Experiments with Cross-lingual Systems for Synthesis of Code-Mixed Text

Sunayana Sitaram[1], *Sai Krishna Rallabandi*[1], *Shruti Rijhwani*[1] *Alan W Black*[2]

[1]Microsoft Research India
[2]Carnegie Mellon University
t-susita@microsoft.com, t-sarall@microsoft.com, t-shruri@microsoft.com, awb@cs.cmu.edu

Abstract

Most Text to Speech (TTS) systems today assume that the input is in a single language written in its native script, which is the language that the TTS database is recorded in. However, due to the rise in conversational data available from social media, phenomena such as code-mixing, in which multiple languages are used together in the same conversation or sentence are now seen in text. TTS systems capable of synthesizing such text need to be able to handle multiple languages at the same time, and may also need to deal with noisy input. Previously, we proposed a framework to synthesize code-mixed text by using a TTS database in a single language, identifying the language that each word was from, normalizing spellings of a language written in a non-standardized script and mapping the phonetic space of mixed language to the language that the TTS database was recorded in. We extend this cross-lingual approach to more language pairs, and improve upon our language identification technique. We conduct listening tests to determine which of the two languages being mixed should be used as the target language. We perform experiments for code-mixed Hindi-English and German-English and conduct listening tests with bilingual speakers of these languages. From our subjective experiments we find that listeners have a strong preference for cross-lingual systems with Hindi as the target language for code-mixed Hindi and English text. We also find that listeners prefer cross-lingual systems in English that can synthesize German text for code-mixed German and English text.

Index Terms: speech synthesis, code-mixing, multilingual systems, pronunciation

1. Introduction

Code-Mixing, in which words from multiple languages are mixed in the same conversation, sentence or word, occurs in multilingual communities around the world. With the advent of social media and conversational content, this phenomenon, which was restricted to speech, is now seen in text as well. Text to Speech (TTS) systems that are used for reading text on the web and social media need to be capable of synthesizing text in multiple languages. Current TTS systems are typically built using recordings from a single speaker in a single language, with the assumption that they will be used for synthesizing text written in that language. There is also a strong assumption that the text that the TTS system synthesizes is written using standardized spellings.

However, code-mixed text provides multiple challenges to TTS systems: first, the TTS system needs to identify all the languages present in the sentence. Next, it must be be able to identify which words in the sentence belong to which language and apply the appropriate pronunciation rules for synthesizing words in each language. Finally, the TTS system should ideally have phonetic coverage of all the languages being mixed, which can be difficult to anticipate in advance and implement in practice.

Sometimes, there are additional complications created by code-mixed text in some language pairs, due to the fact that some languages are not written in their native script, and instead borrow the script of the language they are mixed with. This can create problems of spelling normalization, as there may not exist standard spellings for writing the foreign language. This is seen in languages of the Indian subcontinent, which are usually written in Romanized script on social media and Arabic chat language (Arabizi).

Further, if we are following the approach of using a single TTS database trained completely or primarily on a single language to synthesize code-mixed text, we may also need to choose which of the mixed languages should be used as the target language. Here, the concept of the matrix language may be useful [1], which is defined as the language whose syntax governs the structure of a code-mixed sentence into which the other language is embedded. It may be possible that using the matrix language of the sentence is the appropriate choice in such cases.

There has been some prior work on building bilingual TTS systems for synthesizing bilingual text written in each language's native script. Previously, we have also introduced a framework for synthesizing code-mixed text by using a TTS system trained on a single language. In this work, we extend this approach to more languages, perform better language identification and perform analysis on which language can be chosen as the target language. We perform TTS experiments on code-mixed Hindi and English written in Romanized script, and German and English written in their native scripts.

Section 2 describes how this work relates to previous work on code-mixing and bilingual TTS synthesis. Section 3 describes the data and resources used for this work. Sections 4 and 5 describe the experimental setup and evaluation techniques with results of listening tests. Section 6 concludes with future directions.

2. Relation to Prior Work

Code-switching and code-mixing have received interest in both the Natural Language Processing and Speech Processing communities recently. Although code-switching and code-mixing are distinct phenomena, in the paper, we use the term code-mixing as a general term to describe the use of multiple languages in the same utterance. Note that code-mixing can also happen at the morpheme level, however, our techniques do not explicitly handle this and it is beyond the scope of this paper.

