Query Expansion for Multi-script Information Retrieval

Parth Gupta*
PRHLT Research Center
Universitat Politècnica de València
Valencia, Spain
pgupta@dsic.upv.es

Kalika Bali
Microsoft Research Labs India
"Vigyan", 9 Lavelle Road
Bangalore, India
kalikab@microsoft.com

Rafael E. Banchs
Institute for Infocomm Research
Singapore 138632
rembanchs@i2r.a−star.edu.sg

Monojit Choudhury
Microsoft Research Labs India
"Vigyan", 9 Lavelle Road
Bangalore, India
monojitc@microsoft.com

Paolo Rosso
PRHLT Research Center
Universitat Politècnica de València
Valencia, Spain
prosso@dsic.upv.es

ABSTRACT

For many languages that use non-Roman based indigenous scripts (e.g., Arabic, Greek and Indic languages) one can often find a large amount of user generated transliterated content on the Web in the Roman script. Such content creates a monolingual or multi-lingual space with more than one script which we refer to as the *Multi-Script space*. IR in the multi-script space is challenging because queries written in either the native or the Roman script need to be matched to the documents written in both the scripts. Moreover, transliterated content features extensive spelling variations. In this paper, for the first time, we formally introduce the concept of *Multi-Script IR*, and through analysis of the query logs of Bing search engine, estimate the prevalence and thereby establish the importance of this problem. We also give a principled solution to handle the multi-script term matching and spelling variation where the terms across the scripts are modelled jointly in a deep-learning architecture and can be compared in a low-dimensional abstract space. We present an extensive empirical analysis of the proposed method along with the evaluation results in an ad-hoc retrieval setting of multi-script IR where the proposed method achieves significantly better results (12% increase in MRR and 29% increase in MAP) compared to other state-of-the-art baselines.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

*Except this author, all authors are sorted alphabetically.

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.

http://dx.doi.org/10.1145/2600428.2609622 .

Keywords

Multi-script information retrieval, transliteration, deep-learning

1. INTRODUCTION

A large number of languages, including Arabic, Russian, and most of the South and South East Asian languages, are written using indigenous scripts. However, due to various socio-cultural and technological reasons, often the websites and the user generated content in these languages, such as tweets and blogs, are written using Roman script [1]. Such content creates a monolingual or multi-lingual space with more than one scripts which we refer to as the *Multi-Script space*. Information retrieval in the multi-script space, which can be termed as *Multi-Script IR* (MSIR), is challenging because queries written in either the native or the Roman scripts need to be matched to the documents written in both the scripts.

The process of phonetically representing the words of a language in a non-native script is called *transliteration* [19]. Transliteration, especially into Roman script, is used abundantly on the Web not only for documents, but also for user queries that intend to search for these documents. Since there are no standard ways of spelling a word in a non-native script, transliteration content almost always features extensive spelling variations; typically a native term can be transliterated into Roman script in very many ways [1, 11]. For example, the word *pahala* ("first" in Hindi and many other Indian languages) can be written in Roman script as *pahalaa, pehla, pahila, pehlaa, pehala, pehalaa, pahela, pahlaa* and so on.

This phenomenon presents a non-trivial term matching problem for search engines to match the native-script or Roman-transliterated query with the documents in multiple scripts taking into account the spelling variations. The problem of MSIR, although prevalent in Web search for users of many languages around the world, has received very little attention till date. There have been several studies on spelling variation in queries and documents written in a single (native) script [14, 34, 9] as well as transliteration of named entities (NE) in IR [3, 31, 33]. However, as we shall see in this paper, MSIR presents challenges that the current approaches for solving mono-script spelling variation and NE transliteration in IR are unable to address adequately, especially because most of the transliterated queries (and

documents) belong to the *long tail* and hence do not have enough clickthrough evidence to rely on.

In this paper, for the first time, we formally introduce the problem of MSIR and related research challenges. In order to estimate the prevalence of transliterated queries, we also analyse a large query log of Bing[1] consisting of 13 billion queries issued from India. As many as 6% of the unique queries have one or more Hindi words transliterated into Roman scripts, of which only 28% queries are pure NEs (people, location and organization). On the other hand, 27% of the queries belong to the entertainment domain (names of movies, song titles, parts of lyrics, dialogues, etc.), which provide complex and ideal examples of transliterated queries. Hindi song music is also one of the most searched items in India[2] and thus, a perfect and practical case for MSIR. This motivated us to conduct our MSIR studies on Hindi song lyrics.

We propose a principled solution to handle the multi-script term matching and spelling variation where the terms across the scripts are modelled jointly. We model the multi-script features jointly in a deep-learning architecture in such a way that they can be compared in a low-dimensional abstract space. The proposed method can find the equivalents of the query term across the scripts; the original query is then expanded using the thus found equivalents. Through rigorous experiments on MSIR for Hindi film lyrics, we further establish that the proposed method achieves significantly better results compared to all the competitive baselines with 12% increase in MRR and 29% increase in MAP over the best performing baseline.

The concrete contributions of this paper are two-fold as listed below:

1. The concept, formal definition and representative analysis of MSIR for Web search,

2. The multi-script joint modelling technique using deep autoencoder.

The rest of the paper is organized as follows. In Section 2, we introduce the notion of MSIR formally and outline the possible applications scenarios and research challenges. Section 3 presents the empirical analysis of Bing search engine's query log, and the prevalence and distribution of transliterated Hindi queries. Section 4 presents our deep-learning based joint script modelling approach to MSIR. In Section 5 the experimental setup and results are presented along with extensive empirical analysis. Finally, in Section 6 we present the related work and we make the concluding remarks in Section 7.

2. MSIR: DEFINITION & CHALLENGES

In this section, we formalize the notion of *Multi-Script Information Retrieval* along the lines of Crosslingual IR (CLIR). We also list out a set of research challenges in the context of MSIR.

2.1 Languages, Scripts and Transliteration

Let $\mathcal{L}$ be a set of (natural) languages $\{l_1, l_2, \ldots, l_n\}$. We assume that every language is generally written using a particular script, which we will refer to as the *native script* of the language. Let s_i be the *native script* for language l_i. Thus, the set of scripts $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ has a one-to-one mapping to $\mathcal{L}$.

Any natural language word (or more generally any text fragment) w has two attributes - the language it belongs to and the script it is written in. We use the notation $w \in \langle l_i, s_j \rangle$ to imply that w is in language l_i, written using the script s_j. When $i = j$, we say that w is in native script. Else, we say that w is in *transliterated form*, where *transliteration* can be defined as the process of loosely or informally representing the sound of a word of one language, l_i using a non-native script s_j.

