Encoding transliteration variation through dimensionality reduction

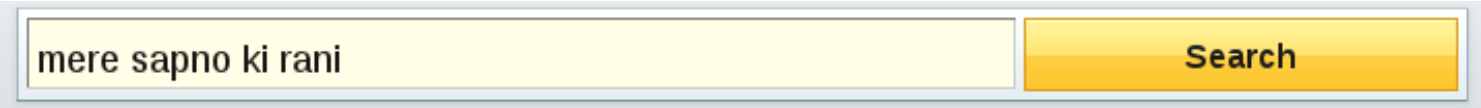
Parth Gupta[1], Paolo Rosso[1] and Rafael E. Banchs[2]

pgupta@dsic.upv.es

[1]Natural Language Engineering Lab
Technical University of Valencia (UPV), Spain
[2] HLT, Institute for Infocomm Research (I[2]R), Singapore


..YOU'RE SUPPOSED TO CATCH MICE...
..NOT ORDER THEM ON e-BAY!
ImpImages.com

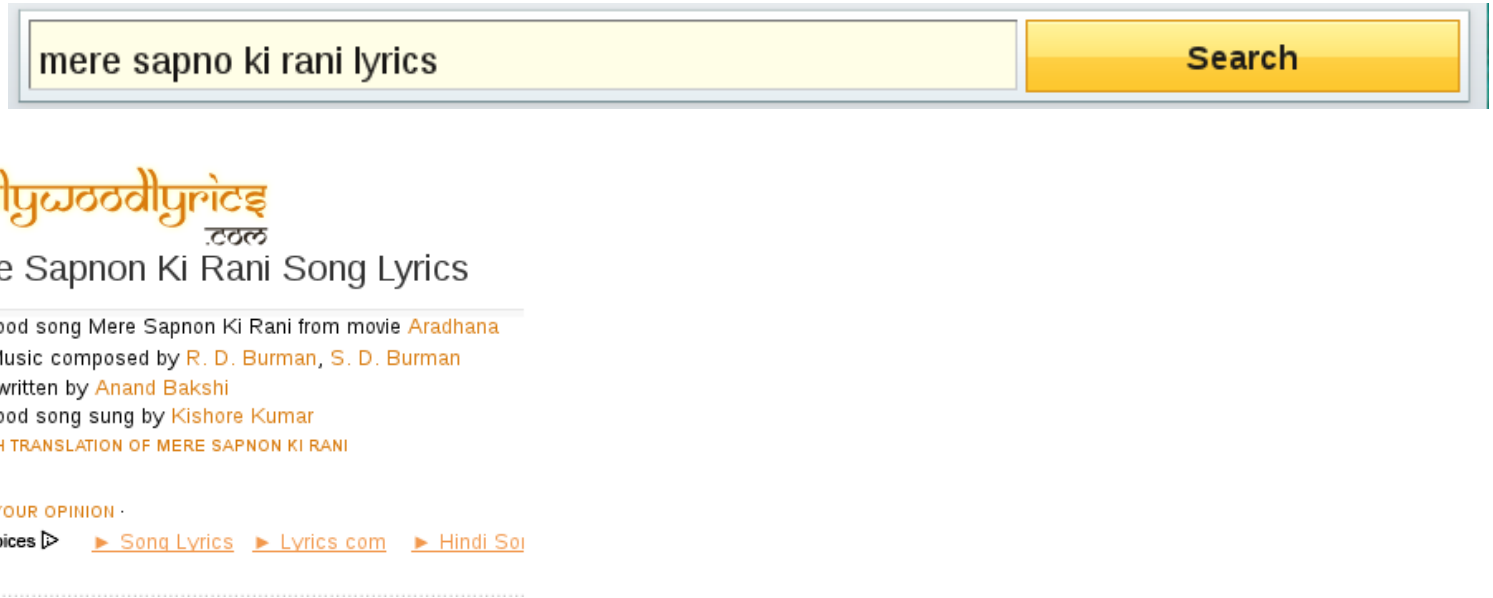
Transliterated Search



(Means: *My Dream Girl*)

