Transliteration as Machine Translation*

Theresa Breiner
tbreiner@seas.upenn.edu

Abstract

Machine transliteration is a difficult task especially when working with low-resource languages. Inspired by the work of Irvine, Callison-Burch, and Klementiev, this paper documents my experiments transliterating names from 67 languages, using data sets of various sizes, into English. I begin by explaining the motivatation for the task with my domain knowledge of linguistics, and reviewing Irvine et al.'s previous work. I then analyze the use of different evaluation metrics and graphs of learning curves, and I compare models trained on different combinations of data. My findings indicate that supplementing training data with data from similar scripts and including larger language models are two ways to use larger-resource language data sets to boost performance of transliteration from lower-resource languages into English.

1 Introduction

As technologies for machine translation continue to improve and reach levels where large amounts of text data can be reliably translated regardless of source or target language, researchers are turning more and more towards techniques to perfect the less significant but persisting errors in the systems. While the huge amounts of text resources on the internet allow translation models to be trained that can handle more and more special cases and unknown words, one subset of words continues to be elusive, a subset for which there will never be enough data to learn a translation model. Named entities, such as people, organizations, and even locations make up what is called a productive lexical set in that new words are always entering the language. Therefore, beyond the sheer numbers of name possibilities that exist, a translation model will never be able to learn 100% coverage of this set, as it is guaranteed by the time the model has finished training there will already be new names in existence. An approach to address this problem is to incorporate a transliteration model.

Transliteration, however, presents its own set of challenges, particularly in regards to lack of verified training data. Irvine, Callison-Burch, and Klementiev (Irvine et al., 2010) experimented with name data mined from Wikipedia to explore possible methods for efficiently producing transliteration models between any two language pairs, despite potential dearth of training data. In this paper I attempt to recreate some of their experiments, to some extent, and focus on two analyses: which evaluation metric might be best to use in future research, and how to improve performance for languages with very low numbers of training instances.

The rest of the paper is organized as follows: in Section 2 I provide background on the task of transliteration, its challenges, some previous work and the motivation towards the approach of Irvine et al. In Section 3 I describe the experimentation of Irvine et al. and the tools they used. In Section 4 I discuss my own data, methods and results. I offer some final analysis, speculation and future ideas in Section 5, and conclude in Section 6.

*This paper serves as the final project for the author's Spring 2016 independent study research under the supervision of Dr. Chris Callison-Burch at the University of Pennsylvania.

2 Background

Transliteration, or transforming text written in one script into another script, is itself a difficult task for many reasons. Different scripts have different structures. Some languages, like English, are written with alphabets, where each grapheme or symbol represents one phone or sound in the language such as "k" or "t". Some, like Japanese, are syllabaries, where each grapheme represents a full consonant-vowel syllable like "ka". Even within these groups, there are situations that can make transliteration more difficult.

For example, the written "ch" in English does not represent two sounds, but one, although not always the same sound, as seen in "chalk" versus "machine". In the same example of "chalk," the "l" doesn't make any sound, although it may affect the pronunciation of the vowel depending on where the speaker grew up. Another type of writing system is semano-phonetic, such as used in Chinese, where symbols carry information about both pronunciation and meaning. Here, there can be variation in pronunciation of the same grapheme, and there are also thousands more graphemes than in the other types of scripts.

Beyond structural differences in scripts, an additional challenge is that most languages contain sounds that do not exist in other languages, so there may not be a direct correlation of a sound from one script to another. In this case, arbitrary choices about how to express the foreign sound in the target language may be made, and there may be more than one way to transliterate the same source token, such as "hanukkah" and "chanukah." Futhermore, some tokens found in a text may themselves be loan words transliterated from another source and may not follow the language's typical phonetic patterns, such as "feng shui."

A transliteration model may attempt to learn the mappings indicating which graphemes correspond to the same sounds or which graphemes are generally chosen to represent foreign phonemes, but focusing only on phonetic similarity will not yield best results as it does not take into account real usage in the target language (Oh and Isahara, 2006). Although it may be true that the sound represented by a grapheme in the source language is most accurately represented by a particular grapheme in the target language, a transliteration using that rule may not produce a word that is most likely to be seen in the target language.

Many researchers have applied the noisy channel model, using Bayes' Rule to find a transliteration that is both an accurate representation of the source text phonetically and a natural fit to the target script (Virga and Khudanpur, 2003). However, in many cases this does not sufficiently address the concerns mentioned above, such as loan words not conforming to language patterns, and problems going from a syllabary script to a semano-phonetic one like Chinese (Haizhou et al., 2004). Using a log-linear statistical machine translation (SMT) approach allows more flexibility in training because additional features and language models beyond those in the traditional model can be used (Och and Ney, 2002). Irvine, Callison-Burch, and Klementiev (Irvine et al., 2010) combined this idea with their goal of discovering a method for producing transliteration models between any two language pairs to experiment using the Joshua decoder on sets of name data mined from Wikipedia.

