Processing Judeo-Arabic Texts

Kfir Bar, Nachum Dershowitz, Lior Wolf,
Yackov Lubarsky, and Yaacov Choueka

Abstract. Judeo-Arabic is a language spoken and written by Jewish communities living in Arab countries. Judeo-Arabic is typically written in Hebrew letters, enriched with diacritic marks that relate to the underlying Arabic. However, some inconsistencies in rendering words in Hebrew letters increase the level of ambiguity of a given word. Furthermore, Judeo-Arabic texts usually contain non-Arabic words and phrases, such as quotations or borrowed words from Hebrew and Aramaic. We focus on two main tasks: (1) automatic transliteration of Judeo-Arabic Hebrew letters into Arabic letters; and (2) automatic identification of language switching points between Judeo-Arabic and Hebrew. For transliteration, we employ a statistical translation system trained on the character level, resulting in 96.9% precision, a significant improvement over the baseline. For the language switching task, we use a word-level supervised classifier, also showing some significant improvements over the baseline.

1 Introduction

Judeo-Arabic is a set of dialects spoken and written by Jewish communities living in Arab countries, mainly during the Middle Ages. Judeo-Arabic is typically written in Hebrew letters, and since the Arabic alphabet is larger than the Hebrew one, additional diacritic marks are added to some Hebrew letters when rendering Arabic consonants that are lacking in the Hebrew alphabet. Judeo-Arabic authors often use different letters and diacritic marks to represent the same Arabic consonant. For example, some authors use ג (Hebrew *gimel*) to represent ج (Arabic *jim*) and גֿ to represent غ (*ghayn*), while others reverse the two. This inconsistency increases the level of ambiguity of a given word, making the reading of Judeo-Arabic texts a challenging task even for an Arabic speaker. For instance, the letter י (Hebrew *yod*) sometimes represents the letter ي (*ya*), such as in the word פי (في, "in/inside"), sometimes represents the letter ى, such as in the word סילת (سُئِلتُ, "I was asked"), and sometimes the letter ئ, as in עלי (علَى, "on/to/at"). In addition, the special signs, *hamza* (the glottal stop), *maddah* (the glottal stop, followed by a long *a* vowel), *waslah* (unpronounceable alif), *shadda* (gemination), as well as short vowels, are usually not marked in the text. Furthermore, Judeo-Arabic texts are often peppered with Hebrew and Aramaic citations and borrowings, which cannot be *transliterated* into Arabic, but rather need to be *translated* into Arabic. Those embedded words sometimes get inflected following Arabic morphological rules; for example, אלשכינה (*al-shkhina*, "the divine spirit"), where the prefix *al* is the Arabic definite article, and the word *shkhina* is the Hebrew word for divine spirit.

A large number of Judeo-Arabic works (philosophy, Bible translation, biblical commentary, and more) are currently being made available on the Internet (for research purposes). However, most Arabic speakers are unfamiliar with the Hebrew script, let alone the way it is used to render Judeo-Arabic. Our main goal in this work is to allow Arabic readers, who are not familiar with the Hebrew script, to nevertheless read and understand these Judeo-Arabic texts. We divide this task into three subtasks:

1. Identifying the language-switching points, also known as code switching.
2. Transliteration of Judeo-Arabic Hebrew letters into Arabic letters.
3. Arabic error correction: post-processing the Arabic-script text to resolve the remaining ambiguities and completing some missing characters.

In this paper, we propose automated techniques in each of these three areas.

Code switching is the act of changing language while speaking or writing, as often done by bilinguals [29]. In our case, the cross-language inflections, in addition to the rich morphology of all the relevant languages, the task of identifying code switching turns out to be non-trivial. We use a sequential classifier that works on the word level, considering both character-level and word-level features calculated for the surrounding words. In this paper, we focus on the Judeo-Arabic–Hebrew pair. The classifier is supervised by a relatively large set of Judeo-Arabic sentences, extracted from various sources, in which Hebrew words have been marked accordingly.

