Using Alternate Representations of Text for Natural Language Understanding

Venkata sai Varada, Charith Peris, Yangsook Park, Christopher DiPersio
Amazon Alexa AI, USA
{vnk, perisc, yangsoop, dipersio}@amazon.com

Abstract

One of the core components of voice assistants is the Natural Language Understanding (NLU) model. Its ability to accurately classify the user's request (or "intent") and recognize named entities in an utterance is pivotal to the success of these assistants. NLU models can be challenged in some languages by code-switching or morphological and orthographic variations. This work explores the possibility of improving the accuracy of NLU models for Indic languages via the use of alternate representations of input text for NLU, specifically ISO-15919 and IndicSOUNDEX, a custom SOUNDEX designed to work for Indic languages. We used a deep neural network based model to incorporate the information from alternate representations into the NLU model. We show that using alternate representations significantly improves the overall performance of NLU models when the amount of training data is limited.

1 Introduction

Building NLU models can be more challenging for languages that involve code-switching. Recent times have seen a significant surge of interest in voice-enabled smart assistants, such as Amazon's Alexa, Google Assistant, and Apple's Siri. These assistants are powered by several components, which include Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) models. The input to an NLU model is the text returned by the ASR model. With this input text, there are two major tasks for an NLU model: 1) Intent Classification (IC), and 2) Named Entity Recognition (NER).

Code-switching is a phenomenon in which two or more languages are interchanged within a sentence or between sentences. An example of code-switching from Hindi is 'मेरी shopping list में dove soap bar जोड़ें' ('add dove soap bar to my shopping list'). NLU models expect ASR to return the text in the above form, which matches with the transcription of the training data for NLU models. Then, NLU model would return the action as *Add Items To Shopping List*, while it also recognizes 'dove soap bar' as the actual item name to be added to the shopping list. However, ASR could inconsistently return 'मेरी shopping list में dove soap बार जोड़ें', where the English word 'bar' is recognized as a Hindi word 'बार'. Note that while the Hindi word 'बार' means something different than the English 'bar', their pronunciation is similar enough to mislead the ASR model in the context of mixed-language utterance. In this case, the NLU model should become robust against such ASR mistakes, and learn that 'dove soap बार' is equivalent to 'dove soap bar' in order to correctly recognize it as an item name. The phenomenon of code-switching is common amongst other Indic languages as well.

ASR inconsistencies can cause significant challenges for NLU models by introducing new data that the models were not trained on. Such inconsistencies can occur due to 1) code-switching in the data, especially when the input text contains more than one script (character set), 2) orthographic or morphological variations that exist in the input text, or 3) due to requirements for multilingual support. In a language such as English in the United States, where code-switching is less common, both ASR and NLU models can perform quite well, as input data tend to be more consistent. However, when it comes to Indic languages such as Hindi, where code-switching is much more common, the question arises as to which representation of the input text would work best for NLU models.

There is ongoing research on solving modeling tasks for data with code-switching. In Geetha et al. (2018), the authors explore using character and word level embeddings for NER tasks. Some of this research focuses on using alternate representations of text for NLU modeling. For example, Johnson et al. (2019) explore the use of cross-lingual transfer learning from English to Japanese for NER by romanizing Japanese input into a Latin-based character set to overcome the script differences between the language pair. Zhang and LeCun (2017) has shown using romanization for Japanese text for sentiment classification hurt the performance of the monolingual model.

In this paper, we explore the possibility of mitigating the problems related to ASR inconsistency and code-switching in our input data by using two alternate representations of text in our NLU model: ISO-15919 and IndicSOUNDEX. ISO-15919[1] was developed as a standardized Latin-based representation for Indic languages and scripts. SOUNDEX is an algorithm that provides phonetic-like representations of words. Most work on SOUNDEX algorithms has focused primarily on monolingual solutions. One of the best known implementations with a multilingual component is Beider-Morse Phonetic Matching Beider and Morse (2010); however, it only identifies the language in which a given word is written to choose which pronunciation rules to apply. Other attempts at multilingual SOUNDEX algorithms, particularly for Indic languages, were smaller studies focused on two Indic languages with or without English as a third language. Chaware and Rao (2012) developed a custom SOUNDEX algorithm for monolingual Hindi and Marathi word pair matching. Shah and Singh (2014) describe an actual multilingual SOUNDEX implementation designed to cover Hindi, Gujarati, and English, which, in addition to the actual algorithm implementation, was aided by a matching threshold declaring two conversions a match even if they differed in up to two characters.

