DCU@FIRE-2014: Fuzzy Queries with Rule-based Normalization for Mixed Script Information Retrieval

Debasis Ganguly
Centre for Global Intelligent Computing (CNGL)
School of Computing
Dublin City University
Dublin, Ireland
dganguly@computing.dcu.ie

Santanu Pal
Institute for Applied Linguistics
University of Saarland
Saarbruecken, Germany
santanu.pal@uni-saarland.de

Gareth J.F. Jones
Centre for Global Intelligent Computing (CNGL)
School of Computing
Dublin City University
Dublin, Ireland
gjones@computing.dcu.ie

ABSTRACT

We describe the participation of Dublin City University (DCU) in the FIRE-2014 transliteration search task (TST). The TST involves an ad-hoc search over a collection of Hindi film song lyrics. The Hindi language content of each document in the collection is either written in the native Devanagari script or transliterated in Roman script or a combination of both. The queries can be in mixed script as well. The task is challenging primarily because of the vocabulary mismatch which may arise due to the multiple transliteration alternatives. We attempt to address the vocabulary mismatch problem both during the indexing and retrieval stages. During indexing, we apply a rule-based normalization on some character sequences of the transliterated words in order to have a single representation in the index for the multiple transliteration alternatives. During the retrieval phase, we make use of prefix matched fuzzy query terms to account for the morphological variations of the transliterated words. The results show significant improvement over a standard baseline query likelihood language modelling (LM) approach. Additionally, we also apply statistical machine transliteration to train a transliteration model in order to predict the transliteration of out-of-vocabulary words. Surprisingly, even with satisfactory transliteration accuracy, we found that automatic transliteration of query terms degraded retrieval effectiveness.

Categories and Subject Descriptors

H.3.1 [**INFORMATION STORAGE AND RETRIEVAL**]: Content Analysis and Indexing—*Abstracting methods*; H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval—*Retrieval models, Relevance Feedback, Query formulation*

General Terms

Theory, Experimentation

Keywords

Fuzzy Query, Rule-based Normalization, Statistical Machine Transliteration

1. INTRODUCTION

Generally speaking, mixed script information retrieval (MSIR) refers to the problem of retrieving relevant documents for ad-hoc search queries, where the textual content of the documents in the collection and of the queries can be represented with more than one script, one native to the language of the document while the others being non-native [1]. Cross script information retrieval (CSIR) represents a special case for MSIR where the queries and the documents are in a single script but different from each other. The transliterated search task (TST) at FIRE (Forum of Information Retrieval Evaluation) 2014 is a shared task to establish benchmark retrieval methodologies for the MSIR research problem. The document collection for TST comprises Hindi song lyrics written both in the Devanagari script and the Roman script. Queries are keyword based and can also be either in Devanagari or Roman script.

The problem of MSIR is particularly hard because of the following reasons. First, the presence of script mixing in the documents and the queries may necessitate different indexing and retrieval strategies for the terms in two different types of scripts. For example, the process of stemming to address the morphological variations of the terms will be different for the native and the foreign script.

Second, the transliteration process of a term in the foreign script usually involves multiple alternatives due to linguistic and cultural differences. For example, the Hindi word "पहला" (EN: "first")[1] written in Devanagari script, the native script of Hindi, may be transliterated into the Roman script as "pehla", "pehlaa", "pahla", "pahlaa" etc. These multiple alternatives can give rise to a vocabulary mismatch problem between queries and documents, e.g. if one of the query terms is "पहला", transliterating this into "pehla" will not be able to retrieve documents that contain any other variants, e.g. "pahla".

An overview of our approach to mitigate this vocabulary mismatch problem is as follows. First, we use a rule-based character sequence normalization to normalize ambiguous character sequences into one unique representation across

[1]Throughout this paper, we write the English meaning corresponding to a non-English word within a pair of parenthesis following that word prefixed with "EN:".

the collection. As an example, the alternative options for transliterating the diacritic Hindi vowel "ा" are "a", "A" or "aa". Case folding leaves open two options eliminating one. This ambiguity can be handled by a rule which replaces all ocurrences of "aa" in the corpus to "a" as a pre-processing step before indexing. This character sequence normalization is also performed over the query terms during the retrieval phase.