Transliteration is the process of converting a text from one script (the input script) into another (the target script). Transliteration is of course much easier than translation; in the latter the target text must convey the same meaning of the input text using words in a different language. All the same, we model the transliteration process using the same noisy-channel approach that is used for statistical machine translation. We employ a phrase-based statistical translation system [19] trained on the character level. The phrase table is generated using bilingual parallel texts of Judeo-Arabic words aligned with their Arabic renderings. To model the Arabic language, we use a large corpus of running Modern Standard Arabic (MSA) text, and train a character-level language model.

As mentioned above, Judeo-Arabic is a set of dialects, each used by the local Jewish community in some of the Arab countries. Some texts are similar to MSA, which is widely used today in formal settings, while other texts are more similar to local Muslim dialects. We focus, for now, on the Judeo-Arabic version that is similar to MSA more than on the colloquial versions.

We proceed as follows: Section 2 cites some related work. Our contributions are described in the following sections:

1. Section 3 proposes a methodology for finding the switching spots between Hebrew and Judeo-Arabic, both of which use the same Hebrew script.
2. Section 4 provides an automatic Arabic transliteration tool for Judeo-Arabic for the first time.
3. Section 5 provides preliminary results of using Bidirectional [10], Long Short-Term Memory Recurrent Neural Network [8,14] for post-processing the output Arabic transliteration to correct some common errors.

Some conclusions are suggested in the final section.

2 Related Work

To the best of our knowledge, this is the first work to deal with code switching and transliteration of Judeo-Arabic. There are several relevant works about code switching involving Arabic. Both [5] and [6] deal with code switching between MSA and colloquial Arabic on the word level. Similar to our work, they use a language model for predicting the label of every word in context, which can be either MSA or colloquial. For words that do not exist in the model, they use an Arabic morphological analyzer to determine its language. In a recent work [27], there is an attempt to identify Arabic words within a noisy Arabizi text, that is, an Arabic chat alphabet, rendered in Roman script. In particular, this work deals with Moroccan Arabic.

Similar to Judeo-Arabic, rendering Arabic from Arabizi text is an ambiguous process. In [3], followed by [7], Arabizi is automatically transliterated into Arabic. As a first step, the authors identify Arabizi in code-mixed texts using a sequential classifier, trained over some character-level, as well as word-level, features. Then, the identified Arabizi words are transliterated into Arabic using statistical machine translation that works on the character level, combined with some dictionary-related and morphological processes. Both papers deal with microblogs extracted from Twitter and manually preprocessed accordingly. Both report an accuracy above 98% for the code-switch task and above 83% for the transliteration task. It seems that transliterating Arabizi into Arabic is a more difficult task than Judeo-Arabic into Arabic, as the variability of writing Arabizi when referring to a specific Arabic word is larger than with Judeo-Arabic. However, this variability is affected by the colloquial language that is usually used in microblogging. Judeo-Arabic texts that are more affected by the colloquial language than the texts we are using in this work, increase the level of ambiguity presented in this work and may introduce some additional challenges. Working with such texts is one of our plans for future investigation.

There are many works that deal with the transliteration of names from one language into their phonetic equivalents in another script. In [26], names written in Arabic, but originated in languages other than Arabic (e.g., *Wall Street*), are transliterated back into English (also known as the back-transliteration task). Obviously this is not a trivial task, since Arabic lacks short vowels, which are needed for reconstruction, and some Arabic letters may have multiple renderings in Roman script. Given an Arabic string that represents a name, they generate a lattice of all possible phonetic sequences in English and then find the best sequence paths using probabilistic models, which they learn from relatively small manually created resources, such as a pronunciation dictionary and bilingual parallel corpora of English names aligned with their corresponding Arabic renderings. This technique was previously applied to Japanese [17]. The limitation of this technique is in the resources they use, which are not available for every language pair. A parallel corpus of personal names, written in both Arabic

and English, is used in [21] for transliterating from Arabic to English. As previously mentioned, the author uses a phrase-based statistical translation system for learning how to transliterate personal names, which achieves an accuracy of 43%.

There are additional works on names transliteration: In [4], a parallel corpus of Arabic-English transliterated pairs, which were obtained automatically following a process called *transliteration mining* (e.g., [24,15,2]), is employed. They train a statistical translation system on the extracted parallel data and use it on unseen (out-of-vocabulary) words to improve a word-based translation system. In [13], a method for transliterating names is introduced in the context of statistical machine translation. They work on the Arabic-English language pair. The transliteration process is performed by looking for the input Arabic name in a large corpus of English words (using an index). For every candidate they calculate a similarity score, based on consonant-matching rules, which they manually developed. Their technique shows a major improvement in an end-to-end translation system, especially for rare names.