3 Previous Work

The paper entitled "Transliterating From All Languages" by Irvine et al. is the foundation for this study. The authors diverge from previous literature that tends to focus on language-specific tools and research and explore possibilities of transliteration pipelines that are both simple to use and also allow models to be trained on any language pairs as long as there is some parallel data available. They use the log-linear SMT approach rather than a noisy channel model, which corresponds to the transliteration task as follows: a grapheme sequence mapping probability serves as a phrase translation model would in SMT; a grapheme substitution model replaces lexical probability; and a grapheme-based language model provides information about the fit of the transliteration attempt in the target script. Unlike in usual SMT, there is no need for re-ordering in transliteration. Since an average script has a much lower count of graphemes than an average language does words, these models are able to support much larger n-gram sizes. Irvine et al. use 10-

gram grapheme sequences to build the target language models, which would yield counts that would be much too sparse if performing standard translation rather than transliteration.

3.1 Joshua

Irvine et al. take advantage of an existing SMT tool in their experiments. The Joshua decoder (Li et al., 2009) is an open-source, ready-to-run tool produced by Johns Hopkins University. It offers a fairly easy to use pipeline that gives the user control over the several phases: preprocessing the data, alignment, parsing, grammar extraction, language model building, tuning, testing, and analysis. Irvine et al. chose to use the Berkeley aligner (DeNero and Klein, 2007) to align the graphemes, which is supported by Joshua. They also chose to limit the synchronous context-free grammar that Joshua learns to be non-hierarchical, since there is no hidden structure such as parts of speech in transliteration, by restricting the number of non-terminals to 0. During the language model stage they include an English names language model built from names found in the English Gigaword Corpus from the Linguistic Data Consortium[1]. The tuning is done using Joshua's in-built MERT optimization, which uses a default objective function of the BLEU score, a standard evaluation metric for machine translation.

3.2 Data

The data used by Irvine et al. consists of sets of name pairs mined from Wikipedia. Using certain helpful starting resources such as Wikipedia pages listing people born in a certain year, they followed inter-language links provided on the site to build up lists of names of people as given on their Wikipedia page in English and the titles of the same translated article as given in all languages linked to the page. Besides English, they pulled 14 languages of interest from this data and focused on cleaning the pairs. In some cases, the name given on the English Wikipedia page does not exactly align with the foreign name, and therefore would not be a useful candidate for training. For example, the English page "Abbas I of Persia" links to a Russian translated page which glosses literally to "Abbas I the Great." A more extreme case is when the names are rearranged, such as "Anders Behring Breivik" appearing as "Breivik, Behring Anders" on the Avar-translated page. To attempt to correct these misaligned pairs, Irvine et al. mined a simple Romanization model from Wikipedia first and realigned the names, keeping the best-scoring aligned pairs and filtering out any pairs whose best alignment still scores below a threshold. Once valid data was gathered, it was preprocessed by adding spaces between each character, and a start of word and end of word symbol was also added. Then, the data could simply be fed into the Joshua pipeline as detailed above.

[1] https://catalog.ldc.upenn.edu/LDC2003T05

3.3 Findings

Irvine et al. saw that their system using the Joshua decoder pipeline, tweaked only slightly to build transliteration models, performed comparably to the systems submitted to the 2009 Named Entities Workshop shared task transliterating English to Russian and Hindi. They used the standard Levenshtein edit distance to evaluate models trained to transliterate different languages into English and found that the model for Russian to English showed the best results, probably due to being the largest data set. However, Arabic, which also had a large data set, did not perform as well as expected. Still, they plotted the learning curves of the models and saw that more training data does improve performance continually even with already large sets. They also saw that producing the 5 best outputs instead of just the top output increased performance by 20%, and producing the 10 best increased by another 10%, but no real benefit was given from examining beyond the 10 best outputs. In analyzing the errors that their systems made they pointed out that knowing the language of origin of the name may help in tough name transliterations.

4 Experimentation

Following Irvine et al.'s work, I used Joshua to train transliteration models from several languages into English. As they had mentioned using edit distance instead of the BLEU score for evaluation, I examined some different evaluation metrics from the NEWS 2015 Machine Transliteration Shared Task (Banchs et al., 2015) on the models' performance

and their learning curves. I experimented with different methods for working with the smallest data sets, including using models trained on another, larger language's data, models trained on combined data for all languages using a similar script, and models trained only on the small language's set but boosted with a larger English language model. I tend to simplify the experimentation process compared to Irvine et al.'s work to allow me to see at least surface level results for these various tasks. For example, I do not perform 10-fold cross validation to take averages for each language model, which means that my results may be somewhat skewed as my models may be overfitting training and tuning data. However, I am still able to get the overall picture for comparing models' performance.

4.1 Data

I used the wikipedia names dataset that Irvine et al. made publicly available via a link in their paper. It consisted of data from about 150 languages that I sifted through to gather only the 67 languages that used scripts notably different from English. They are written using 28 different scripts, which I sort into 8 language families plus one category "other." (see Table 1) I split each language's foreign and English-transliterated pairs into about 80% training, 10% tuning, and 10% testing sets after first shuffling the data within the language. In some cases where the data sets were larger than 20,000 pairs I limited the tuning and testing sets to 2,000 pairs each and allowed the training set to be larger than 80%. More detailed information about the sizes and families of the language sets can be found in Appendix A. It seems that compared to Irvine et al. I am incorporating much more data per language, and I am also examining more and smaller language sets.

I preprocessed the data before feeding it into Joshua by replacing spaces between words with underscores and then inserting spaces between every character. Unlike Irvine et al., I did not add any additional beginning- or end-of-word characters, although some of that information would be learned from the underscore serving as a word boundary. Also, some scripts such as Cyrillic use capitalization like English does which may not be filtered out during Joshua's preprocessing, thus potentially providing additional information to the models.