Second, during retrieval time, we make use of fuzzy query terms to help alleviate the vocabulary mismatch. Specifically, a fuzzy query term allows at most 2 suffix characters to be different in order to consider a match as a valid one, e.g. "tera" (EN: yours) matches with "tere" (EN: your), whereas "mera" (EN: my) does not match with "tera" since the mismatch occurs in the prefix instead of the suffix. We hypothesize that in the absence of a stemmer for the transliterated words, such a scheme of prefix-biased approximate matching may potentially work well to bridge the vocabulary gap.

The rest of the paper is organized as follows. In Section 2, we describe the different indexing approaches to bridge the vocabulary gap including the dictionary construction and the rule based normalization. Section 3 describes our retrieval phase processing including query expansion, approximate matching and query transliteration. This is followed by Section 4, where we investigate the individual contributions from each of the proposed approaches and present various results obtained with on the set of training queries. We also provide an overview of the official results of the TST 2014 obtained with the test set queries. Finally, Section 5 concludes the paper with directions for future work.

2. INDEXING APPROACH

In this section, we describe the two approaches using which we hoped to alleviate the vocabulary mismatch problem in MSIR. We first describe a dictionary based document expansion approach, following which we describe a rule based character sequence normalization method.

2.1 Dictionary based Term Expansion

The document collection in the TST'14 task is comprised of about $60K$ documents constituted of song lyrics. The text in a large number of these documents is simply a concatenation of the song lyrics in two scripts, as a result of which it is a straight forward process to construct a bi-directional dictionary of words from the native script to the foreign script representation of a word and vice versa. An example of such a document with mixed script content is shown in Figure 1.

A one-one mapping can hence be constructed in a straightforward way using the information from these mixed script documents, e.g. "aa.Ndhii" is mapped to "आँधी", "chalii" is mapped to "चली" and so on. The dictionary that we constructed in this way has $21,018$ word mappings. Table 1 shows 10 words from this dictionary. Note that in the absence of such a well aligned structure of documents, a general approach would be to use a pre-built dictionary. However, for this particular task this approach of constructing the dictionary makes sense.

The dictionary is then used to expand a document as follows. For every word of a document in native or foreign script, we check if its counterpart exists in the dictionary; if it does, we add this dictionary word in the complementary script to the document index. Note that due to the

aa.Ndhii chalii to naqshekafepaa nahii.n milaa
dil jisase mil gayaa wo dubaara nahii.n milaa
ham a.njuman me.n sabakii taraf dekhate rahe
apanii tarah se ko_ii akelaa nahii.n milaa
aawaaz ko to kaun samjhataa ki duur duur
Kaamoshiyo.n kaa dardshanaasaa nahii.n milaa
kachche gha.De ne jiit lii nadii cha.Dhii huii
majabuut kashtiyo.n ko kinaaraa nahii.n milaa
आँधी चली तो नक़्शएकफ़एपा नहीं मिला
दिल जिससे मिल गया वो दुबार नहीं मिला
हम अंजुमन में सबकी तरफ़ देखते रहे
अपनी तरह से कोई अकेला नहीं मिला
आवाज़ को तो कौन सम्झता कि दूर दूर
ख़ामोशियअॅ का दर्दशनासा नहीं मिला
कच्चे घड़े ने जीत ली नदी चढ़ी हुई
मजबूत कश्तियअॅ को किनारा नहीं मिला

Figure 1: An example mixed script document from the TST collection which shows the same content written in both native and foreign script.

inherent transliteration ambiguity, some of the native script words in our dictionary may point to a set of corresponding foreign script words instead of a single one. For example, the dictionary entry "पहला" (EN: first) would point to the set {"pehla", "pehlaa", "pahla", "pahlaa"}. We ensure that in such a case, we always choose the lexicographically least one, which for this particular example is "pahla". We also convert each occurrence of the other transliteration variants into this representative class member, e.g. "pehla" is converted to "pahla" and so on. This dictionary-based cross-script term expansion is conducted on the query side as well.