In this study, we use ISO-15919 and IndicSOUNDEX representations of text in a deep neural network (DNN) to perform multi-task modeling of IC and NER. We experiment on one high-resource Indic language (Hindi) and three low-resource Indic languages (Tamil, Marathi, Telugu). In Section 2, we describe the two alternate representations of text that we explore. In Section 3, we describe our data, model architecture, and detail our experimental setup. In Section 4, we present our results followed by our conclusions in Section 5.

[1] https://en.wikipedia.org/wiki/ISO_15919

2 Alternate representations of text

Using data transcribed in the original script of a language can cause problems for both monolingual and multilingual NLU models. For a monolingual model in a language where code-switching, orthographic variations, or rich morphological inflections are common, NLU models may not be able to perform well on all the variations, depending on the frequency of these variations in the training data. For multilingual models, words with similar sound and meaning across different languages (e.g., loanwords, common entities) cannot be captured if the words are written in their original script. For example, the same proper noun 'telugu' is written as 'तेलुगु' in Hindi, as 'தெலுகு' in Tamil, and as 'తెలుగు' in Telugu. Although they sound similar and mean the same thing, NLU model will see them as unrelated tokens if they are represented in their original scripts in the input data.

From the NLU point of view, a text representation that can minimize variations of the same or similar words within a language and across different languages would be beneficial for both IC and NER tasks. In this section, we explore two alternative ways of text representations for Indic languages: ISO-15919 and a SOUNDEX-based algorithm, which we call IndicSOUNDEX. Compared to using the original scripts, these two alternatives can represent the variants of the same word or root in the same way. For example, in the original Hindi script (i.e., Devanagari), the word for 'volume/voice' can be represented in two forms: 'आवाज़' and 'आवाज'. These two forms, however, are uniformly represented as 'āvāj' in ISO-15919 and as the string 'av6' in IndicSOUNDEX. Similarly, the English word 'list' may be transcribed as 'list' or as 'లిస్ట్' in Telugu; however, they map to the same IndicSOUNDEX representation, 'ls8'.

2.1 ISO-15919

ISO-15919 is a standardized representation of Indic scripts based on Latin characters, which maps the corresponding Indic characters onto the same Latin character. For example, 'क' in Devanagari, 'క' in Telugu, and 'க' in Tamil are all represented by 'k' in the ISO-15919 standard. Consequently, ISO-15919 can be used as a neutral common representation between Indic scripts. However, ISO-15919 has some downsides. Indic scripts often rely on implicit vowels which are not represented in the orthography, which means they cannot be reliably added to a transliterated word. Additionally, Indic scripts have a character called a halant, or virama, which is used to suppress an inherent vowel. This character, although usually included orthographically in a word, does not have an ISO-15919 representation and so is lost in an exact, one-to-one conversion. Finally, it is not always the case that the same word in two different Indic scripts will have the same ISO-15919 conversion due to script-to-script and language-to-language differences and irregularities. Table 1 below shows some examples of conversion using ISO-15919.

2.2 IndicSOUNDEX

SOUNDEX algorithms provide phonetic-like representations of words that attempt to reduce spelling and other orthographic variations for the same or similarly-sounding words, mainly to increase recall in information retrieval or text search tasks. This is done by following a set of simple steps which may include removing vowels, reducing character duplication, and mapping sets of characters to a single character, based on whether they 1) sound similar or 2) are used in similar environments in similar sounding words. For example, at its most extreme, American SOUNDEX maps 'c', 'g', 'j', 'k', 'q', 's', 'x', and 'z' to the same character: '2'.