Transliterated Search (A special case of Lyrics Retrieval)

Transliterated Search (A special case of Lyrics Retrieval)



Transliterated Search (A special case of Lyrics Retrieval)

Transliterated Search (A special case of Lyrics Retrieval)

bollywoodlyrics.com

Mere Sapnon Ki Rani Song Lyrics

Bollywood song Mere Sapnon Ki Rani from movie Aradhana
Hindi Music composed by R. D. Burman, S. D. Burman
Lyrics written by Anand Bakshi
Bollywood song sung by Kishore Kumar
ENGLISH TRANSLATION OF MERE SAPNON KI RANI

WRITE YOUR OPINION ·
AdChoices ▷ ► Song Lyrics ► Lyrics com ► Hindi So

ST Lyrics

Mere Sapno Ki Rani Lyrics
by Kishore Kumar. From Rough Guide to Bollywood Gold.
Mere Sapno Ki Rani Ringtone

s in Agra
rtrip.com/Agra
2000 Cashback On Your Hotel Booking. Use Code CTHTLSEM. Hurry!

O, eh hey hey, ha ha
Mere sapnon ki rani kab aayegi tu

फिल्म कहानी

आराधना-मेरे सपनो की रानी
Sapno Ki Rani Lyrics)

गुरूवार, 17 जून 2010 19:06 प्रशासक

(10 Votes)
उपयोगकर्ता अंक: / 10
खराब श्रेष्ठ अंक दीजिये

मेरे सपनो की रानी (Mere Sapno Ki Rani)

What is *query* and *document*?

- Query - `Mere Sapno ki rani`
  - The most repeated lines in the song e.g. `Ooh la la ooh la la`
  - The first line of the song e.g. `Tadap tadap ke`
  - The “catchiest” part of the song e.g. `Billo Rani`
  - Quite unique line e.g. `Mujhko saja di pyar ki`
- Document
  - Webpage/document containing that song’s lyrics in [Roman|Devnagari] script

Some challenges

- Extensive spelling variation, e.g. “ayega”, “aaega”, “ayegaa”
- Match across the scripts e.g. आयेगा, “आएगा”
- Unlike normal documents, some words/lines are repeated many times (statistical drift?)

Looking at the Problem...

- Basically the problem is two-fold
  1. Handling spelling variation in the same script
     - Edit Distance?
     - Edit distance is `Integer` *i.e.* many entries at same distance like *Sapney* $\rightarrow$ *Sapne, Apney, Samney* (same distance)
     - Smarter Edit Distance? - *Editex* (uses Phonix and Soundex info) in calculating edit distance
     - Need mature Soundex and Phonix standards for the language.
  2. Performing transliteration generation/mining operation to operate in the other script
     - Basically motivated to operate across the script

Looking at the Problem...

- Basically the problem is two-fold
  1. Handling spelling variation in the same script
     - Edit Distance?
     - Edit distance is `Integer` *i.e.* many entries at same distance like *Sapney* $\rightarrow$ *Sapne, Apney, Samney* (same distance)
     - Smarter Edit Distance? - *Editex* (uses Phonix and Soundex info) in calculating edit distance
     - Need mature Soundex and Phonix standards for the language.
  2. Performing transliteration generation/mining operation to operate in the other script
     - Basically motivated to operate across the script

Looking at the Problem...

- Basically the problem is two-fold
  1. Handling spelling variation in the same script
     - Edit Distance?
     - Edit distance is `Integer` *i.e.* many entries at same distance like *Sapney* $\rightarrow$ *Sapne, Apney, Samney* (same distance)
     - Smarter Edit Distance? - *Editex* (uses Phonix and Soundex info) in calculating edit distance
     - Need mature Soundex and Phonix standards for the language.
  2. Performing transliteration generation/mining operation to operate in the other script
     - Basically motivated to operate across the script

Looking at the Problem...