In [9] a naïve-Bayes classification approach is taken for the problem of identifying foreign (borrowed) words and transliterated names in Hebrew. They build two models: one to generate an original Hebrew word and another one to generate foreign words written in Hebrew. For training they use a noisy dataset, which they produce automatically, containing English words with all possible Hebrew (possibly incorrect) renderings. They achieve 80% precision and 82% recall running on a set of about 4K unique words.

In [16], a method for transliterating Hebrew words into English is introduced. As a first step, they focus on identifying the terms to be transliterated, using a supervised classifier that considers the part-of-speech tags of the word in focus and of the preceding word. The tags are obtained automatically using a probabilistic tagger, and the positive examples for the classifier are extracted in a semi-supervised fashion, using some manually created rules that capture words that do not originate in Hebrew. For transliteration they employ a statistical translation system, which was trained over a parallel corpus that they extracted automatically from Wikipedia. They report 38% error reduction when compared to a baseline algorithm.

Our work applies some of the techniques we described above for the special case of Judeo-Arabic. Since we already have a parallel transliteration corpus at hand, we use it for training a statistical translation system that works on the character level.

3 Code Switching

For language-switch detection, we built a supervised framework to train a classifier that predicts the language of every word in the order written. To train our classifier, we needed a corpus of Judeo-Arabic texts, annotated for language boundaries. Fortunately, the Friedberg Jewish Manuscript Society recently released a corpus of Judeo-Arabic texts, containing a relatively large number of

books and essays, corresponding to about 4M words. What's more, as part of the digitization process, the words were manually annotated for language (Arabic, Hebrew/Aramaic).

We work on the word level; that is, every word is considered as an instance for the classifier, formatted as a feature vector. Naturally, our classifier works sequentially; in that sense, word instances are predicted one-by-one, potentially using the predicted labels of the previous words as features for classifying the following word. In this paper, a word must be annotated with exactly one label out of the three options: Judeo-Arabic, Hebrew, and Other. The latter is used for punctuation marks and is determined automatically using some simple text-search queries. We believe that some additional labels, such as one for indicating a name, could contribute to overall performance, since names may have a different distribution of the letters than the two languages in focus. Furthermore, we noticed that annotators usually marked named entities as Hebrew, as in most cases the names are originated in Hebrew rather than in Arabic; we believe that this situation confuses our models. In a recent code-switching shared task [25], a special label for names was used along with labels to indicate ambiguous words and mixed languages. Recognizing occurrences of names within Judeo-Arabic texts is definitely a natural direction for future investigation.

3.1 Features

We use features from a window of $\pm n$ words before and after the word of interest. We report on the results of using different window sizes. As mentioned above, for every word, we also consider as features the predicted labels of the n prior words. We use three main groups of features.

For every instance, the following classes of features are extracted from all the surrounding words as well as the word in focus:

Word level: This is the surface-form word, as it appears in the text.

Intra-word level: Both Judeo-Arabic and Hebrew are morphologically rich languages, demonstrating a complicated inflectional morphology. Usually, inflections are expressed by adding some word prefixes and suffixes. Therefore, to capture repeating suffixes and prefixes that may help identifying the language, we consider as features substrings of 1–3 prefix and suffix letters of the words.

We use three additional features for capturing the level of uncertainty of seeing the sequence of characters that form the specific word. This is done by employing a 3-mer[1] character-based language model, trained over a monolingual corpus in each language. Then, the two language models, one for each language, are applied on the word to calculate two log-probability values. We bin those values into 10 discrete values that are used as the features' values. The third feature is a boolean value indicating which of the two models return a lower

[1] We use the term "k-mer" for character k-grams, in contradistinction to word n-grams.

log probability. We train the models on related texts. For Hebrew, we use the Bible to support the classical nature of the language that is usually embedded in Judeo-Arabic texts. For the Judeo-Arabic model, we use a pair of books: (1) *Kitab al Khazari (The Kuzari)*, a medieval philosophical treatise composed by Judah Halevi in Andalusia in around 1140; and (2) *Dalalat al-Ha'Irin (The Guide for the Perplexed)*, a philosophical work composed by Maimonides in Egypt, in about 1190. The Hebrew corpus contains about 5M characters, while the Arabic one contains about 20M characters.