Code-mixing has been studied recently with applications in Information Retrieval and Machine Translation, with a fo-

cus on identifying the languages that are being mixed. The Code Switching shared task at EMNLP 2014 [2] consisted of data from 4 mixed languages (English-Spanish, Nepali-English, Arabic-Arabic dialect, Mandarin-English) and the task was to identify for each word which language it belonged to, or whether it was mixed, ambiguous or a named entity. Chittaranjan et al. [3] describe a CRF based approach for word level Language Identification for this task, in which they used various lexical and character-based features. Vyas et al. [4] created a manually annotated corpus of code-mixed social media posts in Hindi-English and used it for POS tagging. They analyzed this corpus and found that 40% of the Hindi words in the corpus were written in Romanized script. They also found that 17% of the data exhibited code-mixing, code switching or both. They found that transliteration and normalization were the main challenges faced while processing such text. Bali et al. [5] further analyzed this data to find that words fall into categories of code mixing, borrowing and ambiguous, with many borrowed words being written in English and many Hindi words being misidentified as English due to spelling. They suggest that a deeper analysis of morpho-syntactic rules and discourse as well as the socio-linguistic context is necessary to be able to process such text correctly. Gupta et al. [6] introduce the problem of mixed-script Information Retrieval, in which queries written in mixed, native or (often) Roman script need to be matched with documents in the native script. They present an approach for modeling words across scripts using Deep Learning so that they can be compared in a low dimensional abstract space.

Code Switching has also been studied in the context of speech, particularly for Automatic Speech Recognition (ASR) and building multilingual TTS systems. Modipa et al. [7] describe the implications of code-switching for ASR in Sepedi, a South African language and English, the dominant language of the region. They find that the presence of code switching makes it a very challenging task for traditional ASR trained on only Sepedi. Vu et al. [8] present the first ASR system for code-switched Mandarin-English speech. They use the SEAME corpus [9], which is a 64 hour conversational speech corpus of speakers from Singapore and Malaysia speaking Mandarin and English. They use Statistical Machine Translation based approaches to build code mixed Language Models, and integrate a Language ID system into the decoding process. Ahmed et al. [10] describe an approach to ASR for code switched English-Malay speech, in which they run parallel ASRs in both languages and then join and re-score lattices to recognize speech.

Bilingual TTS systems have been proposed by [11] for English-Mandarin code switched TTS. This approach uses speech databases in both languages from the same speaker and a single TTS system that shares phonetic space is built. Microsoft Mulan [12] is another bilingual system for English-Mandarin that uses different frontends to process text in different languages and then uses a single voice to synthesize it. Both these systems synthesize speech using native scripts, that is, each language is written using its own script. Polyglot systems [13] enable multilingual speech synthesis using a single TTS system. This method involves recording a multi language speech corpus by someone who is uent in multiple languages. This speech corpus is then used to build a multilingual TTS system. The primary issue with polyglot speech synthesis is that it requires development of a combined phoneset, incorporating phones from all the languages under consideration. Another type of multilingual synthesis is based upon phone mapping, whereby the phones of the foreign language are substituted with the closest sounding phones of the primary language. This method results in a strong foreign accent while synthesizing the foreign words, which may or may not be acceptable. Also, if the sequence of the mapped phones does not exist or does not occur frequently in the primary language, the synthesis quality can be poor. To overcome this, an average polyglot synthesis technique using HMM based synthesis and speaker adaptation has been proposed [14]. Such methods make use of speech data from different languages and different speakers.

Recently, we proposed a framework for speech synthesis of code-mixed text [15] in which we assumed that two languages were mixed, and one of the languages was not written in its native script but borrowed the script of the other language. Our framework consisted of first identifying the language of a word using a dictionary-based approach, then normalizing spellings of the language that was not written in its native script and then transliterating it from the borrowed script to the native script. Then, we used a mapping between the phonemes of both languages to synthesize the text using a TTS system trained on a single language. We conducted experiments on code-mixed Hindi-English sentences and found that users preferred the system that handled code-mixing to systems that assumed that the input was monolingual, that is either Hindi or English. In this work, we also assumed that the matrix language of all the sentences was Hindi, and made the Hindi synthesizer capable of synthesizing code-mixed Hindi-English sentences written in Romanized script.