IndicSOUNDEX is a custom SOUNDEX approach designed to work on Hindi, Marathi, Telugu, Tamil, Malayalam, Punjabi, Bengali, Kannada, Gujarati, and English. Table 1 shows some examples of conversion using IndicSOUNDEX:

3 Experimental Setup

3.1 Data

For our experiments, we chose datasets from four Indic languages: Hindi, Tamil, Marathi, and Telugu. For Hindi, we use an internal large-scale real-world dataset; for the other three languages, we use relatively small datasets collected and annotated by third party vendors. We perform a series of experiments to evaluate the use of the alternate representations of text, ISO-15919 and IndicSOUNDEX, described in Section 2.

Our Hindi training dataset consists of 6M data. We separate a portion (1%) of the data into an independent test set. We execute a stratified split on the remainder, based on intent, and choose 10% of this data for validation and the rest as training data. For the three smaller-scale datasets, we execute a stratified split with 60% for training, 20% for validation, and 20% for testing. Table 2 shows the data partitions across different languages used for our experiments.

Each of the four datasets contain code-switching. The transcription was done either in the original script of the language (for words from that language) or in the standard Latin (for words from other languages including English). However, the transcription was not always consistent, especially in the third party data, so some Indic words were transcribed in the so-called 'colloquial' Latin (i.e., a casual, non-standard way of representing Indic languages in online resources) and some English words were represented in the original script of the Indic languages (e.g., 'లిస్ట్' for the English word 'list'). See Table 3 for the total counts of tokens in each script in each of the training and test data, which reflects the use of code-switching in each training dataset. Note that Hindi and Marathi both use the same script (Devanagari) in their writing systems.

3.2 Model architecture

For our NLU models, we used a multi-task modeling framework that predicts Intent and Named Entities, given an input sentence. A schematic of our model is given in Figure 1 and Figure 2. For a given corpus, we built alternate text representations using the ISO-15919 and IndicSOUNDEX approaches mentioned in

Table 1: Examples of conversion using ISO-15919 and IndicSOUNDEX

Script	Indic Original	ISO-15919	IndicSOUNDEX
Devanagari (Hindi)	इन्दिके	indikē	i034
Devanagari (Marathi)	इतिहासाचा	itihāsācā	i8hs2
Telugu	పడినది	pḍindi	p303
Tamil	பிரவீண்	pirvīṇ	plv0

Table 2: Data partition across languages

Language	Train	Test	Valid	Total
Hindi	5.4M	600K	54K	6M
Tamil	27K	9k	9k	45K
Marathi	27K	8.5k	8.5K	42K
Telugu	27K	9K	9k	46K

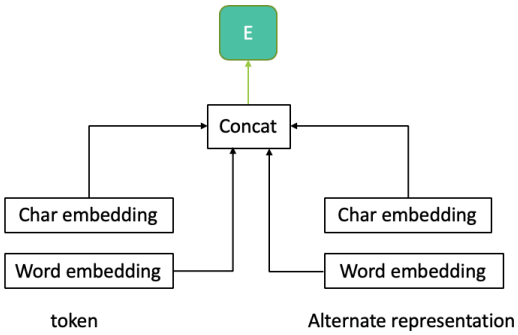

Figure 1: Embedding for a token

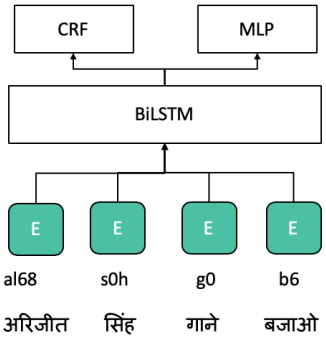

Figure 2: Modeling architecture with tokens and alternate representation (IndicSOUNDEX) as input

Section 2. We used both the original sentence as well as the alternate representations of the sentence to inform our models via added embedding layers.

First, we built a baseline model without using any alternate representations using the Baseline Embedding block architecture below. Next, we built two candidate models: the first with an embedding layer using alternate representations from IndicSOUNDEX, and the second with the alternate representations from ISO-15919. Our modeling architecture is designed as follows:

Baseline Embedding Block: For our baseline model, we designed our embedding block with two layers and concatenated the outputs. The first layer is a token embedding layer of dimension 100, while the second one is a character-based token embedding built using convolutional neural network (CNN) subsequence embedding with dimension of 16. The final embedding used in our models will be generated by concatenating these two embeddings.

