Improving the Multilingual User Experience of Wikipedia Using Cross-Language Name Search

Raghavendra Udupa
Microsoft Research India
Bangalore, India.

Mitesh Khapra [*]
Indian Institute of Technology Bombay
Powai, India.

Abstract

Although Wikipedia has emerged as a powerful collaborative Encyclopedia on the Web, it is only partially multilingual as most of the content is in English and a small number of other languages. In real-life scenarios, non-English users in general and ESL/EFL [1] users in particular, have a need to search for relevant English Wikipedia articles as no relevant articles are available in their language. The multilingual experience of such users can be significantly improved if they could express their information need in their native language while searching for English Wikipedia articles. In this paper, we propose a novel cross-language name search algorithm and employ it for searching English Wikipedia articles in a diverse set of languages including Hebrew, Hindi, Russian, Kannada, Bangla and Tamil. Our empirical study shows that the multilingual experience of users is significantly improved by our approach.

1 Introduction

Since its inception in 2001, Wikipedia has emerged as the most famous free, web-based, collaborative, and multilingual encyclopedia with over 13 million articles in over 270 languages. However, Wikipedia exhibits severe asymmetry in the distribution of its content in the languages of the world with only a small number of languages dominating (see Table 1). As a consequence, most users of the under-represented languages of the world have no choice but to consult foreign language Wikipedia articles for satisfying their information needs.

[*]This work was done when the author was a summer intern at Microsoft Research India.

[1]*English as Second Language* and *English as Foreign Language*.

Table 1: Linguistic asymmetry of Wikipedia

Language	Speakers	Contributors	Articles
English	1500M	47.1%	3,072,373
Russian	278M	5.2%	441,860
Hebrew	10M	0.7%	97,987
Hindi	550M	0.06%	50,926
Bangla	230M	0.02%	20,342
Tamil	66M	0.04%	19,472
Kannada	47M	0.02%	7,185

Although consulting foreign language Wikipedia is not a solution for the problem of linguistic asymmetry, in the specific case of ESL/EFL users who form a sizable fraction of Internet users of the world [2], it is arguably the most practical option today. Typically, ESL/EFL users are reasonably good at reading and extracting relevant information from English content but not so good at expressing their information needs in English. In particular, getting the spellings of foreign names in English correctly is very difficult for most ESL/EFL users due to the differences in the way a foreign name is pronounced in the native languages. For instance, Japanese EFL speakers often break consonant clusters in foreign names using vowels (see Table 2) and Hindi ESL speakers find it difficult to differentiate between 'an', 'en', and 'on' in English names (such as 'Clin-

[2]As per some estimates, there are about 1 Billion ESL and EFL speakers in the world today and their number is growing.

ton') and will most likely use 'an' ('Clintan').

Table 2: Influence of native language on the English spelling of names.

Wikipedia Entity	Hindi	Japanese	Kannada
Stephen Hawking	Stefan Hoking	Suchifun Houkingu	Steephan Haakimg
Paul Krugman	Pol Crugmun	Pooru Kuruguman	Paal Kragaman
Haroun al-Rashid	Haroon al-Rashid	Haruun aru-Rasheedo	Haroon al-Rasheed
Subrahmaniya Bharati	Subramaniya Bharati	Suburaamaniya Bahaarachi	Subrahmanya Bharathi

In principle, English spell-checkers (Ahmad and Kondrak, 2005) can handle the problem of incorrect spellings in the queries formed by ESL/EFL users. But in practice, there are two difficulties. Firstly, most English spell-checkers do not have a good coverage of names which form the bulk of user queries. Secondly, spelling correction of names is difficult because spelling mistakes are markedly influenced by the native language of the user. Not surprisingly, Wikipedia's inbuilt spell-checker suggests *"Suchin Housing"* as the only alternative to the query *"Suchifun Houkingu"* instead of the correct entity *"Stephen Hawking"* (See Table 3 for more examples).

The inability of ESL/EFL speakers to express their information needs correctly in English and the poor performance of spell-checkers highlight the need for a practical solution for the linguistic asymmetry problem of Wikipedia. In this work, we argue the multilingual user experience of ESL/EFL users can be significantly improved by allowing them to express their information need in their native language. While it might seem that we would need a fully functional cross-language retrieval system that supports translation of non-English queries to English, we note that a good number of the pages in Wikipedia are on people. This empirical fact allows us to improve the multilingual experience of ESL/EFL Wikipedia users by means of cross-language name search which is less resource demanding than a fully functional cross-language retrieval system.