Table 1: An extract from the automatically constructed dictionary using a simple one-one mapping.

Devanagari	Roman
रखकर	rakhakar
चारासाज़	chaaraasaaz
पट्के	patke
सजाना	sajaanaa
खैरात	khairaat
पखावज	pakhaavaj
पट्का	patkaa
कमसीन	kamasiin
कब्ज़े	kabze
सजाने	sajaane

2.2 Rule-based Normalization

The dictionary is only able to handle those words which can be aligned due to the presence of transliterated content within the same document. To reduce the term mismatch problem for out-of-dictionary words, we employed a simple rule-based character sequence normalization method to create a single equivalent representation for the ambiguous character sequences in the index. This process is also performed on the query side.

Table 2 lists the rules that we applied while constructing the index along with illustrative examples. All the rules are applied iteratively, as a result of which a word can go through a sequence of multiple intermediate transformations before eventually reaching the final normal form, e.g. according to the rules of Table 2, "<u>dh</u>oom" → d<u>oo</u>m → d<u>um</u> → d<u>am</u>.

Table 2: Rules applied for normalizing alternative transliterations.

Found	Substituted	Example
aa	a	laagan → lagan
ay	ai	sapnay → sapnai
ae	ai	sapnae → sapnai
ii	i	mahii → mahi
ee	i	mahee → mahi
oo	u	pooja → puja
uu	u	huzuur → huzur
q	k	qayamat → kayamat
ia	ya	dooria → doorya
hh	h	chhaya → chaya
v	w	havas → hawas
bh	b	bharat → barat
cch	c	iccha → ica
ch	c	chaya → caya
gh	g	ghungru → gungru
jh	j	jharoka → jarokha
sh	s	shaan → saan
th	t	hathi → hati
dh	d	dhoom → doom
um	am	hum → ham
ain	ai	main → mai
dh	ai	main → mai

2.3 Index Generation

Previous research in probabilistic and language models of IR has shown that using field specific models for IR usually out-performs flat ones, mainly because the document term frequency maximum likelihood estimates are more reliable on a per-field basis than on a global one [6, 2]. In our case, since a large number of documents in the collection are comprised of mixed script content, separating the content of the document into two individual fields, one for the native script and the other for the foreign enables us to compute document term frequency and collection statistics relative to each individual field. Moreover, each document in the collection has a title and a body. It is fairly intuitive that a match of a query term in the title of a song lyric may bear more importance than a match in the body of the song. To achieve this, we further partitioned the content of each script field (native or foreign) of each document in two more additional fields, namely the title and the body.

A document in our index thus manifests itself as a set of four individual fields, namely the Devanagari title (HN_Title), Devanagari body (HN_Body), Roman title (EN_Title) and Roman body (EN_Body). The index is generated with the help of Lucene[2], a widely used freely available Java toolkit for indexing and retrieval. In the case of song lyric search the order of the query terms may in fact be quite important, e.g. "diwana dil" and "dil diwana" refers to two different songs. In order to take the term order into account, along with the unigrams we also index the word bi-grams with the help of the ShingleFilter[3] utility of Lucene.

Table 3 shows the index statistics for each index constructed without and with the pre-processing steps described in Section 2.1 and 2.2. It can be observed that the vocabulary is much more diverse for the transliterated content (EN_Title and EN_Body) than the native script content (HN_Title and HN_Body) due to the inherent ambiguities in the transliteration. Character sequence normalization helps to reduce this diversity by grouping together words into equivalent classes and using only a single class member from each of these equivalent classes as the indexing units. Transliteration helps to expand the document lengths by adding equivalent cross-script word representations. The indexing strategy undertaken in the last row of Table 3 is expected to provide the best results due to the combination of the two strategies for mitigating vocabulary mismatch.