Embedding block with alternate representations: For our alternate representation models, we modified the baseline embedding block to accommodate these representations. All other components, including encoder and decoder, stayed the same. The embedding block for this setting was modified to have four layers with final embedding used in our model being the concatenation of these. A schematic is shown in Figure 1 The four layers are as follows:

1. Token embedding layer of dimension 100.
2. Token embedding layer for alternate representations of tokens of dimension 100 .
3. Character embedding built using convolutional neural network (CNN) subsequence embedding with dimension of 16.
4. Alternate representations character embedding built using convolutional neural network (CNN) subsequence embedding with dimension of 16.

Eventually, we concatenated the output of the embedding layer to obtain the final word embedding.

Table 3: The number of tokens in each script within each training and test set

	Hindi		Tamil		Marathi		Telugu	
Script	Train	Test	Train	Test	Train	Test	Train	Test
Latin	13.9M	101K	22.5K	7.5K	29.1K	9.7K	40.7K	13.4K
Devanagari	15.5M	102K	0	0	110K	36K	0	0
Tamil	8	0	108K	36K	0	0	0	0
Telugu	0	0	0	0	0	0	112K	37.5K
Other	1K	0	2	0	0	0	0	0
Total	29.5M	203K	131K	43.5K	139K	46K	153K	51K

Encoder: We defined a biLSTM (bidirectional Long Short Term Memory) encoder with 5 hidden layers and a hidden dimension of 512. A schematic is given in Figure 2. In order to handle our multi-task prediction of Intent and NER, we have two output layers:

1. A multi-layer perceptron (MLP) classifier with the number of classes set to the vocabulary size of Intent.
2. A conditional random fields (CRF) sequence labeler with the number of classes set to the vocabulary size of Named Entities.

Decoder: We used a joint task decoder for our purpose. The Intent-prediction task is accomplished by single class decoder, while the label-prediction task is achieved by sequence labeling decoder. For this setup, the loss function is a multi-component loss, with cross-entropy loss for Intent and CRF-loss (Lafferty et al., 2001) for NER.

Evaluation metric: We use Semantic Error Rate (SemER) Su et al. (2018) to evaluate the performance of our models. The definition of SemER is as follows:

$$SemER = \frac{(D + I + S)}{(C + D + S)} \quad (1)$$

where D (deletion), I (insertion), S (substitution), C (correct slots).

As the Intent is treated as a slot in this metric, the Intent error is considered as a substitution.

Experimental Process: Our experimental setup consists of the following process. For each language, we build three models with the model architecture explained in model architecture section.

- Baseline model - a baseline model with architecture explained in the model architecture section using word and character embeddings from tokens.
- IndicSOUNDEX model - a model with IndicSOUNDEX alternate representations i.e., using an embedding layer with tokens, the IndicSOUNDEX representation of tokens, characters from tokens and the IndicSOUNDEX representations of characters.
- ISO-15919 model - a model with ISO-15919 alternate representations i.e., using embedding layer with tokens, the ISO-15919 representation of tokens, characters from tokens and the ISO-15919 representations of characters.

4 Results

To evaluate each model, we used our withheld test dataset and measured SemER scores associated with the three different models. To obtain confidence intervals, we bootstrapped our test dataset by sampling ten times with replacement, and evaluated our models on each of these ten datasets. Final SemER was taken as the average of all the ten iterations. We used a Student's t-test to calculate the significance of the improvements of the IndicSOUNDEX and ISO-15919 models with respect to the baseline. Here we present the results of our three models: the Baseline DNN model, IndicSOUNDEX model, and ISO-15919 model on each of the four languages.