There are several challenges that need to be addressed in order to enable cross-language name search in Wikipedia.

Table 3: Spelling suggestions by Wikipedia.

User Input	Wikipedia's Suggestion	Correct Spelling
Suchifun Houkingu	Suchin Housing	Stephen Hawking
Stefan Hoking	Stefan Ho king	Stephen Hawking
Pol Crugman	Poll Krugman	Paul Krugman
Paal Kragaman	Paul Krugman	Paul Krugman
Suburaamaniya Bahaarachi	Subramaniya Baracchi	Subrahmaniya Bharati

- Firstly, name queries are expressed by ESL/EFL users in the native languages using the orthography of those languages. Transliterating the name into Latin script using a Machine Transliteration system is an option but state-of-the-art Machine Transliteration technologies are still far away from producing the correct transliteration. Further, as pointed out by (Udupa et al., 2009a), it is not enough if a Machine Transliteration system generates a correct transliteration; it must produce the transliteration that is present in the Wikipedia title.

- Secondly, there are about 6 million titles (including redirects) in English Wikipedia which rules out the naive approach of comparing the query with every one of the English Wikipedia titles for transliteration equivalence as is done typically in transliteration mining tasks. A practical cross-language name search system for Wikipedia must be able to search millions of Wikipedia titles in a fraction of a second and return the most relevant titles.

- Thirdly, names are typically multi-word and as a consequence there might not be an exact match between the query and English Wikipedia titles. Any cross-language name search system for Wikipedia must be able to deal with multi-word names and partial matches effectively.

- Fourthly, the cross-language name search system must be tolerant to spelling variations in the query as well as the Wikipedia titles.

In this work, we propose a novel approach to cross-language name search in Wikipedia that addresses all the challenges described above. Further, our approach does not depend on either spell-checkers or Machine Transliteration. Rather we transform the problem into a geometric search problem and employ a state-of-the-art geometric algorithm for searching a very large database of names. This enables us to accurately search the relevant Wikipedia titles for a given user query in a fraction of a second even on a single processor.

1.1 Our Contributions

Our contributions can be summarized as follows:

1. We introduce a language and orthography independent geometric representation for single-word names (Section 3.1).

2. We model the problem of learning the geometric representation of names as a multi-view learning problem and employ the machinery of Canonical Correlation Analysis (CCA) to compute a low-dimensional Euclidean feature space. We map both foreign single-word names and English single-word names to points in the common feature space and the similarity between two single-word names is an exponentially decaying function of the squared geometric distance between the corresponding points (Section 3).

3. We model the problem of searching a database of names as a geometric nearest neighbor problem in low-dimensional Euclidean space and employ the well-known ANN algorithm for approximate nearest neighbors to search for the equivalent of a query name in the English Wikipedia titles (Arya et al., 1998) (Section 3.3).

4. We introduce a simple and efficient algorithm for computing the similarity scores of multi-word names from the single-word similarity scores (Section 3.4).

5. We show experimentally that our approach significantly improves the multilingual experience of ESL/EFL users (Section 4).

2 Related Work

Although approximate similarity search is well-studied, we are not aware of any non-trivial cross-language name search algorithm in the literature. However, several techniques for mining name transliterations from monolingual and comparable corpora have been studied (Pasternack and Roth, 2009), (Goldwasser and Roth, 2008), (Klementiev and Roth, 2006), (Sproat et al., 2006), (Udupa et al., 2009b). These techniques employ various transliteration similarity models. Character unigrams and bigrams were used as features to learn a discriminative transliteration model and time series similarity was combined with the transliteration similarity model (Klementiev and Roth, 2006). A generative transliteration model was proposed and used along with cross-language information retrieval to mine named entity transliterations from large comparable corpora (Udupa et al., 2009b). However, none of these transliteration similarity models are applicable for searching very large name databases as they rely on brute-force search. Not surprisingly, (Pasternack and Roth, 2009) report that *“.. testing [727 single word English names] with fifty thousand [Russian] candidates is a large computational hurdle (it takes our model about seven hours)”*.

Several algorithms for string similarity search have been proposed and applied to various problems (Jin et al., 2005). None of them are directly applicable to cross-language name search as they are based on the assumption that the query string shares the same alphabet as the database strings.

Machine Transliteration has been studied extensively in the context of Machine Translation and Cross-Language Information Retrieval (Knight and Graehl, 1998), (Virga and Khudanpur, 2003), (Kuo et al., 2006), (Sherif and Kondrak, 2007), (Ravi and Knight, 2009), (Li et al., 2009), (Khapra and Bhattacharyya, 2009). However, Machine Transliteration followed by string similarity search gives less-than-satisfactory solution for the cross-language name search problem as we will see later in Section 4.