3. RETRIEVAL APPROACH

The retrieval model that we use for all the IR experiments reported in this paper is the language model (LM) query likelihood [2]. The parameter λ (the importance of term presence) was set to 0.3 after tuning on the development set queries. The subsequent retrieval experiments use this value of λ, i.e. 0.3. In the following sections, we describe two retrieval time techniques of mitigating the vocabulary mismatch between documents and queries.

3.1 Approximate Query Term Matching

Stemming plays an important role in normalizing the morphological variations of a word, e.g. "friendly" and "friends" are represented by the root word "friend" [5]. However in the context of MSIR, stemming words written in the foreign script is difficult because of the multiple alternatives inherent in these suffixes. We therefore undertake a simple approach of approximately matching the query terms with the index terms in the inverted list instead of exact matches. The utility class of Lucene that we make use of for this is the "FuzzyQuery"[4]. For efficiency reasons, edit distances of up to 2 are allowed in the approximate matching. The prefix length value that must match for two terms to match approximately is set to $\tau \times max(len(t_d, t_q))$, where t_d is an index term that we seek to match with the query term t_q, $len(.)$ is the length function and τ is a parameter ($\tau \in [0, 1]$). We set the value of $\tau = 0.7$ for all the experiments reported in this paper involving fuzzy term matching.

3.2 Automatic Transliteration

For some query terms we expect to find an equivalent cross-script representation exists in the dictionary. This cross-script representation for the current query term is added to the query. For out-of-dictionary terms, we employ a statistical machine translation (SMT) approach to automatically predict the transliteration of unseen words. The similarity of transliteration with translation can be observed from the fact that while SMT involves translation of a source language sentence to a target language one, automatic transliteration on the other hand involves transforming a source

[2]`http://lucene.apache.org/`

[3]`http://lucene.apache.org/core/4_9_0/analyzers-common/org/apache/lucene/analysis/shingle/ShingleFilter.html`

[4]`https://lucene.apache.org/core/4_6_0/core/org/apache/lucene/search/FuzzyQuery.html`

Table 3: Collection statistics from indexes obtained with different pre-processing.

Pre-processing before indexing	#Collection Frequency			
	EN_Title	HN_Title	EN_Body	HN_Body
No pre-processing	106,697	51,074	1,211,136	473,144
Normalization	78,331	51,074	804,834	473,144
Transliteration + Normalization	89,855	51,115	832,460	474,058

Table 4: Retrieval results obtained on the development set of 34 queries.

Parameters				Evaluation Metrics			
Normalization	Term Expansion		Fuzzy Query				
	Document-side	Query-side		MAP	MRR	Recall	BPREF
no	no	no	no	0.3623	0.7688	0.5576	0.5688
no	no	yes	no	0.2718	0.5907	0.6057	0.6628
no	no	no	yes	0.3829	0.7760	0.6201	0.6029
no	no	yes	yes	0.3294	0.6523	0.6923	0.7186
yes	no	no	no	0.4371	0.7593	0.6394	0.6153
yes	no	yes	no	0.4495	0.7898	0.6826	0.7160
yes	no	no	yes	0.4390	0.7455	0.6538	0.6365
yes	no	yes	yes	0.4546	0.7689	0.6971	0.7381
yes	yes	yes	no	0.5641	0.8400	0.7211	0.7430
yes	yes	yes	yes	**0.5671**	**0.8467**	**0.7548**	**0.7757**
yes	yes	yes (Dict. + Moses)	no	0.3988	0.6417	0.6737	0.7134
yes	yes	yes (Dict. + Moses)	yes	0.4131	0.6769	0.6875	0.7346

script character sequence to a target script character sequence.

