound not to be complete in its coverage of transliterated foreign words, and so the above-mentioned disambiguators unfortunately fall on the wrong side of the pipeline – it is precisely for improving such systems that foreign words identification is needed. Dealing with unsegmented data is an interesting problem, left for future work.

Context Our method is context free, and assigns probabilities to words. But context does matter in some cases (for example, some words can be either foreign or native, depending on the reading). Context can also help in the segmentation problem. In addition, integrating our method with the output of a POS tagger is also a promising direction.

Arabic The Arabic language is similar to Hebrew in terms of the sound patterns and writing system. Therefore, we expect that given the proper training material, our method will perform well for Arabic as well.

Different Kinds of Borrowings This work focuses on identification of foreign words, yet no real distinction was made between cognates, borrowings and "real" transliterations. For most parts we can indeed ignore this distinction: "heavily incorporated" cognates and borrowing, which we consider as native words, tend to adopt the Hebrew pronunciation (and writing), and are considered Hebrew also by our algorithm, while less incorporated cases of borrowing exhibit inconsistent writing patterns, and demand special treatment similar to that of "pure" transliterations. However, finding an algorithmic way of separating these groups is an interesting research question.

6 Conclusions

We presented a method for identification of borrowed words and transliterated names in Hebrew text. The method is very loosely supervised, does not require annotated data, and achieves satisfactory results (precision/recall of 80%/82%

with a purely statistical method, and 91%/70% when combined with a Hebrew lexicon). We verified that our approach of learning from a large amount of automatically generated data greatly outperforms learning with a small amount of manually annotated data. Giving specific care to identification of proper nouns was shown to improve performance.

References

[1] B. Stalls and K. Night: Translating Names and Technical Terms in Arabic Text. In: Proc. of the COLING/ACL Workshop on Comp. Approaches to Semitic Languages. (1998)

[2] Y. Al-Onaizan and K. Knight: Machine Transliteration of Names in Arabic Text. In: Proc. of ACL Workshop on Comp. Approaches to Semitic Languages. (2002)

[3] Yoon, S.Y., Kim, K.Y., Sproat, R.: Multilingual transliteration using feature based phonetic method. In: Proc. of ACL. (2007)

[4] Klementiev, A., Roth, D.: Named entity transliteration and discovery from multilingual comparable corpora. In: Proc. of NAACL. (2006)

[5] Sherif, T., Kondrak, G.: Bootstrapping a stochastic transducer for arabic-english transliteration extraction. In: Proc. of ACL. (2007)

[6] Oh, J., Choi, K.: A statistical model for automatic extraction of korean transliterated foreign words. Int. J. of Computer Proc. of Oriental Languages **16** (2003)

[7] Nwesri, A.F., Tahaghoghi, S., Scholer, F.: Capturing out-of-vocabulary words in arabic text. In: Proc. of EMNLP2006. (2006)

[8] Cavnar, W.B., Trenkle, J.M.: N-gram-based text categorization. In: Proc. of SDAIR-94. (1994)

[9] Dunning, T.: Statistical identification of language. Technical report, Computing Research Lab, New Mexico State University (1994)

[10] Qu, Y., Grefenstette, G.: Finding ideographic representations of japanese names written in latin script via language identification and corpus validation. In: Proc. of ACL. (2004)

[11] Manning, C.D., Schutze, H.: Foundations of Statistical Natural Language Processing. MIT Press (1999)

[12] Bird, S., Loper, E.: NLTK: The natural language toolkit. In: The Comp. Vol. to the Proc. of ACL. (2004)

[13] Thede, S.M., Harper, M.P.: A second-order hidden markov model for part-of-speech tagging. In: Proc. of ACL. (1999)

[14] Har'el, N., Kenigsberg, D.: HSpell - the free Hebrew spell checker and morphological analyzer. Israeli Sem. on Comp. Ling. (2004)

[15] Adler, M., Elhadad, M.: An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In: Proc. of the ACL. (2006)

[16] Shacham, D., Wintner, S.: Morphological disambiguation of Hebrew: A case study in classifier combination. In: Proc. of EMNLP-CoNLL. (2007)