Script Family	**Languages**	
Arabic	Arabic	Egyptian Arabic
	Pashto	Uyghur
Bengali	Assamese	Bengali
Chinese	Gan	Wu
	Chinese	Cantonese
Cyrillic	Abkhazian	Avar
	Bashkir	Belarusian
	Classical Belarusian	Bulgarian
	Chechen	Chuvash
	Kazakh	Kyrgyz
	Meadow Mari	Macedonian
	Mongolian	Erzya
	Ossetian	Russian
	Rusyn	Sakha
	Serbian	Tajik
	Tatar	Ukranian
Devanagari	Bihari	Hindhi
	Marathi	Nepali
	Newar	Sanskrit
Georgian	Georgian	Mingrellian
Hebrew	Hebrew	Yiddish
Perso-Arabic	Sorani	Persian
	Kashmiri	Mazanderani
	Western Punjabi	Urdu
Other	Amharic	Azerbaijani
	Tibetan	Greek
	Armenian	Inuktitut
	Japanese	Khmer
	Kannada	Korean
	Lao	Malayalam
	Burmese	Oriya
	Punjabi	Sinhalese
	Tamil	Telugu
	Thai	

Table 1: Languages by Family

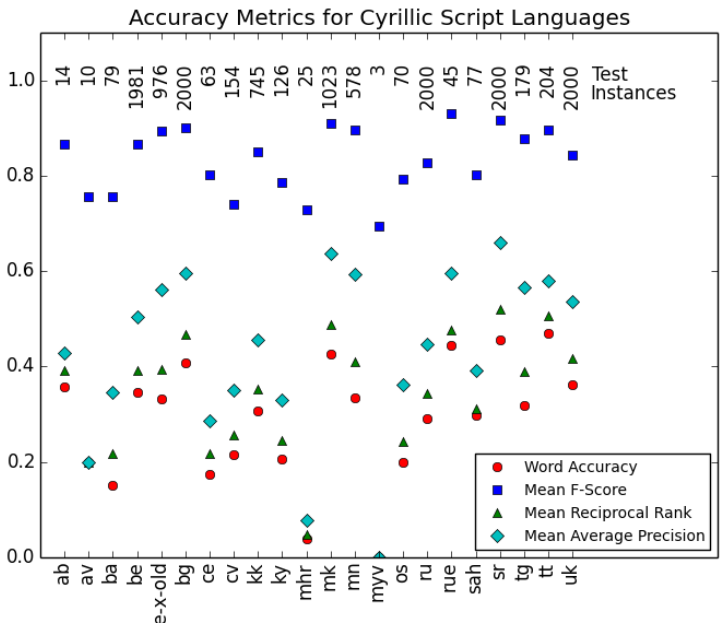

Figure 1: Four Evalauation Metrics on Cyrillic Languages

Just as was done in Irvine et al., I used the Berkeley aligner and the default MERT optimization during tuning. However, I allowed Joshua to build the target language model from the input corpus rather than using a pre-trained language model based on the English Gigaword corpus.

4.2 Initial Results and Evaluation Metrics

After training models on each of the 67 languages, I got the top 100 outputs for each name in the testing set to use for evaluation. Since Irvine et al. had shown that they didn't see much improvement in reference coverage after the top 10 best outputs, I did not feel that it was necessary to produce beyond 100 outputs per name.

I calculated 4 different evaluation metrics for each model following the NEWS 2015 shared task report (Banchs et al., 2015). Word accuracy is the percentage of test names for which the top 1 output was exactly correct. Mean F-Score is similar to the evaluation metric used by Irvine et al. as it incorporates the edit distance from the top output to the reference. It calculates the F-Score using the least common substring to calculate precision and recall, taken as .5 * (length(output) + length(reference) – editDistance(output to reference))[2]. Mean reciprocal rank finds the rank in the top 100 of the first correct output and averages 1 over the rank for all names tested. Mean average precision examines every correct output over the top 100 and allows higher ranked correct outputs to have more influence over the precision.

After graphing my results, I can see that the mean F-Score is the most forgiving metric, yielding the highest numbers across all language families. Mean average precision is generally second highest but groups closely with mean reciprocal rank and word accuracy, following the same pattern across most languages. Languages with larger data sets show mean average precision has more distance from the mean reciprocal rank and word accuracy, while on smaller data sets the three are often closer together. Overall, most models achieve a mean F-score above .8, with exceptions being the smaller data sets. The word accuracies tend to be in the .3-.4 range or lower with some such as the Bengali scripts, certain Devanagari languages and certain Cyrillic languages scoring over .4. These results, while coming from models trained on other languages transliterating into English, seem to fall in the same ranges

[2]See (Banchs et al., 2015) for detailed equations

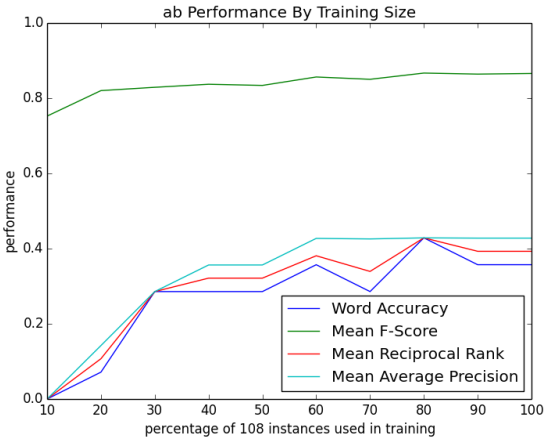

Figure 2: Learning Curve, Abkhazian (Cyrillic)