Note that a particular language might be traditionally written in more than one script. For instance, Kurdish is written using the Roman, Cyrillic and Arabic scripts. However, such cases are rare. On the other hand, it is very common to use a script for writing several languages. For instance, the Roman script (with slight variations or additions of diacritics) is used to write English, French, German, Italian, Turkish and many other languages around the world. Similarly, the Devanagari script is used for writing Hindi, Sanskrit, Nepali and Marathi languages. Our definition does not preclude such a possibility, but we would like to emphasize that it is useful to treat the same script differently when used for writing different languages because the same sequence of letters might have different pronunciations in different languages. Consequently, transliterating a word of l_i (say Hindi) into the scripts s_j (say, Roman script as used in English orthography) and s_k (say, Roman script as used in French orthography) could yield very different results, even though the two scripts use almost an identical alphabet.

2.2 Multi-script IR

Given a query q and a document pool $\mathcal{D}$, the task of an IR engine is to rank the documents in $\mathcal{D}$ such that the ones relevant to q appear at the top of the ranked list. Depending on the language in which q and $\mathcal{D}$ are presented, one can define two basic kinds of IR settings. Without loss of generality, let us assume that $q \in \langle l_1, s_1 \rangle$. In *monolingual IR*, $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ consists of only those documents that are in the same language and script, i.e., for all k, $d_k \in \langle l_1, s_1 \rangle$. This simple scenario is modified in the context of CLIR, where

$$\mathcal{D} = \bigcup_{i=1\ldots n} \mathcal{D}_i$$

where $\mathcal{D}_i = \{d_{i,1}, d_{i,2}, \ldots, d_{i,N}\}$ are documents in language l_i, i.e., for all k, $d_{i,k} \in \langle l_i, s_i \rangle$. Note that all the documents in a typical CLIR setup are assumed to be written in the corresponding native scripts.

Based on this fundamental idea of CLIR, we can define a corresponding *Multi-script IR* (MSIR) setup as follows. Let $q \in \langle l_1, s_j \rangle$ be a query, where j may or may not be equal to 1. The document pool,

$$\mathcal{D} = \bigcup_{k=1\ldots n} \mathcal{D}_{1,k}$$

where $\mathcal{D}_{1,k} = \{d_{1,k,1}, d_{1,k,2}, \ldots, d_{1,k,N}\}$ are documents in language l_1 written in script s_k, i.e., for all m, $d_{1,k,m} \in \langle l_1, s_k \rangle$. In other words, in the MSIR setup, the query and the documents are all in the same language, say l_1, but they are written in more than one different scripts. The task of the IR engine is to search across the scripts.

[1] `http://www.bing.com/`

[2] Zeitgeist 2010: India - `http://www.google.com/intl/en/press/zeitgeist2010/regions/in.html`

In the literature, sometimes CLIR is distinguished from Multilingual IR in the sense that the former refers to a case where $n = 2$, whereas the latter is a generalization to any $n > 2$. Likewise, for monolingual IR, n can be assumed to be 1. One could make a similar distinction between Mono-script, Cross-Script and Multi-Script IR scenarios, where the query and the documents are in one language, but in 1, 2 or more than 2 scripts respectively. Nevertheless, we will refer to both the latter cases as MSIR. All our experiments involve a single language, namely Hindi, and two scripts – Devanagari and the Roman script (English orthography), but the proposed approach can be easily extended to a larger set of scripts.

One can also further generalize the setup to *Multi-script Multilingual IR*, where q as well as $\mathcal{D}$ can be in one of the several languages written in one of the several scripts. This is also a useful and practical setup, though we will not discuss it any further in this work.

It should also be noted that like CLIR it is possible in the MSIR setting that for $q \in \{l_i, s_j\}$, the information might be available only in a $d_{i,j,k}$ where $i \neq j$. In such cases, often the user issuing the query might be able to read and write both s_j and s_j and hence $d_{i,j,k}$ would have solved users info need. However, without MSIR this would not be possible to achieve.

2.3 Mixed and Transliterated Queries & Documents

The definition of MSIR setup assumes that the entire query and the each of the documents are in a single language and single script. However, in practice, one can find queries or documents that contain text fragments written in more than one language or script or both. Furthermore, depending on whether the parts of a query or document are written in a language using the native or a non-native script, one can have native or transliterated queries/documents.

A practical way to address the issue of mixed documents could be to split them into several sub-documents such that each of the sub-documents are in a single language and single script as discussed in [4] given mixing is not at sub-sentence level which falls under different case of code-mixing and out of the scope of this study. Mixed queries, however, cannot be handled through simple splitting because matching parts of a query to the documents does not make sense in the context of IR. Therefore, we extend our MSIR setup to include mixed queries. We define query q as a string of words $w_1 w_2 \dots w_m$, where $w_1 \in \langle l_{i_1}, s_{j_1} \rangle$, $w_2 \in \langle l_{i_2}, s_{j_2} \rangle$ and so on can all belong to different languages, or scripts or both.

2.4 Challenges in MSIR

The two primary challenges in MSIR are: (a) how to tackle the extensive spelling variations in the transliterated queries and documents during the term matching phase, and (b) how to identify, process and represent a mixed query (and also, the mixed and transliterated documents). In CLIR, there are broadly two approaches to model the crosslingual space – either documents and queries are *translated* to bring all words into the same monolingual space, after which monolingual IR techniques and matching algorithms can be directly applied [33], or the crosslingual space is modelled jointly as an abstract topic or semantic space, and documents and queries in all languages are mapped to this common space [5]. Likewise, in MSIR one can "transliterate" the text to bring everything into a common space and then apply standard matching techniques in the single-script space, or one can jointly model an abstract orthographic space for representing the words written in different scripts. In this work, we shall explore the latter, which we believe is a more robust and generic solution to the multi-script space modelling problem as it can simultaneously handle spelling variations in a single script and across multiple scripts. Nevertheless, we do recognize that the transliteration based approach is worth exploring, and machine transliteration, though a well studied problem [19], could present interesting and challenging research problems when applied in the context of Web scale IR.

Mixed query processing is another interesting research challenge, which includes language identification of the query words, which can be either in native or transliterated scripts, and labeling those with semantic or other tags (e.g., entities, attributes). This is challenging especially because depending on the context of the query, the same word, say "man", could represent the English word *man*, or a transliterated Hindi word *man* meaning "mind", or another transliterated Hindi word *maan* meaning "reputation". However, the same word with similar meanings are also used in many other Indian languages; and it can also have different connotations in other languages (e.g., in Bengali this could also mean "to get offended"). Hence, language identification seems to be an extremely challenging problem in the MSIR setting, especially when multiple languages are involved. In this work, we limit our experiments to only two languages, namely English and Hindi, and describe some initial results with language identification for transliterated and mixed queries.

Apart from these basic challenges, result presentation in MSIR is also an interesting problem because this requires the information on whether the user can read all the scripts, or prefer some scripts over other. There are no user studies related to MSIR and it is ripe with several such open problems.