In this work, we extend our previous work by performing experiments on German-English code-mixing in addition to Hindi-English. We improve upon our previous dictionary-based technique of performing Language Identification for code-mixed text. We also conduct experiments to determine which language's TTS database should be used when synthesizing code-mixed text.

3. Data and Resources

In this work, we conducted experiments on code-mixed Hindi and English and German and English. For synthesis, we used one bilingual speaker's databases for Hindi-English and twobilingual speakers' databases for German-English. Next, we describe the data and tools we used for performing experiments.

3.1. TTS databases

Our speech databases consisted of Hindi and English data recorded by a female native Hindi speaker and German and English data recorded by two male native German speakers. The Hindi database was 2.5 hours long and contained prompts from a Hindi book by Premchand that is out of copyright. The English database recorded by the same speaker consisted of the CMU ARCTIC [16] A set and was 35 minutes long.

The German AHW database was around 32 minutes long, and the German FEM database was 53 minutes long. The English database recorded by AHW was around 35 minutes long, while the English database recorded by FEM was around 30 minutes long, and both were created with recordings from the ARCTIC A set. The prompts for the German databases were taken from Europarl data [17].

All three speakers had high proficiency in English. There were no English words in the Hindi and German data, other than borrowed words from English in the German data. Since the Hindi data came from books written in the 19th century, they contained no borrowed words from English.

3.2. Code-mixed test data

Since our aim was to synthesize code-mixed text, our test sentences consisted of data from social media in Hindi-English and German-English. The Hindi-English data consisted of a corpus of comments on a Hindi recipe website which consisted of code-mixed Hindi and English written entirely in the Romanized script. The German-English data consisted of tweets that were crawled from Twitter. We selected 15 sentences from each corpus for conducting subjective listening tests, which we synthesized using our techniques. Example sentences from Hindi-English and German-English are shown below.

Hindi-English: *Dear nisha mujhe hamesha kaju barfi banane mein prob hoti h plzz mujhe kaju katli ki easy receipe bataiye*
Translation: Dear Nisha I always have a problem making Kaju Barfi please give me an easy recipe for Kaju Katli.

German-English: *Gay marriage now legal in all US states Der US Supreme Court hat entschieden und wir feiern*
Translation: Gay marriage now legal in all US states the US Supreme Court has ruled and we celebrate.

In the Hindi example above, the Romanized script is used to write Hindi words, and standard spellings do not exist for Hindi words. In addition, this data also has non-standard spellings and contractions for English words. The German examples are cleaner with less spelling variations for both German and English words, even though this data came from Twitter. The Hindi-English data has many switch points, where the languages alternate. However, most of the German-English sentences had one or at the most two switch points.

3.3. Spelling normalization

As we saw in the examples above, Hindi written in Romanized script does not have standardized spellings. To deal with this and the problem of non-standard spellings and contractions, we used a spelling normalization technique which replaced each word with a high-frequency word in the Hindi recipe corpus. The high-frequency word was chosen using the SoundEx algorithm as described in [15].

We did not perform spelling normalization for our German-English code-mixed data. However, the same approach can be easily extended to other languages provided we have a sufficiently large text corpus that has code-mixing in the language pair.

The image below shows the spelling variants of the word 'recipe' found in the Hindi-English recipe corpus. Each spelling variant was replaced by the highest frequency word, which was the correct spelling of 'recipe' in this case. In cases where a high-frequency match was not found, the word was left unnormalized.

recipe receipe recepie Recipe recipie recipy reciepe recipi recepi RECIPE recipee resipe recpie racipe resipi recepe recipei receipy reciepy recipel recepy reciepi receipi resepi recpi reciepie recip

Figure 1: Spelling variants in the "recipe" cluster

3.4. Language identification

Most language identification (LID) systems classify each document or sentence with a single language [18, 19, 20, 21]. With code-mixed data, it becomes necessary to identify the language of each word, as code-mixing, apart from at the sentence level, often occurs at the phrase, word or morpheme level [22, 1].