Inter-word level: We use 3-gram word-level language models, trained over large corpora in each of the languages. We apply the models to the focus word, considering it to be the last in a sequence of three words (with the two previous words) and calculate log probabilities. The values are binned into 10 discrete values, which are then used as the features' values. An additional boolean feature is used, indicating which of the two models return a lower log probability.

3.2 Experimental Approach and Results

We use the *Guide* as a corpus for training. Then, we test the system on a different book, the *Kuzari*. We train on one book and test on a different one in order to relax the effect of vocabulary repetition. Table 1 presents the label distributions for both books.

Table 1: Label distribution for the *Guide* and the *Kuzari*

	Guide	*Kuzari*
Word count	128K	23 K
Hebrew label	29K	3.5K
Judeo-Arabic label	90K	17 K
Other label	9K	2.5K

We train our classifier with SVM [1], using a degree-2 polynomial kernel. We use different window sizes for extracting features and compare our system's performance with a maximum-likelihood baseline algorithm (MLE), which for every word simply assigns the label with the highest probability as was seen in the corpus. We use WEKA [12] as a platform for the following experiments. The results are presented in Table 2. Clearly, our system significantly outperforms the baseline algorithm. The best accuracy and Hebrew precision are achieved when using a narrow window of ± 1; with larger windows the data becomes too sparse. The recall for Hebrew is slightly better when using a ± 2 window; however, the drop in precision is more prominent. The F-measure of the ± 1 window is 94.93, while for the ± 2 window is only 94.03.

Table 2: Some evaluation results.

	Accuracy	Hebrew		Judeo-Arabic	
		Precision	Recall	Precision	Recall
Baseline	68.38	36.30	88.64	90.40	75.86
Classifier ±1	**98.19**	**91.92**	97.94	99.46	**98.02**
Classifier ±2	97.80	89.61	**98.46**	**99.58**	97.57
Classifier ±3	97.68	89.19	98.24	99.52	97.46

4 Transliteration

As implied above, the transliteration into Arabic cannot be done deterministically. For some Hebrew letters there is no one-to-one mapping into Arabic; the correct one may be determined using the context it is mentioned in. Therefore, we imitate the transliteration process with an automatic translation one, performed on the character level rather than on the word level. We implement our system with Moses [18], an open-source instance of a phrase-based statistical translation system [19], and treat each character as a word and each word as a complete sentence. In particular, the "phrase table" (henceforth, not adorned with scare quotes) is composed of entries of character sequences in Hebrew mapped to their corresponding Arabic ones, and the language model is trained on sequences of characters of running Arabic texts. (We note that reordering is unnecessary for this particular task, as we assume that the Hebrew letters and the transliterated Arabic sequences tend to be generated monotonically.) In the following subsections, we describe the data sources and models we build, and the set of experiments we perform for evaluation.

4.1 Data Sources and Models

We generate the phrase table from pairs of Judeo-Arabic words, one in Hebrew script and one in transliterated Arabic form. The parallel words are extracted from the original Judeo-Arabic version of the *Kuzari*, alongside its Arabic translation, prepared by Nabih Bashir [11]. This is the only Judeo-Arabic resource we could find that was manually transliterated into Arabic and is available digitally.[2] The Judeo-Arabic version of this book was taken from the Friedberg Jewish Manuscript Society.[3] In the original text, Judah Halevi brings citations from various sources, mainly in Hebrew. Those citations are usually translated into Arabic by Bashir in his translated edition. We manually removed those translated pairs from parts of the parallel texts, resulting in a corpus of 8,560 parallel words, corresponding to 36K parallel characters. Each pair of words is

[2] We gratefully thank Nabih Bashir for providing us with a digital copy of the book.
[3] http://www.jewishmanuscripts.org

placed in its own line and spaces are inserted between the characters so that Moses will treat each character as an individual translation constituent. Figure 1 shows a sample of the generated corpus.