- Basically the problem is two-fold
  1. Handling spelling variation in the same script
     - Edit Distance?
     - Edit distance is `Integer` *i.e.* many entries at same distance like *Sapney* $\rightarrow$ *Sapne, Apney, Samney* (same distance)
     - Smarter Edit Distance? - *Editex* (uses Phonix and Soundex info) in calculating edit distance
     - Need mature Soundex and Phonix standards for the language.
  2. Performing transliteration generation/mining operation to operate in the other script
     - Basically motivated to operate across the script

Looking at the Problem...

- Basically the problem is two-fold
  1. Handling spelling variation in the same script
     - Edit Distance?
     - Edit distance is `Integer` *i.e.* many entries at same distance like *Sapney* $\rightarrow$ *Sapne, Apney, Samney* (same distance)
     - Smarter Edit Distance? - *Editex* (uses Phonix and Soundex info) in calculating edit distance
     - Need mature Soundex and Phonix standards for the language.
  2. Performing transliteration generation/mining operation to operate in the other script
     - Basically motivated to operate across the script

Looking at the Problem...

- Basically the problem is two-fold
  1. Handling spelling variation in the same script
     - Edit Distance?
     - Edit distance is `Integer` *i.e.* many entries at same distance like *Sapney* $\rightarrow$ *Sapne, Apney, Samney* (same distance)
     - Smarter Edit Distance? - *Editex* (uses Phonix and Soundex info) in calculating edit distance
     - Need mature Soundex and Phonix standards for the language.
  2. Performing transliteration generation/mining operation to operate in the other script
     - Basically motivated to operate across the script

Our Model

- We observe the association among the inter/intra script terms at character uni/bi-gram level.

  1. Intra script, *e.g.* s→sh, f→ph, j→z, मु (mu) → मू (moo)
  2. Inter script *e.g.* k→क, kh→ ख

- Ideally the algorithm should automatically derive such mappings BUT the end goal is to find equivalents considering this information.
- We model inter/intra script equivalents *jointly*.

Our Model

- We observe the association among the inter/intra script terms at character uni/bi-gram level.

  1. Intra script, *e.g.* s→sh, f→ph, j→z, मु (mu) → मू (moo)
  2. Inter script *e.g.* k→क, kh→ ख

- Ideally the algorithm should automatically derive such mappings BUT the end goal is to find equivalents considering this information.
- We model inter/intra script equivalents *jointly*.

Our Model

- We observe the association among the inter/intra script terms at character uni/bi-gram level.

  1. Intra script, *e.g.* s→sh, f→ph, j→z, मु (mu) → मू (moo)
  2. Inter script *e.g.* k→क, kh→ ख

- Ideally the algorithm should automatically derive such mappings BUT the end goal is to find equivalents considering this information.
- We model inter/intra script equivalents *jointly*.

Our Model

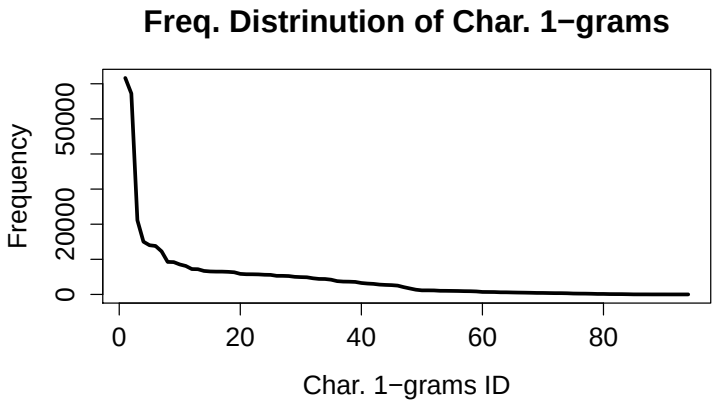
- We observe the association among the inter/intra script terms at character uni/bi-gram level.