In previous work on synthesizing Hindi-English code-mixed text, we used a naive dictionary-based approach, that assigned an English language tag to all words found in CMUdict [23], a lexicon containing English words, and tagged all remaining words as Hindi. In subjective listening tests for Hindi-English, we found that there was a gap between listener preference for our system compared to a system with ground truth language labels and manual spelling normalization. We felt that we could get a large improvement in our system by performing better LID. Next, we describe in brief a language identification system for German-English code-mixed text (work in submission) that we used as part of our system. For Hindi-English text, we use the system designed by Gella et al. [24] with a small modification.

For German-English code-mixing, we use a Hidden Markov Model (HMM) that has a state for each language (German and English, in this case) and a state to represent extralinguistic tokens (punctuation, digits and other special characters). Using the Viterbi decoding algorithm, each word in the input sentence is sequentially labeled by the HMM with either a language or as extra-linguistic. Unlike previous work on code-mixing identification [2, 25, 26, 27, 28, 29], our technique does not require annotated monolingual data or code-mixed data with word-level language annotations. Such data is challenging to obtain and expensive to annotate. Instead, we tuned the HMM parameters using automatically identified tokens in German and English as weakly-labeled data. The weakly-labeled data contained 100,000 tokens in each language. The word-level labeling accuracy, tested on German-English data from Twitter, was 98.4%.

We used the technique developed by Gella et al. [24] for LID on Hindi-English code-mixed text. This was the best performing system in the FIRE 2013 shared task on Hindi-English word-level language labeling [27]. Gella et al. [24] use 5000 instances from the FIRE 2013 training data [27]. The system uses maximum entropy classifiers trained on character n-grams from Hindi and English words. For each word in the input, the system gives two values that represent the probability of the word coming from English and Hindi. With these probabilities, we obtained the most likely sequence of labels using the Viterbi algorithm. While decoding, we also introduced a parameter that penalizes code-mixing, as remaining in the same language (monolingual sequences) is more common than code-mixing. This parameter was tuned on a development set containing 500 Hindi-English tweets.

3.5. Synthesis techniques

All TTS experiments were carried out using the Festvox voice building tools [30]. We built standard CLUSTERGEN [31] Statistical Parametric Synthesis voices that ran using the Festival [32] Speech Synthesis engine.

We used the Festvox Indic front end [33] to build the Hindi voice and the Festvox German frontend to build the German voices. The Festvox Indic frontend contains hand-written rules to handle various linguistic phenomena in Indian languages such as schwa deletion, lexical stress rules, contextual nasalization etc. The Festvox German frontend made use of the BOMP lexicon [34] and letter-to-sound rules. The English voices were built using the standard Festvox US English front end that used CMUdict as the lexicon and a letter-to-sound model built using

CMUdict for predicting the pronunciation of unseen words.

4. Experiments

We refer to our approach of building voices that are capable of speaking languages other than the language of the training TTS database as the cross-lingual approach to distinguish it from a bilingual approach where TTS databases in both languages being mixed are used. We build monolingual systems in all the languages being mixed, assuming that the input was in a single language. We also built cross-lingual systems for Hindi-English and German-English, described below.

4.1. Monolingual systems

We built monolingual systems using 5 TTS databases (3 English and 2 German) in which we used the standard frontends and assumed that all the test sentences were in the target language. This was not the case for the Hindi system, since the text was in Romanized script and the Hindi system assumed that Hindi input would be in Devanagari. Instead, we built a monolingual system for Hindi by transliterating all the Romanized text to Devanagari and treating all the text as Hindi. We performed this transliteration by using a decision-tree based model trained on a few hundred Romanized Hindi-Devanagari pairs. This is described in more detail in [35].

4.2. Cross-lingual systems

We built cross-lingual systems for Hindi-English using the approach we described in [15]. We normalized spellings using the approach described earlier. We identified the language of each word in the sentence by using the dictionary-based technique or the Maximum Entropy based technique. Once the languages were identified, we transliterated the Hindi words into Devanagari using the transliteration model. Then, we ran the words through their respective frontends (Hindi for the Devanagari words, English for the Romanized words) and mapped phonemes to the target language's frontend. Finally, we synthesized the sentences by using the monolingual English TTS system and Hindi TTS system.