סילת	س ئ ل ت
עמא	ع م ا
ענדי	ع ن د ي
מן	م ن
אלאחתגאג	ا ل ا ح ت ج ا ج
עלי	ع ل ى
מכ'אלפינא	م خ ا ل ف ي ن ا
מן	م ن
אלפלאספה	ا ل ف ل ا س ف ة
ואהל	و أ ه ل
אלאדיאן	ا ل أ د ي ا ن
ת'ם	ث م
עלי	ع ل ى

Fig. 1: A sample of the parallel corpus.

For the language model, we use a large portion of Arabic Gigaword (4th ed.) [23] to train a 3-mer model. Arabic Gigaword is a large archive of newswire text articles, published by various media outlets. Since we are after modeling the most probable Arabic character sequences, we are ignoring the effect of using texts from a different domain.

We use a small part of the parallel corpus for tuning the weights of the models, using minimum error weight training [22].

After training the system, we use it to transliterate Judeo-Arabic sentences, written in the original Hebrew script, into Arabic. To do that, we first break the sentence into words, using some trivial word-boundary indications, and then we run each word through the translation system, after we insert spaces between the characters. The decoder of the translation system returns the best sequence of Arabic letters, which are then glued into words by removing the redundant spaces. Words are then pasted together to form the complete Arabic output sentence. Figure 2 summarizes the flow of the system.

4.2 Experimental Approach and Results

To evaluate our transliteration system, we set aside 500 words out of the parallel corpus we put together as described in the previous section, and use them as a test set. In addition, we use 550 pairs of words out of the same parallel corpus for tuning. This left us with 7,558 parallel words for building the phrase table. Given the 500 input words, our system was able to predict 96.9% of the letters

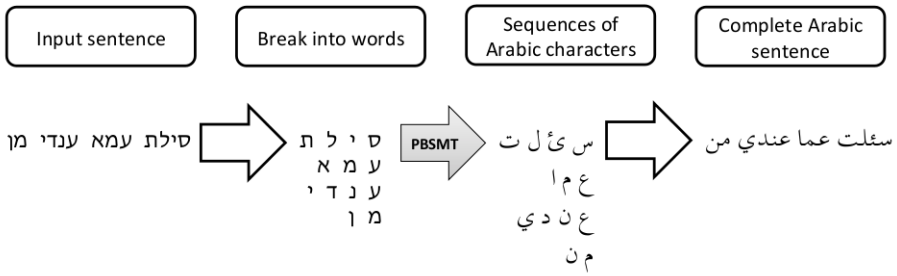

Fig. 2: The complete flow of the transliteration system, as modeled by phrase-based statistical machine translation (PBSMT).

correctly. However, it turns out that most Judeo-Arabic Hebrew characters can be transliterated deterministically into Arabic, especially when a character has a one-to-one mapping; hence, a simple baseline algorithm with some deterministic rules is able to correctly transliterate 93.4% of the test set, only 3.5% less than our system's accuracy. We performed a qualitative evaluation; Figure 3 summarizes the common mistakes of both systems. It is clear that our system learned how to deal with most of the common ambiguities; however, it still confuses about whether to place *hamza* (the glottal stop) at the end of a word or not, and it still cannot properly predicts when to use the letter ه (*ha*) or ة (*ta marbuta*) at the end of a word. Furthermore, both system confuses about when and where to place a *shadda* (gemination).

ת ← ت, ث	א ← ا, اء
ד ← د, ذ	ה ← ه, ة
א ← ا, اء	
ג ← ج, غ	
א ← إ, أ, ا, ؤ	
ה ← ه, ة	
י ← ئ, ي, ى	

Fig. 3: Common mistakes made by the baseline (on the left) vis-à-vis our transliteration system (on the right). Each entry represents one common mistake; the left part of the entry represents the Hebrew letter in the original Judeo-Arabic text, and the right part represents a list of possible Arabic renderings, which the algorithm confuses between.

Furthermore, to remove author-specific effects, we evaluated our transliteration system on a different resource. We train the system with the same Kuzari parallel corpus as described above, and test it on the seventh section of the *Book of Beliefs and Opinions* written in 933 C.E. by Sa'adia Gaon. The Arabic version of this treatise was also created by Nabih Bashir. The test-set contains 50 words, extracted in a similar way to the one described above. The same baseline algorithm was able to predict merely 88.7% of the letters, while our system was able to improve on this, correctly transliterating 91.6% of the test-set letters. Table 3 summarizes the results of the two experiments.