Equivalent cross-script words from our dictionary (see Section 2.1) were used to constitute a list of parallel corpora (aligned at the character level), which in turn was used to train a transliteration model. For example, the word pair ("khushbuu", "खुशबू") from the dictionary is converted to the following character sequence pair.

k h u s h b u u → ख ◌ु श ब ◌ू

The goal of the alignment function is to learn the correct mapping between the character sequences, which in this case is "k h" → ख, "u" to ◌ु, "s h" to श and "b u u" to बू.

Our experimental settings for training the transliteration model were:

1. Log-linear phrase based statistical machine translation (PB-SMT) model [3], implemented in "Moses"[5].
2. Maximum phrase length of 5 in translation model and 3-gram (character sequence) for language model.
3. GIZA++ implementation of IBM alignment model with grow-diagonal-final-and heuristics for performing source target character alignment and phrase-extraction [4].
4. No reordering performed and distortion parameter word penalty set to zero.

The accuracy of our transliteration model was measured with the help of 10-fold cross validation. The average precession, recall and F measure values are shown in Table 5. Table 5 shows that we achieve satisfactory transliteration accuracy (measured by the BLEU score), the results being better from foreign to native script than the other way round because of the inherent transliteration ambiguity. Since, we obtain more accurate transliterations from the foreign to the native script, we use only this direction for query expansion. More precisely, given an out-of-dictionary Roman script transliterated word, we transliterate this word back to the native script and add this term to the query.

The column named "Character" in Table 5 indicates that the output character sequences are evaluated with respect to the character sequences of the reference (test set) data. The "Word" column on the other hand indicates that the evaluation is conducted at the word level that is by removing the intermediates spaces between the character sequences of the output and the reference words.

Table 5: 10-fold cross validation results on transliterated word prediction.

	EN-HI		HI-EN	
	Character	Word	Character	Word
Precision	0.9177	0.8806	0.9312	0.8109
Recall	0.9293	0.8780	0.9073	0.8109
F-score	0.9235	0.8793	0.9191	0.8109
BLEU	0.7494	0.6185	0.6470	0.4295

4. RESULTS

In this section, we report the results of our experiments. In order to investigate the effectiveness of various indexing and retrieval approaches proposed in Sections 2 and 3, we first report the results obtained on the training (development) set of queries with an incremental application of each of these approaches. We then select the best experimental settings and apply them to the test set, the approach which we in fact used for the official submissions. The software used for the indexing and retrieval experiments reported in this paper is made publicly available[6].

[5]http://www.statmt.org/moses/

[6]`https://github.com/gdebasis/msir/`

4.1 Development Set Results

In order to compare our results with that of [1], we use the binary relevance judgments for evaluation. Similar to [1], we consider the documents with manually assessed scores of over 2 (i.e. 3 and 4) out of a 5-point scale from 0 to 4 as relevant. We retrieve 1000 documents for each query and all our reported results are calculated based on this ranked list of 1000 documents. The total number of queries in the development set is 35, out of which one query has no relevant documents (i.e. assessed scores of above 2). We therefore consider 34 queries for our evaluation.

Table 4 shows the results. It can be observed that according to our hypothesis, the best results are obtained when we use a combination of dictionary-based term expansion on both the document and the query-side coupled with rule based normalization and approximate query term matching. It is somewhat surprising to see that despite the good performance of the transliteration model trained with Moses, expanding the out-of-dictionary query terms with automatically transliterated words degrades retrieval quality.

In order to directly compare our results with [1], we compare our best performing approach (best settings of Table 4) with the best results reported in [1]. For this, we evaluate our approach only on the first 25 queries constituting of terms only in the foreign script (the same test set used for the TST'13 task at FIRE 2013 [7] and [1]). The results are shown in Table 6. We see that our approach produces second best results only with a simple term based approach without modeling the equivalence of sub-word features, e.g. character n-grams with the help of auto-encoders [1].

Table 6: Comparative evaluation on top 10 retrieved documents with previously reported results.