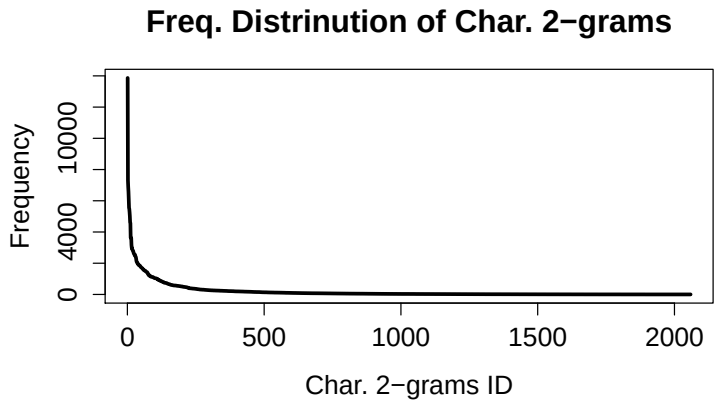
  1. Intra script, *e.g.* s→sh, f→ph, j→z, मु (mu) → मू (moo)
  2. Inter script *e.g.* k→क, kh→ ख

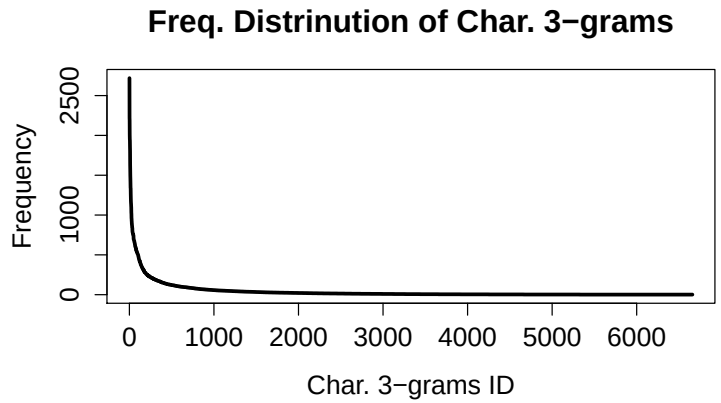
- Ideally the algorithm should automatically derive such mappings BUT the end goal is to find equivalents considering this information.
- We model inter/intra script equivalents *jointly*.

Distribution of Units - Character n-grams in Terms

- The character n-grams in terms follow same distribution as terms in documents with some variation.







Modeling the terms

1. We create unique character uni/bi-gram joint space ($\mathbb{C}^n$) of both scripts out of the training terms, n=dimensionality.
   e.g. `[a b c ...  क ख ..  ch ks ..  ये आ..]`