We noticed that in some cases when the decoder concludes with the wrong sequence, the correct one exists as a second or third best choice. Therefore, we use a 3-gram (word-based) Arabic language model, trained on a large portion of Arabic Gigaword, to re-rank the five best sequences as returned by the decoder. With re-ranking, we were able to significantly improve on the 91.6% accuracy on Sa'adia Gaon's book, correctly transliterating 96.2% of the letters. This is an absolute improvement of 4.6%.

Table 3: Evaluation results: the *Kuzari* and the *Book of Beliefs and Opinions*.

	Kuzari	**Beliefs**
Baseline	93.4	88.7
Our system	96.9	91.6

5 Arabic Error Correction

The results of the previous step left us with some errors, which may cause some inconvenience for a native Arabic reader. We mention some of those errors in Figure 3. In this section we suggest an elegant way for post-processing the text to correct the such errors automatically. This part of the work is still in progress; some encouraging results are presented.

We build an individual predictor for each correction task and train it over a corpus of correct MSA texts. We experiment initially with two correction tasks: (1) *hamza* (the glottal stop) reconstruction at the end of a word (as in مَسَاء, "evening"); and (2) deciding on the correct Arabic letter corresponding to י (Hebrew *yod*) at the end of a word (either ي or ى). To make sure the training data we use is written correctly, we use the Arabic Treebank 1 Version 2 (ATB1v2.0) [20], including 734 stories from the Agency France Press. We only use part of the corpus, corresponding to about 120K words. We use a Bidirectional Long Short-Term Memory (BLSTM) Recurrent Neural Network (RNN) with 3 layers, and train a separate network for each task. In fact, for the second task, we use

two networks, one for predicting ي (*ya*) and one for predicting ى (dagger *alif*). Our implementation is based on CURRENNT [28]. Some preliminary results for these two tasks are presented in Table 4, where we use a split of 90% of the corpus for training and 10% for testing. We plan to investigate this topic further and apply similar algorithms on the results of the previous transliteration step.

Table 4: Error correction with BLSTM RNN. Results for predicting the existence of *hamza* at the end of a word, and for deciding on the correct Arabic letter between ي (*ya*) and ى (dagger *alif*), occurring at the end of a word.

Network	**Precision**	**Recall**	**F-Measure**
hamza	96	93	94
ي	97	94	96
ى	98	97	97

6 Conclusion

This work shows some techniques for processing Judeo-Arabic text, a special case of Arabic – written in Hebrew letters. Our goal is to allow Arabic speakers and readers to read Judeo-Arabic texts in the Arabic script rather than in the original, ambiguous, Hebrew one. Inspired by previous works, we employ a phrase-based statistical translation system that works on the character level, and train it using a parallel corpus of Judeo-Arabic words mapped to their corresponding Arabic renderings. This method was found to show encouraging results (e.g., 96.2% accuracy as opposed to 91.6% obtained by a baseline algorithm) when trained on one book and then applied to another book. However, in both books the authors use Judeo-Arabic that is pretty similar to MSA. One can imagine that, when working with a more colloquial version of Judeo-Arabic, the transliteration into Arabic will prove more challenging.

We have seen that reranking the top few results generated by the translation system improves the overall accuracy. The reranking is done by combining information about word contexts in addition to the character-level information considered by the decoder of the translation system.

As another step towards improved results, one may employ error-correction algorithms applied to the Arabic output, such as completing *hamza* at the end of a word – something that our current transliteration algorithm tends to miss. We show some preliminary, encouraging results; we plan on pursuing this direction further.

As mentioned earlier, not all words in Judeo-Arabic texts should be transliterated in the same fashion. Usually there are some words and citations in Hebrew

and Aramaic, sometimes modified using Arabic morphological rules. To identify the Judeo-Arabic words that are originated in Arabic, we employ a supervised sequential classifier that predicts the language of every word in order. We combine word- and character-level features, learned from an annotated corpus and achieve a major improvement over a baseline algorithm.