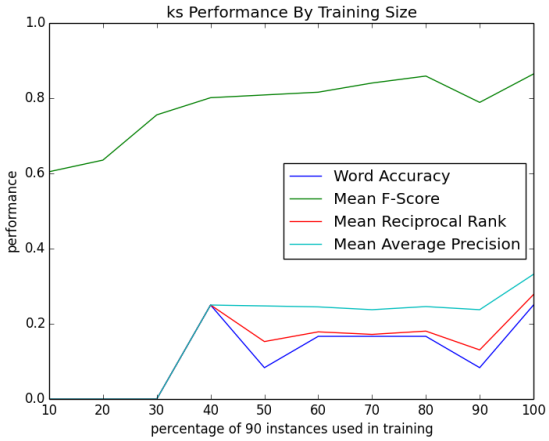

Figure 3: Learning Curve, Kashmiri (Perso-Arabic)

as the scores in Irvine et al.'s experiments from English into Russian and Hindi. An example graph of the Cyrillic script language family is given in Figure 1, and the others are available in Appendix B.

Producing the learning curves for each language follows the Irvine et al. paper in that generally, more data results in better evaluations. The data sets that are already very large, such as Russian, do not show vast improvement even above 10% of the training data, but still slowly improve with more data[3]. Certain unexpected dips in performance even as more training instances are added is likely due to overfitting to some of the new training data since I did not perform cross validation here, and many of the training sets are small to begin with. The most interesting piece to take from the results is that while the mean reciprocal rank and word accuracy do often show some erratic behavior over the course of the curve, the mean average precision and mean F-score are more stable, as can be seen in the curves for Abkhazian and Kashmiri (Figures 2 and 3). This may indicate that they are not as prone to being swayed by overfitting as the other metrics. Since the mean reciprocal rank and word accuracy tend to generally follow the same patterns as the mean average precision, however, in the rest of the experiments I will no longer show their results.

[3]Russian's curve as well as other select learning curves can be found in Appendix C, attached

4.3 Cross-Language Testing

Some of the top 7 languages by data set size (see Table 2) are members of language families containing low-resource language sets. To see if a model trained on a larger data set in the same script family but still a different language could produce better output on these lower-resource languages, I ran the model trained on each of the 7 largest sets on the testing data for the other languages in the family, and for some other languages as well for comparison. I ran the model trained on Arabic on every other language's testing data as well.

Interestingly, the Arabic model did not show higher accuracy on any of the other languages. However, I saw that in a several cases, despite being trained on a different language, the fact that it was the same or a similar script and also had vastly more training data allowed the system to produce higher accuracies compared to the models trained on the language's own, small data set. For example, the Chinese model produces higher mean F-Scores and mean average precision scores when tested on Gan, Wu, and Cantonese (see Figure 4). Table 3 shows the other small languages that saw improvement in testing using models trained only on separate but larger languages.

Even the model trained on Japanese was able to do fairly well on the Chinese script languages due to the crossover between the scripts, although it did not produce as good results as the models trained on those languages despite their small size.

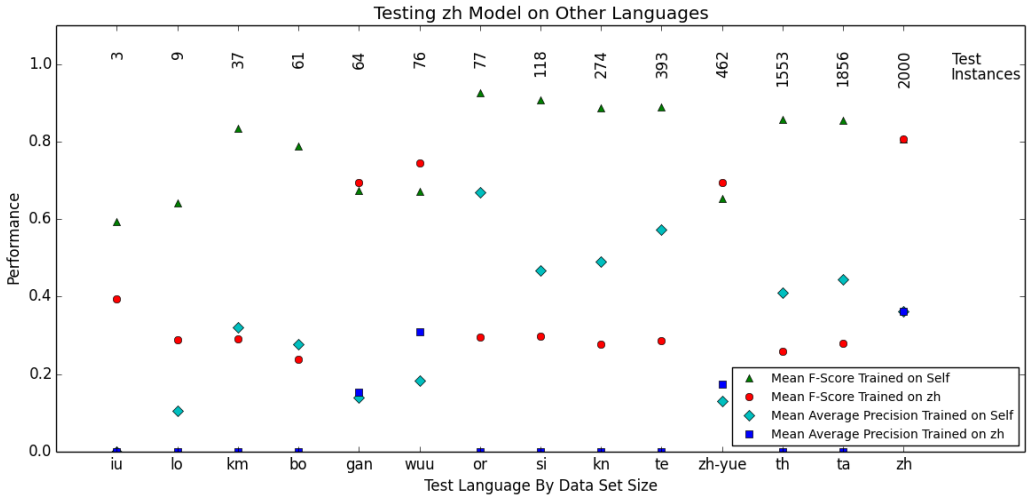

Figure 4: Model trained only on Chinese tested on other Languages

Language	Training Set Size	Script Family
Russian	225680	Cyrillic
Japanese	144481	Japanese
Farsi	88295	Perso-Arabic
Chinese	81439	Chinese
Ukranian	71054	Cyrillic
Arabic	55356	Arabic
Korean	52865	Korean