Table 4: Comparison of ISO-15919 and IndicSOUNDEX model performance on Hindi w.r.t baseline. '+' indicates degradation, '-' indicates improvement. '*' denotes the results that are not statistically significant

Language	Model	Topics improved	Average improvement	Topics degraded	Average degradation	Overall change
Hindi	IndicSOUNDEX	3	8%	1	5%	–0.07*%
	ISO-15919	3	4%	4	9%	+1.08%

4.1 Results for a high-resource language - Hindi

Our Hindi data consisted of data from 22 topics including Music, Reminders, Alarms, Weather, Search, News, and so on. See Appendix for the full list of topics. Table 4 shows the performance on Hindi. Results revealed that, out of the 22 topics, the use of IndicSOUNDEX resulted in 3 topics showing improvement in SemER, with an average improvement of 8% and 1 topic showing 5.36% degradation. The use of ISO-15919 resulted in 3 topics showing an average improvement of 4% and 4 topics showing an average degradation of 9%. Rest of the topics showed results that are not statistically significant. We note that two topics showed improvement across both models: SmartHome and Global.

Our results show that there is no overall (i.e., across all topics together) statistically significant improvement seen for the IndicSOUNDEX model. However, we note that the improvements in 3 three topics: Global, SmartHome, and ToDoLists are significant. The one topic that showed a degradation was Shopping.

On the other hand, the ISO-15919 model shows an overall 1.08% degradation that is statistically significant. The ISO-15919 model shows statistically significant improvements in Global, Video, and SmartHome topics, and degradation in Weather, Music, QuestionAndAnswer, and Media.

In summary, for a high-resource language such as Hindi, we find that neither IndicSOUNDEX nor ISO-15919 shows an overall significant improvement. However, there are certain topics that could benefit from using these alternate representations either for IC or NER. Note that the majority of the training data for Hindi were transcribed by well-trained internal human transcribers and went through some cleaning processes for common transcription errors. Also, given the size of the training data, the NLU models were well trained on the various spelling variations represented in the original script. Owing to this relatively high consistency in transcription and the existence of various tokens with similar meaning and sound in the training data, we believe that using the alternate representations of text was not effective for improving the performance of the NLU model.

4.2 Results for low-resource languages - Tamil, Marathi, and Telugu

Unlike the case of Hindi, we see much more significant overall improvements where training data are sparse. All three languages showed significant improvement in overall performance for ISO-15919 model, whereas IndicSOUNDEX showed significant improvement for Tamil and Marathi. Within Tamil and Marathi, IndicSOUNDEX showed a larger improvement than ISO-15919.

Our data for the low-resource languages consisted of 19 different topics. Table 5 shows the performance of each language. At topic level for Tamil, we found that using the IndicSOUNDEX model improved the performance in 15 out of 19 topics with an average SemER drop of 11%. With the ISO-15919 representation, 13 out of 19 topics showed improvement with an average SemER drop of 13%. For Marathi, IndicSOUNDEX improved 7 topics with an average drop in SemER of 18%, whereas ISO-15919 improved 9 topics but with a lower average drop in SemER (10%). For Telugu, using IndicSOUNDEX or ISO-15919 improved the performance of 4 topics each with the same average drop in SemER of 15%. There were two topics that showed improvement across all three languages with the IndicSOUNDEX model: MovieTimes and Media. Furthermore, Calendar and Music topics showed significant improvement across all three languages with the ISO-15919 model.

Table 5: Comparison of ISO-15919 and IndicSOUNDEX model performance w.r.t baseline on low resource languages. '+' indicates degradation, '-' indicates improvement. '*' denotes the results that are not statistically significant

Language	Model	Topics improved	Average improvement	Topics degraded	Average degradation	Overall change
Tamil	IndicSOUNDEX	15	11%	1	7%	-7.6%
	ISO-15919	13	13%	2	2%	-6.2%
Marathi	IndicSOUNDEX	7	18%	6	8%	-2.30%
	ISO-15919	9	10%	6	11%	-1.50%
Telugu	IndicSOUNDEX	4	15%	5	7%	-0.20*%
	ISO-15919	4	15%	4	8%	-2.4%

Table 6: Percentage change in SemER for candidate models w.r.t baseline model across languages. '+' indicates degradation, '-' indicates improvement

Language	% Change in IndicSOUNDEX	% Change in ISO-15919
Hindi	-0.07%	+1.08%
Tamil	-7.6%	-6.2%
Marathi	-2.3%	-1.5%
Telugu	-0.2%	-2.4%

In Table 6, we provide the relative change in performance w.r.t baseline of all the models across high and low resource languages.