3. TRANSLITERATED QUERIES IN WEB SEARCH

Although the current Web search engines do not support MSIR, they still have to handle a large traffic of mixed and transliterated queries from linguistic regions that use non-Roman indigenous scripts. We are not aware of any previous study on percentage of transliterated queries seen by the commercial search engines. Consequently, we do not know the distribution of transliterated queries across various topics and domains, which could provide deeper understanding of the MSIR space and its users. In this section, we present an analysis of mixed and transliterated queries extracted from a large query log of Bing. Our study relies on automatic identification and classification techniques for mixed queries that have been developed specifically for this analysis.

3.1 Methodology

We conducted the analysis on 13.78 billion queries sampled from the logs of Bing that were issued from India. India provides an interesting socio-linguistic context for studying mixed queries because of abundance of Roman transliteration and multiplicity of languages and scripts. This dataset consists of 30 million unique queries with an average length of 4.32 words per query. Almost all the queries (99.998%)

are in Roman script. For ease of computation, we randomly sampled 1% (i.e., 300,000) of the unique queries and conducted the study on this smaller sample.

We automatically identify the transliterated queries using a language classifier, which has been built as follows. We train a maximum entropy classifier using character n-grams for $n = 1$ to 5 as features for both Hindi and English words, which is based on a similar word-level language identification work by King and Abney [18]. The classifier was trained on 5000 frequent transliterated Hindi words from Bollywood song lyrics [11]. This dataset is freely available for research purpose. For English examples, we have taken 5000 frequent words from the Leipzig Corpus[3]. We define a query q to be mixed or transliterated if at least 40% of the words in q are classified as Hindi. We tested the performance of the classifier on a set of 2500 unseen words and the accuracy was found to be 97%. Note that the query log is expected to contain transliterated queries in other Indian languages as well. Due to a large shared vocabulary, lot of the Roman transliterated words in other Indic languages have almost similar spellings as in Hindi, and hence, we observe that our classifier is able to identify those as well. Nevertheless, our analysis is targetted primarily on Hindi transliterated queries and the actual fraction of transliterated queries, considering all Indic languages, can be expected to be much higher than the numbers revealed by this study.

After observing the transliterated queries pulled out by our method, we identified six broad categories or topics to which most of these queries belong: *Named Entities*, *Entertainment*, *Information Source*, *Culture*, *Recipe* and *Research*. Each of these were further refined into a set of subcategories; e.g., *Named Entities* can be of three types *people*, *location* and *organization*. Besides, we also observed a few interesting subcategories, which we put together under a catch-all seventh category – *Others*. Table 1 lists all these categories and sub-categories along with example queries.

In order to automatically classify the queries into these categories, we resort to a rather simple minimally supervised approach. Through manual inspection of the transliterated queries, we selected five representative and reasonably frequent examples for each sub-category. We extract all queries from our dataset that have at least one word in common with at least one of the five representative queries, and then we extract the top 100 most frequent words in this set of queries. The standard English stop words are then removed from these 100 words; we shall refer to the remaining words as the *cue words* for the particular subcategory. In this way, we obtain the cue words for each sub-category with very few overlaps, giving us a total of 180 such words. Some example cue words for each of the sub-categories are reported in Table 1.

Let c^j_1 to $c^j_{m_j}$ be the cue words associated with the j^{th} sub-category. For each of the transliterated queries $q = w_1 w_2 \dots w_m$ that we want to categorize, we remove all the stop words and cue words. For each of the remaining words in the query, say w_i, we count the number of queries, $f^j_{i,k}$, in the log where w_i co-occurs with the cue-word c^j_k. Also, let f_i be the number of queries in which w_i occurs. We compute the score of q with respect to a sub-category j as (if w_i is a stop or cue word then it is not considered for score computation):

$$score(q, j) = \sum_{i=1}^{k} \sum_{k=1}^{m_j} f^j_{i,k}/f_i$$

q is assigned to the sub-category j^* for which this score is maximum.

3.2 Observations

In our dataset, as much as 6% of the unique queries were identified as transliterated, which means that at least 40% of the words in these queries are Roman transliterations of Hindi words. The average query length for the transliterated queries is 2.86, which is less than the average query length of all queries, 4.32. The frequency of the transliterated queries are in general less than that of the non-transliterated ones. Hence, they only constitute about 0.011% of all the queries in our dataset. However, their frequency distribution follows the same power-law pattern as the regular queries, albeit spanning mainly the medium and low frequency spectra. This also implies that a large number of transliterated and mixed queries belong to the *long tail* of distribution and may not have enough clickthrough data to help a search engine process them accurately. They must be processed differently recognizing the fact that they are rare, but together they do form a sizeable mass of the search traffic.

Table 1 reports the percentage of the transliterated queries in each of the identified sub-categories. The numbers do not add to 100% because a small fraction, 18% of unique but only 2% of all, queries could not be mapped to any of the categories. It is not surprising that a large fraction of the queries are NEs. Along with *Websites*, NEs form 50% of the unique queries, though when query frequencies are taken into account NEs only constitute 6% of all queries. Consequently, processing of transliterated NEs has received some attention from the IR researchers [21]. *Entertainment* is the second largest category (27%), of which movies and songs are the most searched categories. These queries are typically longer and more complex than NE queries, and constitutes more than 32% of the transliterated query traffic. Yet, this category has hardly received any special attention from the researchers [6, 11]. We believe that *Entertainment* is a rich and practically important domain for MSIR, and hence we conduct our MSIR experiments on Hindi song lyrics dataset obtained from [11].

4. DEEP-LEARNING OF TERM EQUIVALENTS

Having defined the basic MSIR setting, and establishing its prevalence in Web search, we now present an approach for modelling the multi-script space that in turn allows us to develop a MSIR system.

4.1 Motivation

As discussed in Sec 2, the primary challenge in MSIR is to model and match the words across the scripts in the presence of a large number of spelling variations of the words, especially in the non-native script. We shall refer to the variants of the same *word* in the native and other scripts as *term equivalents*. The term matching problem could be addressed by using existing approaches such as approximate string matching [14, 34] and transliteration mining [30, 21, 20]. The former techniques can be used to handle the spelling variation in a single script (especially, variations in non-native script), while the latter can help in matching the

[3] `http://corpora.uni-leipzig.de/`

Table 1: Classification of transliterated Hindi queries. In the last column, we present the % of unique queries in each category followed by the % of total queries within parentheses. Transliterated words are italicized.