Similarly for German-English, we identified the language of each word with the dictionary-based technique and the HMM-based technique described earlier. We mapped phonemes from the German frontend to English and vice-versa. When an exact phoneme match was not found, we substituted it with the closest sounding phoneme based on its phonetic features. In case of German, we did not use a transliteration model to map words since the words were already written in the correct script. We synthesized all code-mixed sentences using the German and English TTS systems of both speakers.

In all, we had 6 cross lingual systems, one for each TTS database.

5. Evaluation and Results

First, we calculated the accuracy of the Language Identification systems described earlier. The test data was manually annotated with ground truth language labels by bilingual speakers of Hindi-English and German-English. Table 1 compares the LID system to the dictionary-based approach. For Hindi. since we felt that spelling normalization was a crucial part of the pipeline, we report LID accuracies for normalized and unnormalized spellings.

From the results in Table 1 we find that the accuracy of the HMM/MaxEnt systems is higher than the dictionary-based systems by a large margin for all the systems. We also find that the accuracy of the LID for normalized Hindi is higher than the accuracy for un-normalized Hindi, which is to be expected.

Table 1: *Language Identification Accuracy*

Data	Dictionary	HMM/MaxEnt
En-Hi (no normalization)	66%	78%
En-Hi (normalized)	69%	89%
En-De	65%	96%

In our previous work, we had used a Hindi TTS system to synthesize sentences in mixed Hindi-English. However, a choice can be made as to whether to use a Hindi system or English system as the target language. A heuristic that can be used in this case is to consider the matrix language to be the language with more words in the sentence. However, we decided to ask users to listen to cross-lingual systems in both languages and choose the one they prefer. To test our synthesized output, we ran listening tests on Amazon Mechanical Turk using the Testvox tool [36] in which we asked 10 bilingual speakers of Hindi-English and German-English to listen to 10 sentences and evaluate our systems. We asked them to choose the system that was easier to understand among the two cross lingual systems in a language pair.

Table 2: *Listener Preference - Matrix Language*

Data	Matrix En	Matrix Hi/De	No difference
En-Hi	17%	79%	4%
En-De (AHW)	76%	16%	3%
En-De (FEM)	82%	11%	7%

From the listening test results in Table 2, we can see that for Hindi-English, there was a strong preference for the Hindi voice as the target language. On analyzing the test sentences, we found that the majority of the words in the Hindi-English data belonged to Hindi (63%). However, the Hindi TTS system was built with significantly more data than the English system, which could also have influenced the listeners' judgments.

In the German-English case, the data was more balanced, with 50.6% German words in the sentences. There was a strong preference for using English as the matrix language when compared to German, even though the amount of German and English in the sentence was roughly the same. This could have been due to two reasons - the quality of the English frontend was superior to the German frontend, which made the English systems better in general. In addition, in most of the test sentences we looked at, the first half of the sentence was in English, while the second half was in German, which may have influenced listeners to pick the English system, which would pronounce the first half of the sentence correctly over the German system.

Our best cross-lingual systems were built with normalization (for Hindi-English), the new LID methods we described and the matrix language that users preferred. Next, we tested our best cross-lingual systems against monolingual systems in the matrix language users preferred. Once again, we asked 10 bilingual Hindi-English and German-English speakers on Amazon Mechanical Turk to listen to 10 sentences in each pair and choose the system that was more understandable. In this case, the base TTS systems were the same, so the only difference the

users heard was in the pronunciation of the words.

Table 3: *Listener Preference - Cross-lingual vs Monolingual*

Data	CrossLingual	Monolingual	No difference
En-Hi	81%	16%	3%
En-De (AHW)	41%	30%	29%
En-De (FEM)	46%	35%	19%

From the listening tests we found a very strong preference for the cross-lingual Hindi-English system over the monolingual Hindi system, that assumed that all the input was in Hindi. We also found a preference for cross-lingual systems in English that could synthesize German words for both the AHW and FEM databases, although the preference was not as high as for Hindi, and many listeners found no difference between the systems. This could be because the pronunciation rules for Hindi and English differ much more than the pronunciation rules for German and English.