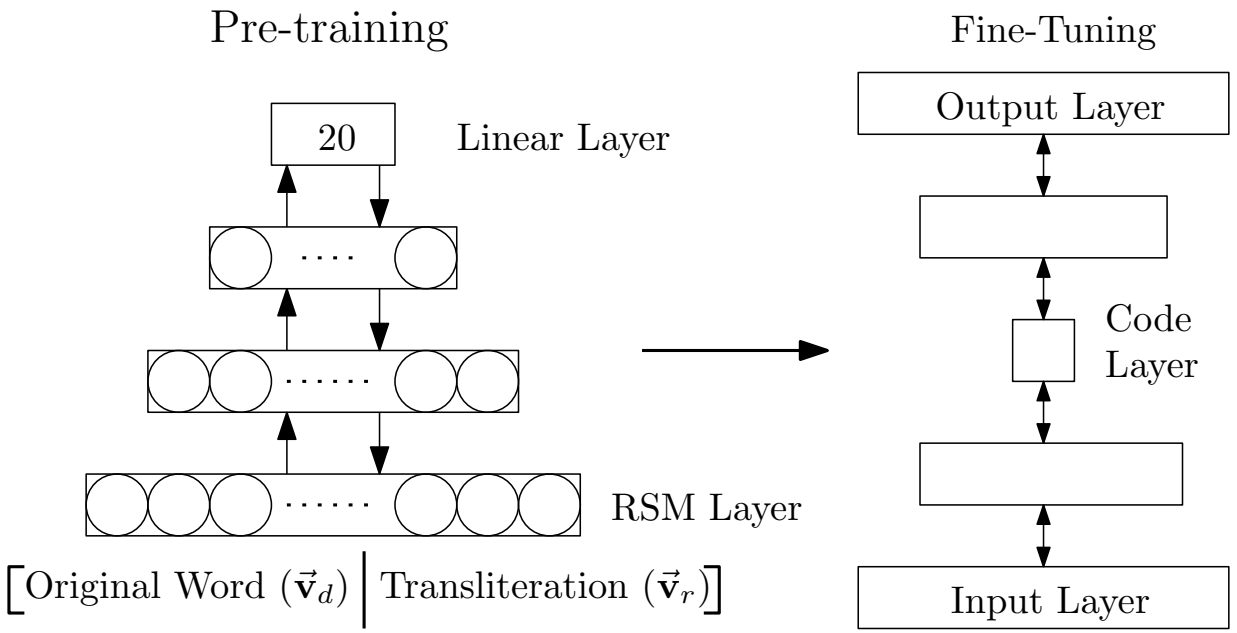
2. The training term-pairs are transformed into feature vector ($\vec{\mathbf{v}}_r, \vec{\mathbf{v}}_d \in \mathbb{C}^n$). *e.g.* $\vec{\mathbf{v}}_r$ = "pyar" and $\vec{\mathbf{v}}_d$ = प्यार.

3. The dimensionality of these pairs are reduced to $\vec{\mathbf{h}}_r, \vec{\mathbf{h}}_d \in \mathbb{R}^m$ such that, $dist(\vec{\mathbf{h}}_r, \vec{\mathbf{h}}_d)$ is minimum where $m << n$.

4. [Important] As there is no distinction between features across the scripts the model can learn *principle components* within (intra) and across (inter) the scripts *jointly*.

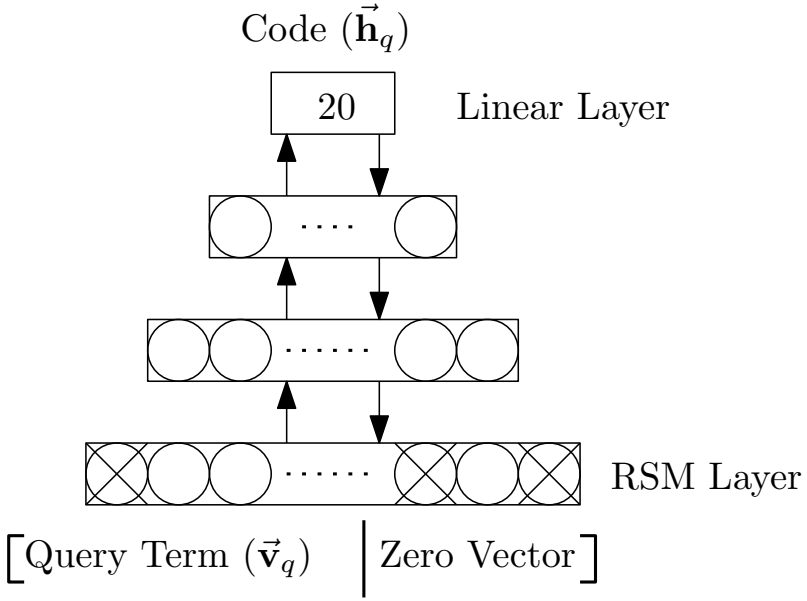
Training Method

- A Deep Autoencoder is trained where the visible layer models the character grams through multinomial sampling [Salakhutdinov and Hinton, 2009].