Category	Topics/sub-categories	Cue words	Example query	% of queries
Named Entity	People	mr, *ji*, *guru*, dr, *swami*	*harmohinder singh gogia*	6 (1.04)
	Organization	ltd, university, bank	*gandharva mahavidyalaya* ddu *marg*	14 (2.8)
	Location	*nagar*, *garh*, *chowk*, hotel	*rajdhani* train timings *chappra* to *guwhati*	8 (2.13)
Entertainment	Movie	movie, film, torrent, video	*himmatwaala* remake	7 (19.56)
	Song/Lyrics/Dialogues	album, tune, lyrics, audio	*ik din ayega* lyrics	18 (12.8)
	Tv soaps/serials	song, lyrics, tv, serial	colors *madhubala ishq ek junoon*	2 (0.62)
Information Source	Books	book, *pustak*, *kitab*	*bade ghar ki beti premchand*	0.005 (0.02)
	magazines/news	*patrika*, times,	*vasundhara eenadu*	3 (14.52)
	websites	blog, com, net , http	*swayamvaram* info	22 (44.18)
Culture	Religion	festival, god, lord	*ahoi ashtami* 2011	0.4 (0.02)
	Art/Literature	yoga, *natyam*, *raaga*	*bharatanaytam* dance *kalakshetra*	0.3 (0.01)
	Astrology	*rashi*, horoscope, *kundali*	*ashwini nakshatra mesha raashi*	0.3 (0.2)
	Attire	saree, *sherwani*, *lehenga*	silk *bandhni chaniya choli*	0.3 (0.04)
Recipe	Recipe/Dish/Food	curry, *biryani*, *paneer*	*matar panir* by *tarala dalal*	1.2 (0.16)
Research	Economic/Agriculture, etc.	*arthik*, *samaj*	*vishwa arthik mandi mein bharat*	0.04 (0.01)
Others	-	meaning	*vibhaa* meaning	0.01 (0.01)

terms across the scripts. However, these methods cannot be directly applied to solve the term matching problem for a single as well as across multiple scripts at the same time.

Therefore, we propose a framework in which the terms across the scripts are modelled jointly. We achieve this by learning a low-dimensional representation of terms in a common abstract space where term equivalents are close to each other. The core of our approach lies in learning such abstract representation. The concept of common abstract space for multi-script modelling is based on the fundamental observation that words are transliterated into a non-native script in such a way that the sound or pronunciation is preserved.

4.2 Formulation

We treat the phonemes as character-level "topics" in the terms. There are some attempts on developing topic models using undirected graphical models like restricted Boltzmann machines (RBMs) [10, 27, 28]. The topic models are usually based on the assumption that each document is represented as a mixture of topics, where each topic defines a probability distribution over words. Similarly, we consider the terms to be represented as mixture of "topics", where each topic defines a probability distribution over character n-grams.

Phonemes of the language can be captured by the character n-grams. Consider the feature set $\mathcal{F} = \{f_1, \ldots, f_K\}$ containing character grams of scripts s_i for all $i \in \{1, ., r\}$ and $|\mathcal{F}| = K$. Let $t_1 = \bigcup_{i=1\ldots r} w_{1,i}$ be a datum from training data T of language l_1 where $w_{1,i}$ represents word w written in language l_1 and script s_i where r is the number of scripts being modelled jointly. The datum can be represented as K-dimensional feature vector $\mathbf{x}$ where x_k is the count of k^{th} feature $f_k \in \mathcal{F}$ in datum t_1. We observe that count data of character grams within terms follow Dirichlet-multinomial distribution. Consider N independent draws from a categorical distribution with K categories . In the present context, $N = \sum_i^K x_i$ and $\{f_1, \ldots, f_K\}$ are K categories, where the number of times a particular feature f_k occurs in the datum t_1 is denoted as x_k. Then $\mathbf{x} = (x_1, \ldots, x_K)$ follows a multinomial distribution with parameters N and $\mathbf{p}$, where $\mathbf{p} = (p_1, \ldots, p_K)$ and p_k is the probability that k^{th} feature takes value x_k. The parameter $\mathbf{p}$ in our case is not directly available hence, we give it a conjugate prior distribution. Therefore, it is drawn from a Dirichlet distribution with parameter vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$. The hyperprior vector $\boldsymbol{\alpha}$ can be seen as pseudocounts (*i.e.*, counts of each feature observed in reference collection) and $\alpha_k = \frac{x_k}{\sum_{i=1}^K x_i}$ in reference collection. Such formulation can be expressed as follows:

$$\begin{array}{lcll} \boldsymbol{\alpha} & = & (\alpha_1, \ldots, \alpha_k) & = \quad hyperprior \\ \mathbf{p}|\boldsymbol{\alpha} & = & (p_1, \ldots, p_K) & \sim \quad Dir(K, \boldsymbol{\alpha}) \\ \mathbf{x}|\mathbf{p} & = & (x_1, \ldots, x_K) & \sim \quad Mult(K, \mathbf{p}) \end{array}$$

The proposed model is based on the non-linear dimensionality reduction methods like deep autoencoder [15]. The RBMs are stacked on top of each other to constitute a deep architecture. The bottom-most RBM of our model, which models the input terms, is character-level variant of the replicated softmax (RSM) model presented in [28] for documents. Despite character n-grams follow Dirichlet-multinomial distribution, we can model them under RSM because during the inference using methods like Gibbs sampling, Dirichlet prior distributions are often marginalised out. Let $\mathbf{v} \in \{0, 1, \cdots, N\}^K$ represent visible multinomial units and let $\mathbf{h} \in \{0, 1\}^m$ be stochastic binary hidden latent units. Let $\mathbf{v}$ be K-dimensional input vector such as feature vector $\mathbf{x}$ for datum t_1, $\mathbf{h}$ be m-dimensional latent feature vector and $N = \sum_{i=1}^K x_i$. The energy of the state $\{\mathbf{v}, \mathbf{h}\}$ is defined as:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{K} v^i a^i - N \sum_{j=1}^{m} b_j h_j - \sum_{i,j} W_j^i h_j v^i \quad (1)$$

where, v^i is the count data x_i, W_j^i is the weight matrix entry between i^{th} visible node and j^{th} hidden node, while a^i and b_j are bias terms of visible and hidden layers respectively. The conditional distributions are given by softmax and logistic functions as below,

$$p(v^i = x_i|\mathbf{h}) = \frac{\exp(a^i + \sum_j h_j W_j^i)}{\sum_{i=1}^K \exp(a_i + \sum_j h_j W_j^i)} \quad (2)$$

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + N \sum_{i=1}^{K} v_i W_j^i) \quad (3)$$

As argued in [27], a single layer of binary features may not be the best way to capture complex structure in the

count data, more layers are added to create a deep autoencoder [15]. The further binary RBM's are stacked on top of each other in such a way that output of the bottom RBM is the input to the above RBM. The conditional distributions of these binary RBMs are given by logistic functions as below,

$$p(v^i = 1|\mathbf{h}) = \sigma(a^i + \sum_j h_j W^i_j) \quad (4)$$

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i v^i W^i_j) \quad (5)$$

4.3 Closed Feature Set - Finite K

The topic models for documents are usually trained over a subset of vocabulary (*top-n* terms) and hence, they have to deal with the non-trivial problem of marginalising over unobserved terms. On the contrary, the term level topic model, proposed in this work, is immune to this problem because the size of phonemes (captured by the character n-grams) for a language is finite and fairly small (only for languages with finite set of alphabets *e.g.* English with alphabets `a-z`). Hence, enough evidence for all the phonemes is found even in a small to moderate size training data, which increases the suitability of our approach to the problem.