5 Conclusion

In this work, we explored the effect of using alternate representations of text for IC and NER models on a high-resource Indic language (Hindi) and three low-resource Indic languages (Tamil, Telugu, Marathi). We adopted a neural network based model architecture, where the alternate representations are incorporated in the embedding layers.

Based on the performance analysis over the baseline models, we saw that the alternate representations, while helping specific topics, do not help as much for the high-resource language overall. This is possibly due to the relatively high consistency in transcription and the existence of various tokens with similar meaning and sound in the training data. However, they helped significantly boost performance in the low-resource languages, thus creating potential applications in bootstrapping new languages quickly and cost-effectively.

In the case of the low-resource languages we saw significant improvements overall when using either ISO-15919 or IndicSOUNDEX. This suggests that smaller datasets stand to benefit from the use of alternative representations. Based on Tamil and Marathi results, where IndicSOUNDEX performed better than baseline and ISO-15919, we can conclude that the mitigation of the impact of multiple different scripts and inconsistent Latin usage accomplished by IndicSOUNDEX seems to produce better results for our NLU models. The lack of impact of IndicSOUNDEX for Telugu merits further investigation.

Our future work includes exploring different model architectures including transformer models, exploring these representations further by pre-training models with a combination of original tokens and alternate representations of tokens. We also plan to explore the use of character bigrams (of both original text and alternate representations of text) instead of unigram characters in the embedding layers.

6 Acknowledgments

We would like to thank Karolina Owczarzak for her support and invaluable discussions on this work and Wael Hamza for useful discussions. We would also like to thank our extended team mates for helpful discussions and anonymous reviewers for valuable feedback.

References

Beider, A. and Morse, S. P. (2010). Phonetic matching: A better soundex. *Association of Professional Genealogists Quarterly*.

Chaware, S. M. and Rao, S. (2012). Analysis of phonetic matching approaches for indic languages.

Geetha, P., Chandu, K., and Black, A. W. (2018). Tackling code-switched NER: Participation of cmu. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 126–131.

Johnson, A., Karanasou, P., Gaspers, J., and Klakow, D. (2019). Cross-lingual transfer learning for Japanese named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 182–189.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Shah, R. and Singh, D. K. (2014). Improvement of soundex algorithm for indian languages based on phonetic matching. *International Journal of Computer Science, Engineering and Applications*, 4(3).

Su, C., Gupta, R., Ananthakrishnan, S., and Matsoukas, S. (2018). A re-ranker scheme for integrating large scale nlu models. *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 670–676.

Zhang, X. and LeCun, Y. (2017). Which encoding is the best for text classification in chinese, english, japanese and korean? *CoRR*.

A Appendix A. Results across different topics on all languages

Table 7: Results for Hindi. Relative change in performance w.r.t baseline on the average of ten bootstrapped test data sets. '+' indicates degradation, '-' indicates improvement

Topic	**% Change in IndicSOUNDEX**	**Is Significant**	**% Change in ISO-15919**	**Is Significant**
Reservations	-1.64%	NO	-0.65%	NO
Books	-0.43%	NO	-5.18%	NO
Calendar	-7.59%	NO	-7.05%	NO
CallingAndMessaging	-4.36%	NO	-0.08%	NO
News	+2.78%	NO	+0.92%	NO
Photos	-6.85%	NO	-12.16%	NO
Media	+7.79%	NO	+11.13%	YES
Global	-2.89%	YES	-4.09%	YES
Help	+6.52%	NO	+5.76%	NO
SmartHome	-5.58%	YES	-4.32%	YES
QuestionAndAnswer	+3.10%	NO	+4.99%	YES
Search	-3.98%	NO	+3.78%	NO
Music	-0.45%	NO	+2.18%	YES
Notifications	+2.74%	NO	-2.49%	NO
OriginalContent	-3.66%	NO	-1.63%	NO
Recipes	-0.44%	NO	-0.39%	NO
Shopping	+5.36%	YES	-0.82%	NO
Sports	0%	NO	0%	NO
ToDoLists	-16.50%	YES	+1.08%	NO
Video	-2.73%	NO	-4%	YES
Weather	-3.89%	NO	+16.36%	YES
Overall	-0.07%	NO	+1.08%	YES