Table 2: Top Biggest Language Sets

Model	Test	Δ F-Score	Δ Avg Prec
Farsi	mzn	-.01	+.06
Russian	myv	+.13	+.66
	mhr	+.06	+.16
	sah	+.02	+.08
	ab	-.01	+.11
	ce	-.01	+.06
	ky	+0.0	+.02
	mn	-.02	+.02
Ukranian	rue	+0.0	+.04
Chinese	gan	+.02	+.01
	wuu	+.08	+.16
	zh-yue	+.04	+.04

Table 3: Cross-Test Improvements, Tests Ordered Smallest Langs to Largest

4.4 Multi-Language Training

4.4.1 Big Languages

Next, I tried training models on combinations of multiple languages. First, I tried training a "bigs" model on all of the top 7 biggest language sets to see if I could produce a one-size-fits-all system that would perform reasonably well transliterating data from several languages into English (see Table 2). Most of the time, it improved testing performance on the smaller languages that had been helped by the larger models seen in 4.3, but had worse performance on the other languages' data compared to the models trained specifically on those languages.

However, there were two interesting findings as can be seen in the first plot of Appendix E. First, performance on some of the larger testing sets, such as Arabic, Russian, Farsi and Ukrainian went up slightly in terms of mean average precision when using the big combined model. This could be due to the fact that Russian and Ukrainian are both written using Cyrillic, and although the Arabic and Perso-Arabic scripts are classified as different writing systems, they are extremely close. Despite some phonological differences in languages that use these two scripts which result in some slightly different graphemes and pronunciations, at least when translating into English they are similar enough that combining the training sets could be boosting performance.

The second finding is that the Korean and Japanese test results also improve very slightly in mean F-score using the bigger model, even though there is no other language in the top 7 biggest that uses a similar script so there should not be any more helpful training data here than when training only on the Korean or Japanese data. Admittedly, these differences are so small that they may be coincidences, which perhaps would be eradicated with a more rigorous, cross-validation-based methodology.

The one difference that could be improving the performance in this case however, and also could contribute to the slight improvement in the other four languages, is that the English language model being used with this model has been trained on all of the English names in all of the pairs from the over 700,000 training instances and therefore is presumably a much more robust English names language model. This realization motivates the next batch of experiments in Section 4.5.

4.4.2 Language Families

Since it was clear that more training data is helpful, and that even other languages' data can be helpful as long as it is written in the same script or one in the same family, I next trained models based on all training data from languages in the same family. The results overwhelmingly showed that data from the same family makes a difference. Even some of the largest-resource languages in a family can be boosted by some extra training data from other languages. For example, the model trained on all Perso-Arabic data was able to do better in terms of both mean F-score and mean average precision on Farsi than the original model, even though Farsi had more training data originally than the other four languages combined (see Figure 5).

The smaller languages saw nontrivial improvements in testing almost across the board. One exception was Uyghur whose mean F-score increased slightly with the help of all the other Arabic data but whose mean average precision decreased (see Figure 6)[4].

As the Cyrillic family model was unable to finish training before submission of this paper, the results are not included.

[4]We must keep in mind that there were only 29 test instances in this case.

4.5 Bigger Language Models

My next round of experiments was to take the 15 lowest-resource languages and retrain their models incorporating the English language model that had been built during the 7 big language model training in Section 4.4. This mimics Irvine et al.'s inclusion of the language model trained on the English Gigawords names, albeit at a smaller scale. Additionally, since Joshua built the language model based on all of the training pairs of names, there will be cases where two different languages contained training data corresponding to the same transliterated English name, which means that there would be repetitive data in the set that the language model was trained on.

Incorporating this larger language model had a slight improvement on testing several of the languages in terms of mean F-score, and more of an improvement in terms of mean average precision (see Figure 7). The three lowest resource sets of Inuktitut (iu), Erzya (myv) and Lao (lo) were not able to complete training models in this experiment and therefore no results are shown[5].

5 Analysis

5.1 Evaluation Metrics

I agree with Irvine et al. that mean F-score or a similar metric involving edit distance is an appropriate choice for transliteration tasks. However, taking into account that it was the most forgiving metric, and in these experiments it could only consider the top output of a system, I think it is a metric that is best suited for applications involving human readability. For example, if the goal is to incorporate the transliteration system into a machine translation tool for mass translating texts online for example, a system with high performance as evaluated by mean F-score should serve well.

However, if the application is less human-driven, such as a translation tool that involves machine search or for example co-reference resolution, then a system with a relatively higher mean average precision may be a better tool if those n-best outputs can

[5]When attempting to train these models, Joshua threw errors that seemed to indicate that the data sets were too small, although models on the same sets had successfully been trained when not including a pre-built language model in earlier experiments.