For example, without loss of generality consider the total number of scripts in datum being modelled $r = 2$ for language Hindi where s_1 be the Devanagari script with 50 letters and s_2 be the Roman script (as used in English orthography) with 26 letters. Then, the size of the feature set $\mathcal{F}$, considering character uni/bi-gram features, is upper bounded by $K = 3252$ $(26 + 26^2 + 50 + 50^2)$.

4.4 Training

The architecture of the autoencoder is shown Fig. 1 (a). The visible layer of the bottom-most RBM is character level replicated softmax layer as described in Section 4.2. The character uni and bi grams of the training datum ($r = 2$) constitute the feature space $\mathcal{F}$. The hidden layer of the top-most RBM is linear which represents the terms as low-dimensional embedding in the abstract space. The autoencoder is trained in two phases: *i)* greedy layer-wise pre-training and; *ii)* fine-tuning through backpropagation. During pre-training, each RBM is trained using contrastive divergence (CD_1) learning for 50 epoch where CD_1 refers to CD with 1 step of alternating Gibbs sampling [16]. Once the network is pre-trained, the autoencoder is unrolled as shown in Fig. 1 (b) and the cross-entropy error between the input and its reconstruction (output) is backpropagated to adjust the weights of the entire network.

As shown in Fig. 1 (a), the autoencoder is trained with native form and its transliterated form together. In this way, the model is able to learn character level "topic" distribution over the features of both scripts jointly.

4.5 Finding Equivalents

Once the model is trained, equivalents discovery involves two steps: *i)* preparing the index of mining lexicon in abstract space (offline) and; *ii)* finding equivalents for the query term (online). The lexicon of the reference collection (ideally multi-script) which is used to find term equivalents is referred as mining lexicon with size n. The former step is a one-time offline process in which the m-dimensional abstract representation for each term in mining lexicon is obtained as shown in Fig. 1 (c) ($\mathbf{x}_{1\times K} \rightarrow \mathbf{h}_{1\times m}$). These representations are stored in index against each term. This index can be seen as an $n \times m$ matrix $\boldsymbol{H}$ where $\mathbf{h} \in \boldsymbol{H}$. While the latter step involves projecting the query term into the abstract space ($\mathbf{x}_q \rightarrow \mathbf{h}_q$) and calculating the similarity with all the terms in the index. It can be seen as a matrix multiplication operation $\boldsymbol{H}\mathbf{h}_q^T$ considering the similarity function to be *cosine*. All the terms with $sim(\mathbf{h}, \mathbf{h}_q) > \theta, \mathbf{h} \in \boldsymbol{H}$ are considered as equivalents of the query word w_q where θ is similarity threshold.

5. EXPERIMENTS AND RESULTS

Now we describe the experimental set up for evaluating the effectiveness of the proposed method for retrieval in *Multi-Script space*.

5.1 Dataset

We used the FIRE 2013 shared task collection on Transliterated Search [26] for experiments and training. The dataset comprises of document collection, queryset ($\mathcal{Q}$) and relevance judgments. The collection ($\mathcal{D}_1$) contains 62,888 documents containing song title and lyrics in Roman, Devanagari and mixed scripts. Statistics of the document collection is given in Table 2 (a). The $\mathcal{Q}$ contains 25 lyrics search queries for Bollywood songs in Roman script with mean query length of 4.5 words. Table 2 (b) lists a few examples of queries from $\mathcal{Q}$.

Table 2: Details of the Dataset.

(a) Corpus Statistics

No. of	
Documents	62,888
Tokens	12,738,191
Vocabulary	135,243

(b) Example of Queries

Sample Queries
`tumse milke aisa laga`
`wah tera kya kehna`
`zindagi ke safar mein`

5.2 Experimental Setup

The experimental setup is a standard adhoc retrieval setting.The document collection is first indexed to create an inverted index and the index lexicon is used as mining lexicon. Being this a lyrics retrieval set up, the sequential information among the terms is crucial for effectiveness evaluation, *e.g.* "*love me baby*" and "*baby love me*" are completely different songs. In order to capture the word-ordering we consider word 2-grams as a unit for indexing and retrieval.

The non-trivial part of multi-script IR is query-enrichment to handle the challenges described in Sec. 2. In order to enrich the query with equivalents, we find the equivalents of the query terms as described in Section 4.5 and the word 2-gram query is formulated as shown in [13].

5.3 Baseline Systems

We consider a variety of systems to be compared with the proposed method. The query formulation is similar for all the systems including the retrieval settings like inverted index, retrieval model and mining lexicon except the method of finding the equivalents.

1. `Naive`: The original query terms are used for the query formulation without any query-enrichment step.
2. `Naive + Trans`: The original query terms and their automatic back-transliteration obtained from a commercial transliteration engine[4] are used for query formulation.

[4]Yahoo! Transliteration: `http://transliteration.yahoo.com/`

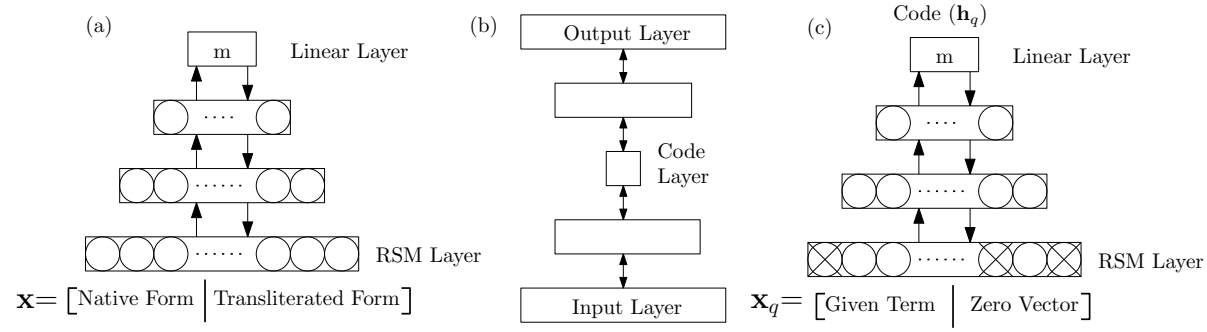

Figure 1: The architecture of the autoencoder (K-500-250-m) during (a) pre-training and (b) fine-tuning. Post training, the abstract level representation of the given terms can be obtained as shown in (c).

3. `LSI`: In this system linear dimensionality reduction technique, latent semantic indexing (LSI) [5] is used to learn the low-dimensional embedding of the terms across the script. Consider matrix $A_{n\times K}$ where a_{ij} is the count data of j^{th} feature $f_j \in \mathcal{F}$ in i^{th} training word-pair. Such matrix A is factored using LSI to learn projection matrix ($V_{K\times m}$) such that $\mathbf{h}_q = \mathbf{x}_q V$. The equivalents of the query term t are obtained from 50-dimensional abstract space as described in Section 4.5. Thus found equivalents along with original query terms are used for query formulation. It can find equivalents across the scripts.