Run Description	Evaluation Metrics	
	MRR	MAP
TUVal-2 [7]	0.8440	0.4240
Deep [1]	**0.8740**	**0.5039**
Our Best Approach (Table 4)	0.8500	0.4793

4.2 Test Set (Official) Results

We submitted two runs in TST '14. The first run applied dictionary based alignment to expand index and query terms along with rule based normalization for indexing and the fuzzy query match for retrieval. The second run applied PBSMT based query term transliteration (see Section 3.2). In Table 7, we report the results obtained on the 35 test queries.

An interesting observation is that the MAP and the MRR values are considerably lower for the test set queries in comparison to the training set ones (c.f. Table 4 and Table 7). In order to investigate what might have caused this, we report the per-query retrieval results for the test set queries. We observed that 8 out of 35 queries produced very low average precision (AP) values of less than 0.1. In fact two of the queries produced a zero AP value. On manual inspection of one of these queries (shown in Table 8), we found that the reason for the bad performance is due to the compounding effect (shown with underlined text). It can be seen that "ek taara" occurs as two distinct words in the query, whereas in both the relevant documents these terms have been juxtaposed together to form a compound, as a result of which our system could not perform well on these types of queries in general.

Table 7: Evaluation on the test set with 10 and 1000 documents retrieved.

	#ret=10		#ret=1000	
Run Description	MRR	MAP	MRR	MAP
Normalization + Doc. & Qry. exp. (dict. based) + fuzzy qry. match	**0.6408**	**0.4040**	**0.6455**	**0.4701**
Normalization + Doc. & Qry. exp. (dict. based) + fuzzy qry. match + Qry. exp. (Moses)	0.4265	0.2633	0.4315	0.3257

Table 8: Compounding effect causes poor results for some queries.

Query	#	Relevant Document Title
koi <u>ek taara</u> <u>ek taara</u>	1	इकतारा - <u>ikataaraa</u>
	2	ओ रे मनवा... इकतारा (महिला स्वर) - o re manawaa... <u>ikataaraa</u> (mahilaa swar)

5. CONCLUSIONS AND FUTURE WORK

We described our participatory work for the transliterated search task (TST) at FIRE 2014. We undertook a purely IR approach to the search task. The main motivation for our work was to mitigate the vocabulary mismatch between the documents and queries where both may be represented in mixed scripts from an IR perspective. Firstly, we make use of a dictionary based transliteration for document and query expansion. Secondly, we apply rule-based normalization to transform each equivalent character sequence into a single representative of its equivalence class. Thirdly, we employ a fuzzy term matching to account for the absence of a stemming algorithm for words in the foreign script. Further, we also applied a PBSMT system to learn the corresponding character sequence alignments across the source and the target scripts.

The results showed that a pure IR approach yields competitive results in comparison to a more sophisticated methods for detecting character sequence level alignments, namely using deep learning with auto-encoders [1]. Surprisingly, the results degraded with the use of PBSMT based query expansion. In future, we plan to investigate the possible reasons for this degradation.

Acknowledgments

This research is supported by Science Foundation Ireland (SFI) as a part of the CNGL Centre for Global Intelligent Content at DCU (Grant No: 12/CE/I2267).

6. REFERENCES

[1] P. Gupta, K. Bali, R. E. Banchs, M. Choudhury, and P. Rosso. Query expansion for mixed-script information retrieval. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 677–686. ACM, 2014.

[2] D. Hiemstra. *Using Language Models for Information Retrieval*. PhD thesis, CTIT, AE Enschede, 2000.

[3] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180. Association for Computational Linguistics, 2007.

[4] P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54. Association for Computational Linguistics, 2003.

[5] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[6] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 42–49. ACM, 2004.

[7] R. S. Roy, M. Choudhury, P. Majumder, and K. Agarwal. Overview and datasets of fire 2013 track on transliterated search. In *Working Notes on FIRE 2013*, FIRE '13, 2013.