Identifying named entities, which are currently annotated as Hebrew, may also help improve the prediction.

References

[1] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[2] Kareem Darwish. Transliteration mining with phonetic conflation and iterative training. In *Proceedings of the 2010 Named Entities Workshop*, pages 53–56. Association for Computational Linguistics, 2010.

[3] Kareem Darwish. Arabizi detection and conversion to Arabic. *arXiv preprint arXiv:1306.6755*, 2013.

[4] Nadir Durrani, Hieu Hoang, Philipp Koehn, and Hassan Sajjad. Integrating an unsupervised transliteration model into statistical machine translation. *EACL 2014*, page 148, 2014.

[5] Heba Elfardy, Mohamed Al-Badrashiny, and Mona Diab. Code switch point detection in Arabic. In *Natural Language Processing and Information Systems*, pages 412–416. Springer, 2013.

[6] Heba Elfardy, Mohamed Al-Badrashiny, and Mona Diab. AIDA: Identifying code switching in informal Arabic text. *EMNLP 2014*, page 94, 2014.

[7] Ramy Eskander, Mohamed Al-Badrashiny, Nizar Habash, and Owen Rambow. Foreign words and the automatic processing of Arabic social media text written in Roman script. *EMNLP 2014*, page 1, 2014.

[8] Felix Gers. Long short-term memory in recurrent neural networks. *Unpublished PhD dissertation, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland*, 2001.

[9] Yoav Goldberg and Michael Elhadad. Identification of transliterated foreign words in Hebrew script. In *Computational Linguistics and Intelligent Text Processing*, pages 466–477. Springer, 2008.

[10] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[11] Yehuda Halevi. *The Kuzari – In Defense of the Despised Faith; transliterated, translated and annotated by Nabih Bashir.* Al-Kamel Verlag, Beirut, 2012.

[12] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[13] Ulf Hermjakob, Kevin Knight, and Hal Daumé III. Name translation in statistical machine translation—learning when to transliterate. In *ACL*, pages 389–397, 2008.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Sittichai Jiampojamarn, Kenneth Dwyer, Shane Bergsma, Aditya Bhargava, Qing Dou, Mi-Young Kim, and Grzegorz Kondrak. Transliteration generation and mining with limited training resources. In *Proceedings of the 2010 Named Entities Workshop*, pages 39–47. Association for Computational Linguistics, 2010.

[16] Amit Kirschenbaum and Shuly Wintner. Lightly supervised transliteration for machine translation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 433–441. Association for Computational Linguistics, 2009.

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

[18] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Interactive Poster and Demonstration Sessions of the 45th Annual Meeting of the ACL (ACL '07)*, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[19] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

[20] Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, pages 102–109, 2004.

[21] David Matthews. Machine transliteration of proper names. Master's thesis, University of Edinburgh, Edinburgh, UK, 2007.

[22] Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 160–167. Association for Computational Linguistics, 2003.

[23] Robert Parker et al. Arabic Gigaword Fourth Edition LDC2009T30. *Linguistic Data Consortium (LDC), Philadelphia*, 2009.

[24] Tarek Sherif and Grzegorz Kondrak. Bootstrapping a stochastic transducer for Arabic-English transliteration extraction. In *Annual Meeting—Association for Computational Linguistics*, page 864, 2007.

[25] Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Gohneim, Abdelati Hawwari, Fahad AlGhamdi,

Julia Hirschberg, Alison Chang, et al. Overview for the first shared task on language identification in code-switched data. *EMNLP 2014*, page 62, 2014.

[26] Bonnie Glover Stalls and Kevin Knight. Translating names and technical terms in Arabic text. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 34–41. Association for Computational Linguistics, 1998.

[27] Clare Voss, Stephen Tratz, Jamal Laoudi, and Douglas Briesch. Finding romanized Arabic dialect in code-mixed tweets. *LREC 2014*, 2014.

[28] Felix Weninger, Johannes Bergmann, and Björn Schuller. Introducing CURRENNT–the Munich open-source CUDA RecurREnt Neural Network toolkit. *Journal of Machine Learning Research*, 15, 2014.

[29] Donald Winford. *Code Switching: Linguistic Aspects*, chapter 5, pages 126–167. Blackwell Publishing, Malden, MA, 2003.