6. Conclusion

In this paper, we extended the capabilities of monolingual systems to synthesize code-mixed text, in which multiple languages are used in the same sentence. We used 6 TTS databases in Hindi, German and English to synthesize Hindi-English and German-English mixed text.

We extended preliminary work on Hindi-English code-mixed synthesis to other databases and also improved our Language Identification system. Further, we also made use of the fact that we had bilingual databases from the same speaker to compare which language could be used as a target language while synthesizing code-mixed text.

We used a straightforward approach for spelling normalization in which words from a large corpus were replaced by their high frequency spelling variants. This approach did not take into account the pronunciation of the words or the context in which they appeared. Using word vectors to find the closest spelling variant could be an interesting direction to pursue for this problem, particularly since many contractions in social media are difficult to normalize using spelling and pronunciation alone.

We are currently working on using the databases used in this work to build bilingual voices. We are exploring techniques to combine the phonetic space in both languages and map pronunciations across languages better. Future work includes comparing the cross-lingual systems we have built with such bilingual systems.

Finally, in this work we used monolingual databases of Hindi, German and English to create systems that were capable of synthesizing code-mixed Hindi-English and German-English. None of these databases were explicitly designed to handle code-mixing, however, the German databases may have had some foreign words in them. Future work in synthesizing code-mixed text includes designing databases explicitly to handle code-mixing and foreign words.

7. References

[1] C. Myers-Scotton, *Duelling languages: grammatical structure in codeswitching*. Oxford University Press, 1997.

[2] T. Solorio, E. Blair, S. Maharjan, S. Bethard, M. Diab, M. Gohneim, A. Hawwari, F. AlGhamdi, J. Hirschberg, A. Chang *et al.*, "Overview for the first shared task on language identification in code-switched data," in *Proceedings of The First Workshop on Computational Approaches to Code Switching*. Citeseer, 2014, pp. 62–72.

[3] G. Chittaranjan, Y. Vyas, and K. B. M. Choudhury, "Word-level language identification using crf: code-switching shared task report of MSR India system," *EMNLP 2014*, p. 73, 2014.

[4] Y. Vyas, S. Gella, J. Sharma, K. Bali, and M. Choudhury, "Pos tagging of English-Hindi code-mixed social media content," *Proceedings of the First Workshop on Codeswitching, EMNLP*, 2014.

[5] K. Bali, J. Sharma, M. Choudhury, and Y. Vyas, ""i am borrowing ya mixing?" an analysis of English-Hindi code mixing in Facebook," *EMNLP 2014*, p. 116, 2014.

[6] P. Gupta, K. Bali, R. E. Banchs, M. Choudhury, and P. Rosso, "Query expansion for mixed-script information retrieval," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 677–686.

[7] T. I. Modipa, M. H. Davel, and F. De Wet, "Implications of Sepedi/English code switching for ASR systems," 2013.

[8] N. T. Vu, D.-C. Lyu, J. Weiner, D. Telaar, T. Schlippe, F. Blaicher, E.-S. Chng, T. Schultz, and H. Li, "A first speech recognition system for Mandarin-English code-switch conversational speech," in *ICASSP*. IEEE, 2012, pp. 4889–4892.

[9] D.-C. Lyu, T.-P. Tan, E.-S. Chng, and H. Li, "Mandarin-English code-switching speech corpus in South-East Asia: SEAME," *Language Resources and Evaluation*, pp. 1–20, 2015.

[10] B. H. Ahmed and T.-P. Tan, "Automatic speech recognition of code switching speech using 1-best rescoring," in *Asian Language Processing (IALP), 2012 International Conference on*. IEEE, 2012, pp. 137–140.

[11] H. Liang, Y. Qian, and F. K. Soong, "An HMM-based bilingual (Mandarin-English) TTS," *Proceedings of SSW6*, 2007.

[12] M. Chu, H. Peng, Y. Zhao, Z. Niu, and E. Chang, "Microsoft Mulan-a bilingual TTS system," in *ICASSP*, vol. 1. IEEE, 2003, pp. I–264.