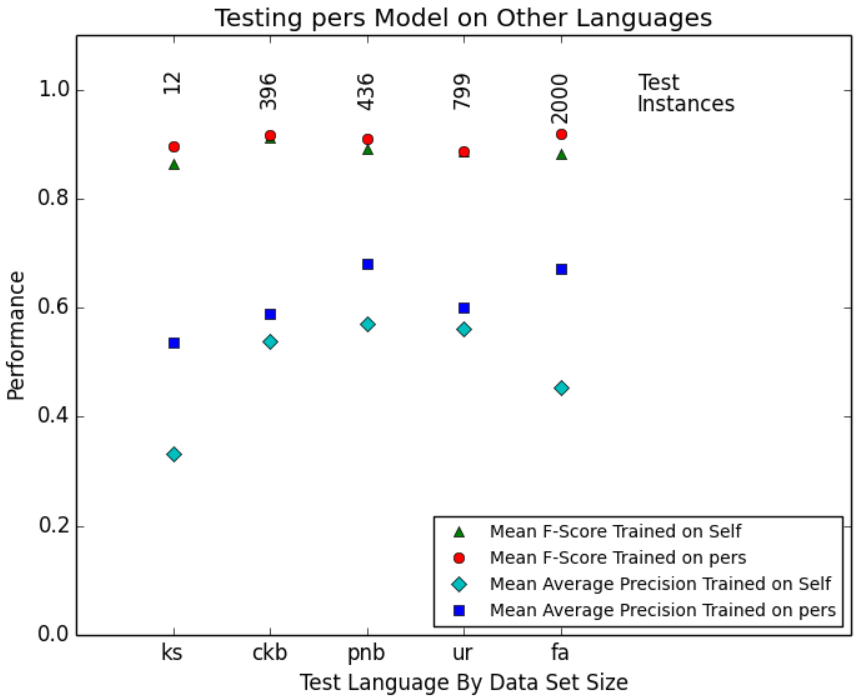

Figure 5: Model trained on whole Perso-Arabic Family

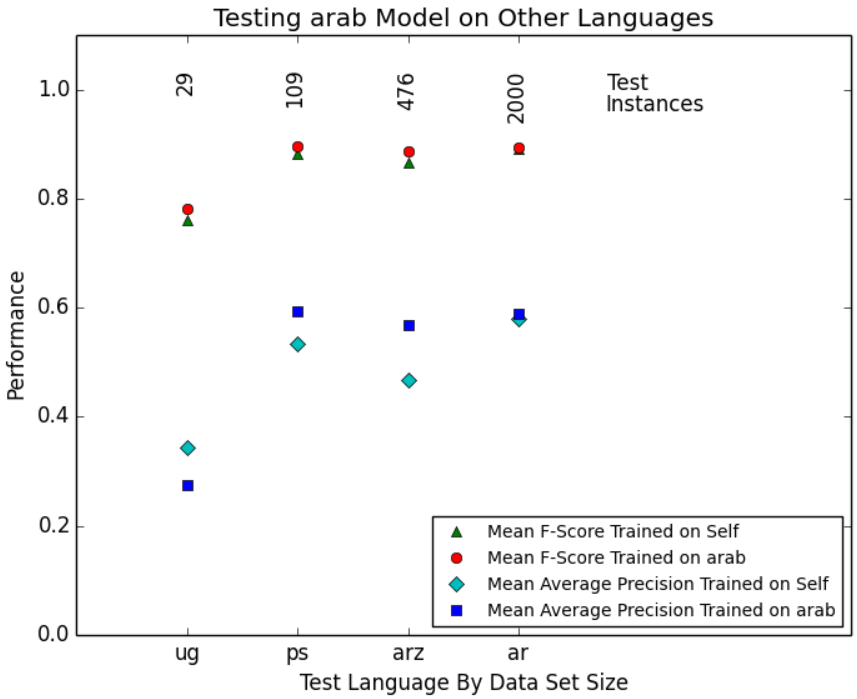

Figure 6: Model trained on whole Arabic Script Family

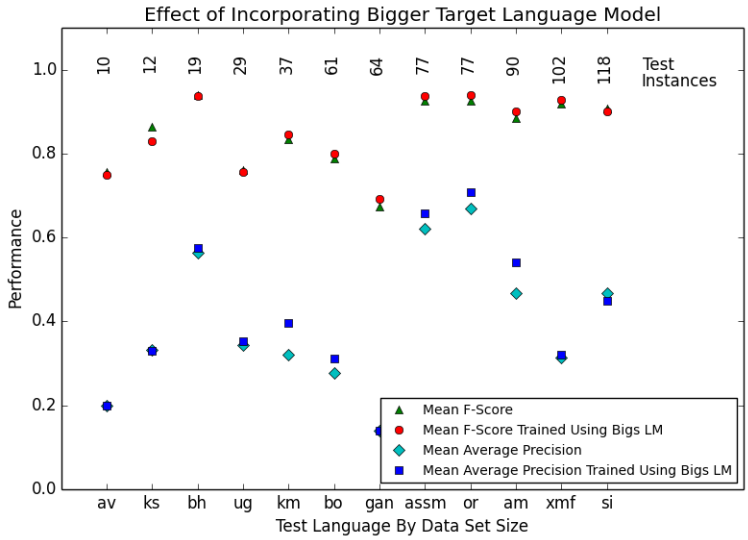

Figure 7: Effects of Including the "Bigs"-trained Language Model

be incorporated since the exact reference names will be found closer to the top of the output list.

As Irvine et al. mention, in future work ultimately one of these metrics should be chosen to use during tuning as the model is being built instead of the BLEU score.

5.2 Low-Resource Languages

While including a more robust target language model did show slight improvement in mean F-score and more improvement in mean average precision, the biggest results for improving transliteration to English from low-resource languages was the inclusion of more training data from other languages in the same family. Even if the languages are not mutually intelligible and may have differences in grammatical structure and pronunciation of characters, as is the case with Mandarin Chinese (zh) and Cantonese (zh-yue), my experiments show that the combination still helps with the transliteration of names, at least.