4. `Editex`: An approximate string matching algorithm for IR proposed in [34] is used to get equivalents of the query term. `Editex` uses advanced Phonix and Soundex information to normalise the pronunciation differences. The distance between such normalised strings is calculated as edit distance. `Editex` can handle strings only in Roman alphabet. Therefore, only Roman script equivalents of the query terms are found using Editex.

5. `CCA`: The problem of finding equWe ivalents is formulated as searching across the views by learning hashing functions as presented in [20]. The problem of learning hash functions is formulated as a constrained minimisation problem over the training data objects. The training terms are represented as character bi-gram features and the learning tries to minimize the distance between similar terms in a common geometric space. In the absence of the affinity matrix (i.e., no prior information about similarity between objects is available) the learning of hash functions becomes a generalized eigenvalue formulation of canonical correlation analysis (CCA). An inverted index of hashcodes is prepared for terms in mining lexicon. The equivalents for the query term are found from this index according to the score given by the graph matching algorithm (according to the cosine similarity in the common geometric space) of [30].

5.4 Results and Analysis

We evaluate the effectiveness of the proposed method, referred as `Deep` and compare it with all the baseline systems. The retrieval performance is measured in terms of mean average precision (MAP) and mean reciprocal rank (MRR) evaluation measures. For each query we evaluated the ranklist composed of top 10 documents. The ranking model is parameter free divergence from randomness (unsupervised DFR) as described in [2] which is shown to be suitable for short queries. The results averaged over $\mathcal{Q}$ are presented in Table 3. For `Deep`, the dimensionality selection was based on our previous experience as described in [12]. For `LSI`, we tried different dimensionalities in the range of [50,200] with step size of 50 but did not observe any statistical significant difference in performance. For `CCA`, we used the implementation from the original authors optimised for English and Hindi language pair. The code for `Deep` is made publicly available[5].

Table 3: The results of retrieval performance measured by MAP and MRR.

Method	MRR	MAP	θ
`Naive`	0.6857	0.2910	NA
`Naive+Trans`	0.6590	0.3560	NA
`LSI`	0.7533	0.3522	0.92
`Editex`	0.7767	0.3788	NA
`Editex+Trans`	0.7433	0.4000	NA
`CCA`	0.7640	0.3891	0.997
`Deep-Mono`	**0.8000**	**0.4153**	0.96
`Deep`	**0.8740**	**0.5039**	0.96

The results in Table 3 are presented after the parameter tuning of θ which is better explained later in this section. High MRR score achieved by `Deep` describes its ability to fetch the first relevant document at very high ranks, a desirable quality for Web search in addition to better overall ranking measured by MAP. Although `Editex` is devised for English and able to operate only in the Roman script space, it performs comparable to `CCA` and `LSI`. In order to make a fair comparison, we report two more configurations: `Deep-Mono` which considers only Roman script equivalents and `Editex+Trans` in which automatic transliteration of terms are added to enrich `Editex`. The results clearly outline the superiority of our method for query enrichment. When compared with linear methods such as `PCA` and `CCA` which have linear objective functions, the strong performance of `Deep` suggests that objective function for modelling terms in multi-script space is actually non-linear and not convex. Because of space constraints, we have left for future work the detailed analysis and comparison of features captured by autoencoder at their different layers. A statistical comparison of methods is presented in Table 4. There is no

[5] `http://www.dsic.upv.es/~pgupta/multi-script-ir.html`

Table 4: The performance comparison of systems presented as x/y where x denotes % increase in MAP and y denotes p-value according to paired significance T-Test.

	N+T	LSI	Editex	CCA	Editex+T	Deep
Naive	22.5%/0.09	21%/0.12	30.1%/0.03	33.7%/0.06	37.45%/0.047	73.1%/0.0006
Naive+Trans	-	-0.01%/0.47	6.2%/0.34	9.1%/0.27	12.2%/0.19	41.3%/0.009
LSI	-	-	7.5%/0.24	10.5%/0.22	13.57%/0.12	43.1%/0.0004
Editex	-	-	-	2.7%/0.42	5.6%/0.28	33.0%/0.002
CCA	-	-	-	-	2.8%/0.391	29.5%/0.007
Editex+Trans	-	-	-	-	-	26.0%/0.009

significant difference in performance of Naive+Trans, LSI, Editex and CCA while Deep significantly outperforms all the baselines, as shown with dark-gray b/g, which clearly shows that term equivalents found by Deep are better than the other methods.

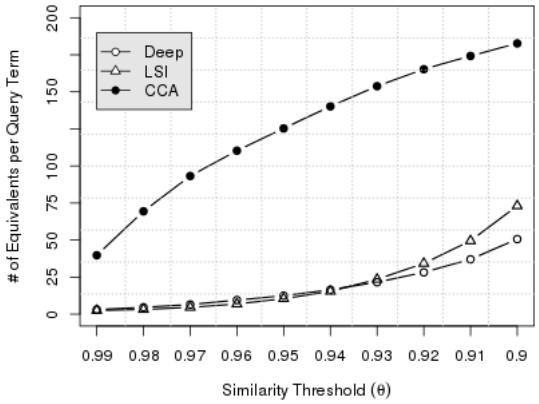

Figure 2: Number of equivalents found in abstract space at similarity threshold (θ) (*c.f.* Section 4.5).

We present an analysis on the impact of θ on number of equivalents, which is directly related to the query latency. Fig. 2 depicts the average number of equivalents for each query term wrt corresponding θ. As can be noticed in Fig. 2, CCA shows a steep increase in number of equivalents which shows, CCA has very dense population in the abstract space and therefore, has around ∼40 equivalents even at a strict threshold of 0.99. Rather Deep and LSI show a moderate increase in the number of equivalents *wrt* θ value.

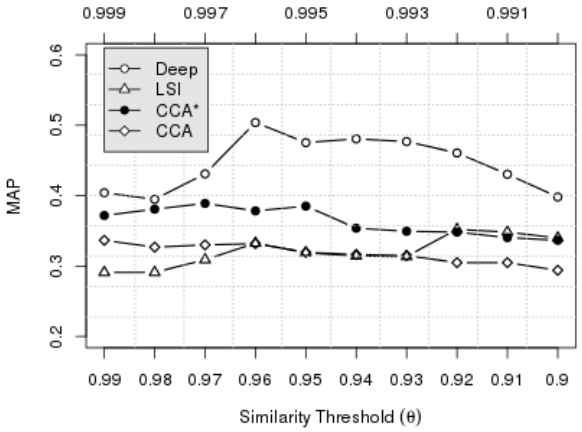

Figure 3: Impact of similarity threshold (θ) on retrieval performance. CCA* follows the ceiling X-axis range [0.999-0.99].