Finding equivalents

- Apriori the complete lexicon of the reference/source collection is projected into the abstract space using the autoencoder.
- Given the query term q_t, its feature vector $\vec{\mathbf{v}}_{q_t}$ is also projected in the abstract space as ($\vec{\mathbf{h}}_{q_t}$).



- All the terms which have $cosine$ similarity greater than θ are considered as equivalents.

Subtask-2 : *Adhoc* Retrieval

- Query Formulation

Original Query	ik din ayega
Variants of “ik”	“ik”, “ikk”, “ig”, “एक”, “इक”
Variants of “din”	“din”, “didn”, “diin”, “दिन”, “डिन”
Variants of “ayega”	“ayega”, “aeyega”, “ayegaa”, “आयेगा”, “आएगा”
Formulated Query	ikdin, ikdidn, ik$diin, ··· diin$ayega, diin$aeyega, diin$ayegaa, एक$दिन, एक$डिन, ··· , डिन$आयेगा, डिन$आएगा

- Ranking Model (word 2-grams variant)
  - TF-IDF
  - unsupervised DFR (free from parameters)

Demo

Transliteration Encoding Demo

Adhoc Retrieval: Evaluation

	min_len	θ	$algo$
Run-1	2	0.95	TF-IDF
Run-2	2	0.95	DFR
Run-3	3	0.95	DFR

Parameters of our Method

1. min_len - minimum term length for query expansion,
2. θ - similarity threshold and,
3. $algo$ - ranking algorithm.

Metric	Run-1	Run-2	Run-3	Score$_{max}$	Score$_{median}$
nDCG@5	0.7669	0.8052	0.7584	0.8052	0.5620
nDCG@10	0.7642	0.8002	0.7534	0.8002	0.5608
MAP	0.4209	0.4236	0.3558	0.4236	0.2355
MRR	0.7747	0.8440	0.7773	0.8440	0.5884

- **Trade-off**: θ Vs. *Word* n*-gram*

Subtask 1: Query Word Labeling

- Identifying language of Query term
  - SVM classifier on uni/bi/tri-character grams
  - training data 10k English terms and 30k Hindi transliterated terms
- Transliteration
  - transliteration mining approach - Most similar Hindi (Devnagari Script) term in projection space

Labeling		Transliteration	
Labeling Accuracy	0.9540	Transliteration F-Score (run-1)	0.4209
Labeling F-Score (English)	0.9019	Transliteration F-Score (run-2)	0.4311
Labeling F-Score (Hindi)	0.9700	Transliteration F-Score (run-3)	0.3796

Subtask 1 Labeling - Comments

- Simple classification scheme is able to fetch descent labeling accuracy 95%.
- Some terms are present in both the language - *e.g. to (तो), me (मे), chain (चैन), fool (फूल)* and so on.
- Such terms need to be handled properly.

Subtask 1 Transliteration- Comments

- We used Wikipedia as reference collection for transliteration mining.
- Hindi Wikipedia is quite noisy and hence our algorithm gets penalized in some cases like,
  - विवाहः, कया, इबदत, विधिं, फ़रीयाद, दुइनया are mined instead of विवाह, क्या, इबादत, विधि, फरियाद, दुनिया
- Utilizing a more extensive and linguistically correct collection can improve performance
- The transliteration accuracy is 0.43 using Wikipedia lexicon of 384k entries despite coverage and misspelling issues.

Comments in relation of Subtask 1 with Subtask 2

- If the final goal is to retrieve relevant documents then why restrict the query to correct transliteration
  - There also exist also phonetic variation of the terms in the indigenous script: popular but not necessarily correct, *e.g.* मोहब्बत (*Mohabbat*) is also frequently used as मुहब्बत (*Muhabbat*) and महोब्बत (*Mahobbat*).

Thank You for your attention!

References (1)

Salakhutdinov, R. and Hinton, G. E. (2009).
Replicated softmax: an undirected topic model.
In *NIPS*, pages 1607–1614.

Extras

- Ram Leela Vs. Haram Leela
- Sohan Papdi Vs. Mohan Papdi