5.3 Limitations

One limitation of this research was that the Wikipedia names data only contains one correct transliteration reference for each training or testing pair. In reality, as mentioned in the background discussion in Section 2, for language pairs where there is a large difference in the phonological makeup there can often be more than one correct transliteration. In the case of names, a simple example is the name "Mohammed" transliterated from Arabic and other languages, which could correctly be spelled "Muhammad" as well as several other variations in English. The lack of multiple references, if possible, for each pair would result in lower word accuracy and mean F-score although would presumably have less of an effect on the two metrics that consider the n-best results.

Another limitation in the data was that despite Irvine et al. weeding out some misaligned pairs, there were still some less-than-perfect instances in the data that I used, such as the aforementioned "Breivik, Behring Anders" compared to "Anders Behring Breivik." There were also some cases where the name was already in English in both languages, I expect since some Wikipedia pages even in other scripts use the English name as the title of the page. In these experiments I did no further data cleaning to address these problems.

Lastly, in regards to time constraints, if time permitted then using cross validation in all aspects of these experiments would have given more reliable results.

5.4 Future Ideas

There are a few models that did not have time to finish training that may be worth examining in the

future. I would have liked to examine the results of a model trained on all of the Cyrillic scripts to see if the lower-resource languages in that family also saw improvements like the other families exhibited. I would also have liked to successfully train the three lowest-resource languages by incorporating bigger English language models.

I was also attempting to train a model on all of the data from all languages combined. However, I am doubtful that I would see any significant results that I didn't already glean from my completed experiments involving the 7 biggest languages or the combined language families; I predict that there would be improvements on the lower-resource languages in the same way that they saw improvements from training on all languages in their family. I do not believe that the input from the other languages would have any major effect, as I generally saw with the 7 biggest languages model.

One experiment that I did not run would be to combine the Farsi data and the Arabic data and test on all languages in both the Arabic and Perso-Arabic families, and do the same with Russian and Ukrainian. There may be a chance that these large sets combined together on the basis of similar script could yield even better results on lower-resource languages in the same families without any actual training data from those languages themselves.

It could also be interesting to experiment with incorporating Japanese in the Chinese language family, since some of the characters in written Japanese come from traditional Chinese script, and the model did not do as poorly on the low-resource Chinese languages as might have been expected.

I also agree with Irvine et al. that future experiments that consider language of origin of the name could yield interesting results. The problem with training transliteration models on data with no guarantees about origin is that one model should be followed if it is being back-transliterated, meaning that the name (originally from English) should follow general English spelling patterns. However, if it is a transliteration of a name that is native to the source language, it will follow a model specific to transliteration from the source language into English. Most difficultly, if it is a name that is not native to either of the languages, it will follow a different model, for transliterating its native language into the target language.

Take the example English sound "j" as in "Julie" being transliterated back into English from Chinese. If that sound is from an English name like "Julie" or "John" it should probably be a J when transliterated[6]. However, if it is from a Chinese name such as "Zheng" (pronounced "jung"), in English that same sound is usually written with a "zh" (standard pinyin). However, say it was from an originally Italian name like "Giorgio Armani" - it would be "gi" even in the English version, despite the fact that "gi" in English is usually pronounced like it is in "give." This category of names acting as loan-words as discussed in Section 2 presents probably the largest difficulty and certainly merits further experimentation to see if models for predicting language of origin could be incorporated into a transliteration model.

A final direction for future work would be to experiment with the other direction of transliteration, from English into various languages, or from various languages into a different target language such as Chinese. In these cases, the results of most of my experiments may hold true, particularly if a larger-sized language model could still be trained, or the training data supplemented, using other data from the target script.

6 Conclusion

While transliteration, like many language processing tasks, is still a difficult problem even when there is enough training data, in these experiments I have shown that the performance of models for transliterating low-resource languages into English can be improved by incorporating more data from higher-resource languages using the same or similar script. Even in cases where the language family does not have any large available resources, some improvements can be made by simply including a bigger target language model trained on any relevant data. As Irvine et al. first showed, easy to use existing translation resources such as the Joshua decoder can facilitate experimentation in transliteration, a fact which I was able to leverage in this work and should be considered useful by any transliteration researchers in future endeavors.

[6]The standard Chinese translation for John actually sounds like "yue-han" instead of "jon;" this is just an example

References

Rafael E. Banchs, Min Zhang, Xiangyu Duan, Haizhou Li, , and A Kumaran. 2015. Report of news 2015 machine transliteration shared task. In *CL/IJCNLP 2015's Named Entity Workshop (NEWS 2015)*.

John DeNero and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *Proceedings of ACL*.

Li Haizhou, Zhang Min, and Su Jian. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ann Irvine, Chris Callison-Burch, and Alexandre Klementiev. 2010. Transliterating from all languages. In *Proceedings of The Ninth Biennial Conference of the Association for Machine Translation in the Americas*, Denver, Colorado.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, , and Omar Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 295–302, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jong-Hoon Oh and Hitoshi Isahara. 2006. Extracting english-korean transliteration pairs from web corpora. In *Computer Processing of Oriental Languages. Beyond the Orient: The Research Challenges Ahead: 21st International Conference, ICCPOL 2006, Singapore, December 17-19, 2006, Proceedings*, Singapore.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. pages 57–64. Association for Computational Linguistics.