We also show how the θ affects the retrieval performance in Fig. 3. The parameter sweep for θ is [0.99-0.90] with step of 0.01. Deep exhibits the best performance throughout the tuning range. For CCA we also considered θ between [0.999-0.99] with step size of 0.001 to better capture its peak performance as shown in Fig. 3 with CCA*.

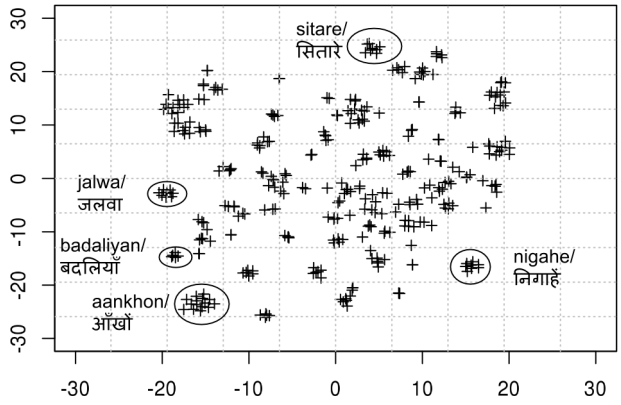

Figure 4: Snippet of mining lexicon projected in abstract space using Deep.

Finally we show the power of Deep for finding equivalents by showing a snippet of 20D abstract space as 2D-view in Fig. 4. It can be noticed that multi-script equivalents of the terms are very close to each other in small clusters and such clusters are well separated from each other. The 2D representation is achieved using the t-SNE algorithm[6]. We show equivalents of a few terms found using Deep with θ=0.96 in Table 5. The category "not sure" depicts the cases where the terms are quite close to the desired term but not correct may be due to a typo *e.g. ehaas* vs. *ehsaas* where the former is not a valid Hindi word.

Table 5: Examples of the variants extracted using Deep with similarity threshold 0.96 (words beginning with *!* and *?* mean "wrong" and "not sure" respectively).

Term	Variants
ehsaas	*ehsas, ehasas, ehsaass, एहसास, ehasaas, ?ehaas, ehsaaas*
mujhe	*muhjhe, !mujhme, ?mujhea, मुझे, !mujheme, mujhee, muhje, muujhe, !मुझमें*
bawra	*bawara, baawra, bavra, !बरवा, bawaraa, baawara, baavra, बावरा, barava, !बिरवा*
pe	*पे, !परे, pee, !ऊपे, ?पए*

5.5 Scalability

Among the two steps involved in finding equivalents listed in Sec. 4.5, the indexing step, being one-time and offline, is not a major concern. But the real time similarity estimation during the online step while searching for equivalents is very crucial for timely retrieval. As the similarity estimation step is essentially a matrix multiplication operation, it

[6] http://homepage.tudelft.nl/19j49/t-SNE.html

can be easily parallelised using multi-core CPUs or GPUs. In our case the size of mining lexicon n=135,243 and abstract space dimensionality m=20. Using a multi-threading framework for matrix multiplication under normal CPU load it takes on an average 0.238 Seconds[7] for the step *(ii)* to find equivalents for each query word. The time-taken is directly proportional to the mining lexicon size n, dimensionality m and the number of CPU/GPU cores.

6. RELATED WORK

Although MSIR has attained very little attention explicitly, many tangentially related problems like CLIR and transliteration for IR do discuss some of the issues of MSIR. While languages like Chinese and Japanese use multiple scripts [24], they may not illustrate the true complexity of the MSIR scenario envisaged here because there are standard rules and preferences for script usage and well defined spellings rules. In Roman transliteration of Hindi, for example, there are no standard rules leading to a large number of variations. Furthermore, these texts are often mixed with English, which makes detection of transliterated text itself a difficult problem.

CLIR typically involve translating queries from one language to another. However, it is often a reasonable choice to transliterate certain OOV words, especially the Named Entities (NEs). While NEs have been worked on extensively in IR and CLIR, transliterated queries where the text, in addition to NE, is represented in the script of another language, typically English, have not received adequate attention. In an analysis of the query logs for Greek web users, Efthimiadis et al. [8] has shown that 90 percent of the queries are formulated using the Roman alphabet while only 8% use the Greek alphabet, and the reason for this [7] is that 1 in 3 Greek navigational queries fail due to the low level of indexing by the search engines of the Greek Web. Want et al. [32] employ a translation based method to classify non-English based queries using an English taxonomy system. Though their method shows some promise, it is heavily dependent on the availability of translation systems for the language pairs in question. Ahmed et al. [1] show that the problem of transliteration is compounded by the fact that due to a lack of standardization in the way a local language is mapped to the Roman script, there is a large variation in spellings. In their work on query-suggestion for a Bollywood Song Search system [6] also stress on the presence of valid variations in spelling Hindi words in Roman script. Related work by [11] goes into the details of handling these variations while mining transliterated pairs from Bollywood song lyric data crawled from the Web. Edit-distance based approaches have also been popular for the generation of such pairs ([29] for English-Telugu, [17] for Tamil-English, for example). [23] propose a method for normalization of transliterated text that combines two techniques: a stemmer based method that deletes commonly used suffixes [22] with rules for mapping variants to a single canonical form. A similar method that uses both stemming and grapheme-to-phoneme conversion is used by [25] to develop a proof-of-concept for a multilingual search engine for 10 Indian languages. Thus, though there has been some interest in the past especially with respect to handling variation and normalization of transliterated text, on the whole the challenge of IR in the *multi-script space* is largely neglected.

For languages like Japanese, Chinese, Arabic and most Indian languages, the challenge of text input in native script means that there is a proliferation of transliterated documents on the web. While the availability of more sophisticated and user-friendly input methods with time has helped resolve this for some of these languages (for example Japanese and Chinese), there is still a large number of languages for which the English keyboard and hence the Roman script remains the main input medium. Further, as a number of relevant documents are available in both the native script and its transliterated form, it also becomes important to deal with not only *Crosslingual* but *Multi-Script* retrieval for such languages. Social media is another domain where the use of transliterated text is widespread. Here text normalization is complicated further by the presence of SMS-like contractions and interjections, and Code-mixing or the switching between languages at phrase, word and morphological levels. As IR becomes more pervasive in social media, dealing with the complexities of transliteration will become more significant for a robust search engine.

7. CONCLUSION AND FUTURE WORK

Although a very important and prevalent problem, *Cross Script* IR (MSIR) has attained very little attention. In this study, the problem of MSIR is, for the first time, introduced formally along with the involved research challenges. We also fill the void of a quantitative analysis of how much Web search traffic is actually affected by MSIR through a large-scale empirical study of Bing query logs, and thereby outline the prevalence and impact of the phenomenon.