[13] C. Traber, K. Huber, K. Nedir, B. Pfister, E. Keller, and B. Zellner, "From multilingual to polyglot speech synthesis." in *Eurospeech*, 1999.

[14] J. Latorre, K. Iwano, and S. Furui, "New approach to the polyglot speech generation by means of an hmm-based speaker adaptable synthesizer," *Speech Communication*, vol. 48, no. 10, pp. 1227–1242, 2006.

[15] S. Sitaram and A. W. Black, "Speech synthesis of code-mixed text," in *LREC*, 2016.

[16] J. Kominek and A. W. Black, "The cmu arctic speech databases," in *Fifth ISCA Workshop on Speech Synthesis*, 2004.

[17] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT summit*, vol. 5, 2005, pp. 79–86.

[18] H. P. Fei Xia, William D Lewis, "Language id in the context of harvesting language data off the web," in *In Proceedings of the 12th EACL*, 2009, pp. 870–878.

[19] M. P. Erik Tromp, "Graph-based n-gram language identification on short texts," in *In Proc. 20th Machine Learning conference of Belgium and The Netherlands*, 2011, pp. 27–34.

[20] T. B. Marco Lui, "langid.py: An off-the-shelf language identification tool," in *In Proceedings of the ACL 2012 System Demonstrations*, 2012, pp. 25–30.

[21] S. Bergsma, P. McNamee, M. Bagdouri, C. Fink, and T. Wilson, "Language identification for creating language-specific Twitter collections," in *Proceedings of the Second Workshop on Language in Social Media*, 2012, pp. 65–74.

[22] J. J. Gumperz, *Discourse strategies*. Cambridge University Press, Cambridge, 1982.

[23] R. Weide, "The CMU pronunciation dictionary, release 0.6," 1998.

[24] S. Gella, J. Sharma, and K. Bali, "Query word labeling and back transliteration for indian languages: Shared task system description," *FIRE Working Notes*, vol. 3, 2013.

[25] B. King and S. Abney, "Labeling the languages of words in mixed-language documents using weakly supervised methods," in *Proceedings of NAACL-HLT*, 2013, pp. 1110–1119.

[26] D. Nguyen and A. S. Dogruoz, "Word level language identification in online multilingual communication," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 857–862.

[27] R. Saha Roy, M. Choudhury, P. Majumder, and K. Agarwal, "Overview and datasets of fire 2013 track on transliterated search," in *FIRE Working Notes*, 2013.

[28] M. Choudhury, G. Chittaranjan, P. Gupta, and A. Das, "Overview of fire 2014 track on transliterated search," 2014.

[29] R. Sequiera, M. Choudhury, P. Gupta, P. Rosso, S. Kumar, S. Banerjee, S. K. Naskar, S. Bandyopadhyay, G. Chittaranjan, A. Das *et al.*, "Overview of fire-2015 shared task on mixed script information retrieval," 2015.

[30] A. W. Black and K. Lenzo, "Building voices in the Festival speech synthesis system," Tech. Rep., 2002. [Online]. Available: http://festvox.org/bsv

[31] A. W. Black, "CLUSTERGEN: a statistical parametric synthesizer using trajectory modeling," in *Interspeech*, 2006.

[32] P. Taylor, A. W. Black, and R. Caley, "The architecture of the Festival speech synthesis system," 1998.

[33] A. Parlikar, S. Sitaram, A. Wilkinson, and A. W. Black, "The festvox indic frontend for grapheme to phoneme conversion," in *WILDRE: Workshop on Indian Language Data - Resources and Evaluation*, 2016.

[34] T. Portele, J. Krämer, and D. Stock, "Symbolverarbeitung im sprachsynthesesystem hadifix," in *Proc. 6. Konferenz Elektronische Sprachsignalverarbeitung*, 1995, pp. 97–104.

[35] S. Sitaram, "Pronunciation modeling for synthesis of low resource languages," Ph.D. dissertation, Carnegie Mellon University, 2015.

[36] A. Parlikar, "TestVox: web-based framework for subjective evaluation of speech synthesis," *Opensource Software*, 2012.