Appendices

Appendix A: Data Counts - Below

Appendix B: Full Model Evaluations - Attached

Appendix C: Selected Learning Curves - Attached

Appendix D: Cross-Language Testing - Attached

Appendix E: Multi-Language Models - Attached

Language	Code	Script	Script Family	# Pairs	Train	Tune/Test
Abkhazian	ab	Cyrillic	Cyrillic	136	108	14
Amharic	am	Ethiopic	Other	892	713	89
Arabic	ar	Arabic	Arabic	59356	55356	2000
Egyptian Arabic	arz	Arabic	Arabic	4751	3800	475
Assamese	assm	Assamese	Bengali	766	612	77
Avar	av	Cyrillic	Cyrillic	92	73	9
Azerbaijani	az	Latin	Other	13678	10942	1368
Bashkir	ba	Cyrillic	Cyrillic	782	625	78
Belarusian	be	Cyrillic	Cyrillic	19807	15845	1981
Classical Belarusian	be-x-old	Cyrillic	Cyrillic	9752	7801	975
Bulgarian	bg	Cyrillic	Cyrillic	41767	37767	2000
Bihari	bh	Hindi	Devanagari	184	147	18
Bengali	bn	Bengali	Bengali	10569	8455	1057
Tibetan	bo	Tibetan	Other	606	484	61
Chechen	ce	Cyrillic	Cyrillic	622	497	62
Sorani	ckb	Perso-Arabic	Perso-Arabic	3951	3160	395
Chuvash	cv	Cyrillic	Cyrillic	1534	1227	153
Greek	el	Greek	Other	24755	20755	2000
Persian	fa	Perso-Arabic	Perso-Arabic	92295	88295	2000
Gan	gan	Chinese	Chinese	634	507	63
Hebrew	he	Hebrew	Hebrew	51887	47887	2000
Hindi	hi	Devanagari	Devanagari	10287	8229	1029
Armenian	hy	Armenian	Other	16390	13112	1639
Inuktitut	iu	Inuktitut	Other	30	24	3
Japanese	ja	Japanese	Other	148481	144481	2000
Georgian	ka	Georgian	Georgian	16466	13172	1647
Kazakh	kk	Cyrillic	Cyrillic	7446	5956	745
Khmer	km	Khmer	Other	361	288	36
Kannada	kn	Kannada	Other	2737	2189	274
Korean	ko	Korean	Other	56865	52865	2000
Kashmiri	ks	Perso-Arabic	Perso-Arabic	113	90	11
Kyrgyz	ky	Cyrillic	Cyrillic	1252	1001	125
Lao	lo	Lao	Other	89	71	9
Meadow Mari	mhr	Cyrillic	Cyrillic	250	200	25
Macedonian	mk	Cyrillic	Cyrillic	10227	8181	1023
Malayalam	ml	Malayalam	Other	9794	7835	979

Table 4: Language Data, 1 of 2

Language	Code	Script	Script Family	# Pairs	Train	Tune/Test
Mongolian	mn	Cyrillic	Cyrillic	5780	4624	578
Marathi	mr	Balbodh	Devanagari	12558	10046	1256
Burmese	my	Burmese	Other	1830	1464	183
Erzya	myv	Cyrillic	Cyrillic	30	24	3
Mazanderani	mzn	Persian	Perso-Arabic	2471	1976	247
Nepali	ne	Devanagari	Devanagari	2249	1799	225
Newar	new	Devanagari	Devanagari	789	631	79
Oriya	or	Odia	Other	762	609	76
Ossetian	os	Cyrillic	Cyrillic	700	560	70
Punjabi	pa	Gurmukhi	Other	4711	3768	471
Western Punjabi	pnb	Shahmukhi	Perso-Arabic	4356	3484	436
Pashto	ps	Arabic	Arabic	1083	866	108
Russian	ru	Cyrillic	Cyrillic	229680	225680	2000
Rusyn	rue	Cyrillic	Cyrillic	444	355	44
Sanskrit	sa	Devanagari	Devanagari	1613	1290	161
Sakha	sah	Cyrillic	Cyrillic	767	613	77
Sinhalese	si	Sinhala	Other	1180	944	118
Serbian	sr	Cyrillic	Cyrillic	28198	24198	2000
Tamil	ta	Tamil	Other	18555	14844	1855
Telugu	te	Telegu	Other	3929	3143	393
Tajik	tg	Cyrillic	Cyrillic	1782	1425	178
Thai	th	Thai	Other	15524	12419	1552
Tatar	tt	Cyrillic	Cyrillic	2034	1627	203
Uyghur	ug	Arabic	Arabic	287	229	29
Ukranian	uk	Cyrillic	Cyrillic	75054	71054	2000
Urdu	ur	Perso-Arabic	Perso-Arabic	7989	6391	799
Wu	wuu	Chinese	Chinese	756	604	76
Mingrelian	xmf	Georgian	Georgian	1011	808	101
Yiddish	yi	Hebrew	Hebrew	2286	1828	229
Chinese	zh	Chinese	Chinese	85439	81439	2000
Cantonese	zh-yue	Cantonese	Chinese	4611	3688	461

Table 5: Language Data, 2 of 2