A principled solution to address the primary challenge of MSIR, the term variations across the scripts, is proposed. The proposed multi-script joint model learns abstract representation of terms across the scripts though deep-learning architecture such that term equivalents are close to each other. The deep autoencoder based approach provides highly discriminative and powerful representation for terms with as low dimension as m=20. An extensive empirical analysis is presented of experiments with a practical and important use-case, adhoc retrieval of songs lyrics. Our experiments suggest that the state-of-the-art methods for handling spelling variation and transliteration mining have strong effect on success of IR in *Multi-Script space* but the proposed method significantly outperforms them.

We lay the stepping stone to the larger goal of MSIR and in future, one should also deal with the associated research avenues such as code-mixing in queries and documents and more general setup of MSIR such as *Multi-Script Multilingual* IR (MS-MLIR).

8. ACKNOWLEDGMENTS

We thank Rishiraj Saha Roy(IIT Kharagpur), Rohan Ramanath (CMU), Prasenjit Majumder (DA-IICT) and Komal Aggarwal (DA-IICT) for helping in preparation of the dataset. The work of the first and last author is supported by WIQ-EI (No. 269180) and DIANA APPLICATIONS (TIN2012-38603-C02-01) projects. Part of this work was carried during the first author's internship at I^2R, Singapore.

9. ADDITIONAL AUTHORS

[7]We used Intel Xeon CPU E5520 @ 2.27GHz with 4 cores, 8 processors and 12GiB memory.

Additional authors: Spandana Gella (Univesity of Melbourne, email: sgella@student.unimelb.ed) and Jatin Sharma (Microsoft India R&D Pvt Ltd, email: jatinsha@microsoft.com).

10. REFERENCES

[1] U. Z. Ahmed, K. Bali, M. Choudhury, and S. VB. Challenges in designing input method editors for indian lan-guages: The role of word-origin and context. In *Proceedings of the WTIM*, pages 1–9, November 2011.

[2] G. Amati. Frequentist and bayesian approach to information retrieval. In *Proceedings of ECIR*, pages 13–24, 2006.

[3] H.-H. Chen, S.-J. Hueng, Y.-W. Ding, and S.-C. Tsai. Proper name translation in cross-language information retrieval. In *Proceedings of ACL*, pages 232–236, 1998.

[4] M. Choudhury, K. Bali, K. Gupta, and N. Datha. Multilingual search for transliterated content. Patent number US 20120278302, 2012.

[5] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[6] N. Dua, K. Gupta, M. Choudhury, and K. Bali. Query completion without query logs for song search. In *Proceedings of WWW (Companion Volume)*, pages 31–32, 2011.

[7] E. N. Efthimiadis. How do greeks search the web?: A query log analysis study. In *Proceedings of iNEWS*, pages 81–84, 2008.

[8] E. N. Efthimiadis, N. Malevris, A. Kousaridas, A. Lepeniotou, and N. Loutas. Non-english web search: an evaluation of indexing and searching the greek web. *Information Retrieval*, 12(3), 2009.

[9] J. C. French, A. L. Powell, and E. Schulman. Applications of approximate word matching in information retrieval. In *Proceedings of CIKM*, pages 9–15, 1997.

[10] P. V. Gehler, A. D. Holub, and M. Welling. The rate adapting poisson model for information retrieval and object recognition. In *Proceedings of ICML*, pages 337–344, 2006.

[11] K. Gupta, M. Choudhury, and K. Bali. Mining hindi-english transliteration pairs from online hindi lyrics. In *Proceedings of LREC*, pages 2459–2465, 2012.

[12] P. Gupta, R. E. Banchs, and P. Rosso. Squeezing bottlenecks: exploring the limits of autoencoder semantic representation capabilities. *CoRR*, abs/1402.3070, 2014.

[13] P. Gupta, P. Rosso, and R. E. Banchs. Encoding transliteration variation through dimensionality reduction: FIRE Shared Task on Transliterated Search. In *Fifth Forum for Information Retrieval Evaluation*, 2013.

[14] P. A. V. Hall and G. R. Dowling. Approximate string matching. *ACM Comp. Surv.*, 12(4):381–402, 1980.

[15] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[16] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[17] S. C. Janarthanam, S. Subramaniam, and U. Nallasamy. Named entity transliteration for cross-language information retrieval using compressed word format mapping algorithm. In *Proceedings of iNEWS*, pages 33–38, 2008.

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

[19] K. Knight and J. Graehl. Machine transliteration. *Comput. Linguist.*, 24(4):599–612, Dec. 1998.

[20] S. Kumar and R. Udupa. Learning hash functions for cross-view similarity search. In *Proceedings of IJCAI*, pages 1360–1365, 2011.

[21] A. Kumaran, M. M. Khapra, and H. Li. Report of news 2010 transliteration mining shared task. In *Proceedings of NEWS*, pages 21–28, 2010.

[22] D. W. Oard, G.-A. Levow, and C. I. Cabezas. Clef experiments at maryland: Statistical stemming and backoff translation. In *Proceedings of CLEF*, pages 176–187, 2000.

[23] D. Pal, P. Majumder, M. Mitra, S. Mitra, and A. Sen. Issues in searching for indian language web content. In *Proceedings of iNEWS*, pages 93–96, 2008.

[24] Y. Qu, G. Grefenstette, and D. A. Evans. Automatic transliteration for japanese-to-english text retrieval. In *Proceedings of SIGIR*, pages 353–360, 2003.

[25] A. A. Raj and H. Maganti. Transliteration based search engine for multilingual information access. In *Proceedings of CLIAWS3*, pages 12–20, 2009.

[26] R. Saha Roy, M. Choudhury, P. Majumder, and K. Agarwal. Overview and Datasets of FIRE 2013 Track on Transliterated Search. In *Fifth Forum for Information Retrieval Evaluation*, 2013.

[27] R. Salakhutdinov and G. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, July 2009.

[28] R. Salakhutdinov and G. E. Hinton. Replicated softmax: an undirected topic model. In *Proceedings of NIPS*, pages 1607–1614, 2009.

[29] V. B. Sowmya and V. Varma. Transliteration based text input methods for telugu. In *Proceedings of ICCPOL*, pages 122–132, 2009.

[30] R. Udupa and M. M. Khapra. Improving the multilingual user experience of wikipedia using cross-language name search. In *Proceedings of HLT-NAACL*, pages 492–500, 2010.

[31] R. Udupa and M. M. Khapra. Transliteration equivalence using canonical correlation analysis. In *Proceedings of ECIR*, pages 75–86, 2010.

[32] X. Wang, A. Broder, E. Gabrilovich, V. Josifovski, and B. Pang. Cross-lingual query classification: A preliminary study. In *Proceedings of iNEWS*, pages 101–104, 2008.

[33] D. Zhou, M. Truran, T. Brailsford, V. Wade, and H. Ashman. Translation techniques in cross-language information retrieval. *ACM Comput. Surv.*, 45(1):1:1–1:44, Dec. 2012.

[34] J. Zobel and P. Dart. Phonetic string matching: lessons from information retrieval. In *Proceedings of SIGIR*, pages 166–172, 1996.