Table 8: Results for Tamil. Relative change in performance w.r.t baseline on the average of ten bootstrapped test data sets. '+' indicates degradation, '-' indicates improvement

Topic	**% Change in IndicSOUNDEX**	**Is Significant**	**% Change in ISO-15919**	**Is Significant**
Books	-14.06%	YES	-18.28%	YES
Calendar	-8.52%	YES	-11.93%	YES
MovieTimes	-7.91%	YES	-7.24%	NO
CallingAndMssaging	-12.44%	YES	-6.38%	YES
News	-15.50%	YES	-6.14%	YES
Media	-12.23%	YES	0%	NO
Global	-4.20%	YES	-4.41%	YES
Help	-20.84%	YES	-17.72%	YES
SmartHome	-7.29%	YES	+0.05%	YES
Search	+7.15%	YES	+3.08%	NO
Music	-8.03%	YES	-8.08%	YES
Notifications	-10.98%	YES	-6.81%	YES
OriginalContent	-9.92%	YES	-23.21%	YES
Shopping	-16.22%	YES	-13.44%	YES
Sports	+8.01%	NO	-30.10%	YES
ToDoLists	-2.14%	NO	+3.03%	YES
Video	-4.01%	YES	-14.65%	YES
Weather	-10.23%	YES	-10.91%	YES
Overall	-7.56%	YES	-6.23%	YES

Table 9: Results for Marathi. Relative change in performance w.r.t baseline on the average of ten bootstrapped test data sets. '+' indicates degradation, '-' indicates improvement

Topic	**% Change in IndicSOUNDEX**	**Is Significant**	**% Change in ISO-15919**	**Is Significant**
Books	-6.95%	YES	-11.81%	YES
Calendar	+9.28%	YES	-0.11%	YES
MovieTimes	-13.74%	YES	-11.03%	YES
CallingAndMessaging	+3.09%	YES	4.97%	YES
News	+16.76%	YES	+1.90%	YES
Media	-24.50%	YES	-12.84%	YES
Global	+0.13%	NO	-2.81%	YES
Help	-8.51%	YES	-0.76%	NO
SmartHome	-14.36%	YES	-5.04%	YES
Search	-0.63%	NO	-1.70%	YES
Music	-0.45%	NO	-4.66%	YES
Notifications	+2.49%	YES	+5.50%	YES
OriginalContent	-22.84%	YES	+8.10%	YES
Shopping	+2.97%	NO	+12.17%	YES
Sports	-38.48%	YES	-35.97%	YES
ToDoLists	+0.70%	NO	-0.23%	NO
Video	+5.25%	YES	+1.59%	NO
Weather	+12.29%	YES	+32.18%	YES
Overall	-2.29%	YES	-1.47%	YES

Table 10: Results for Telugu. Relative change in performance w.r.t baseline on the average of ten bootstrapped test data sets. '+' indicates degradation, '-' indicates improvement

Topic	**% Change in IndicSOUNDEX**	**Is Significant**	**% Change in ISO-15919**	**Is Significant**
Books	+2.42%	NO	+0.37%	NO
Calendar	+4.51%	YES	-10.33%	YES
MovieTimes	-13.57%	YES	-8.83%	YES
CallingAndMessaging	+5.41%	YES	+3.79%	YES
News	+0.97%	NO	+15.24%	YES
Media	-13.91%	YES	-0.61%	NO
Global	+2.93%	YES	+3.57%	NO
Help	+12.35%	YES	-0.45%	NO
SmartHome	-0.15%	NO	+0.31%	NO
Search	-7.41%	YES	+2.90%	YES
Music	+1.02%	NO	-4.55%	YES
Notifications	+1.33%	NO	-4.15%	NO
OriginalContent	-3.94%	NO	+2.10%	NO
Shopping	+0.55%	NO	+1.59%	NO
Sports	-4.43%	NO	-14.31%	NO
ToDoLists	+10.44%	YES	+10.05%	YES
Video	-0.72%	NO	-0.97%	NO
Weather	-25.98%	YES	-35.37%	YES
Overall	-0.21%	NO	-2.42%	YES