CCA was introduced by Hotelling in 1936 and has been applied to various problems including CLIR, Text Clustering, and Image Retrieval (Hardoon et al., 2004). Recently, CCA has gained importance in the Machine Learning community as a technique for multi-view learning. CCA computes a common semantic feature space for two-view data and allows users to query a database using either of the two views. CCA has been used in bilingual lexicon extraction from comparable corpora (Gaussier et al., 2004) and monolingual corpora (Haghighi et al., 2008).

Nearest neighbor search is a fundamental problem where challenge is to preprocess a set of points in some metric space into a geometric data structure so that given a query point, its k-nearest neighbors in the set can be reported as fast as possible. It has applications in many areas including pattern recognition and classification, machine learning, data compression, data mining, document retrieval and statistics. The brute-force search algorithm can find the nearest neighbors in running time proportional to the product of the number of points and the dimension of the metric space. When the dimension of the metric space is small, there exist algorithms which give better running time than brute-force search. However, the search time grows exponentially with the dimension and none of the algorithms do significantly better than brute-force search for high-dimensional data. Fortunately, efficient algorithms exist if instead of exact nearest neighbors, we ask for approximate nearest neighbors (Arya et al., 1998).

3 Cross-Language Name Search as a Geometric Search Problem

The key idea behind our approach is the following: if we can embed names as points (or equivalently as vectors) in a suitable geometric space, then the problem of searching a very large database of names can be casted as a geometric search problem, i.e. one of finding the nearest neighbors of the query point in the database.

As illustrative examples, consider the names *Stephen* and *Steven*. A simple geometric representation for these names is the one induced by their corresponding features: $\{St, te, ep, ph, he, en\}$ and $\{St, te, ev, ve, en\}$ [3]. In this representation, each character bigram constitutes a dimension of the geometric feature space whose coordinate value is the number of times the bigram appears in the name. It is possible to find a low-dimensional representation for the names by using Principal Components Analysis or any other dimensionality reduction technique on the bigram feature vectors. However, the key point to note is that once we have an appropriate geometric representation for names, the similarity between two names can be computed as

$$K_{mono}(name1, name2) = e^{-||\phi_1 - \phi_2||^2/2\epsilon^2} \qquad (1)$$

where ϕ_1 and ϕ_2 are the feature vectors of the two names and ϵ is a constant. Armed with the geometric similarity measure, we can leverage geometric search techniques for finding names similar to the query.

In the case of cross-language name search, we need a feature representation of names that is language/script independent. Once we map names in different languages/scripts to the same feature space, we can essentially treat similarity search as a geometric search problem.

3.1 Language/Script Independent Geometric Representation of Names

To obtain language/script independent geometric representation of names, we start by forming the language/script specific feature vectors as described in Section 3. Given two names, *Stephen* in Latin script and स्टीफन in Devanagari script, we form the corresponding character bigram feature vectors ϕ (using features $\{St, te, ep, ph, en\}$) and ψ (using features {स्ट, टी, ीफ, फन}) respectively. We then map these vectors to a common geometric feature space using two linear transformations A and B:

$$\phi \rightarrow A^T\phi = \phi_s \in R^d \qquad (2)$$

$$\psi \rightarrow B^T\psi = \psi_s \in R^d \qquad (3)$$

The vectors ϕ_s and ψ_s can be viewed as language/script independent representation of the names *Stephen* and स्टीफन.

[3]Here, we have employed character bigrams as features. In principle, we can use any suitable set of features including phonetic features extracted from the strings.

3.1.1 Cross-Language Similarity of Names

In order to search a database of names in English when the query is in a native language, say Hindi, we need to be able to measure the similarity of a name in Devangari script with names in Latin script. The language/script independent representation gives a natural way to measure the similarity of names across languages. By embedding the language/script specific feature vectors ϕ and ψ in a common feature space via the projections A and B, we can compute the similarity of the corresponding names as follows:

$$K_{cross}\left(name1, name2\right) = e^{-||\phi_s - \psi_s||^2/2\epsilon^2} \quad (4)$$

It is easy to see from Equation 4 that the similarity score of two names is small when the projections of the names are negatively correlated.

3.2 Learning Common Feature Space using CCA

Ideally, the transformations A and B should be such that similar names in the two languages are mapped to close-by points in the common geometric feature space. It is possible to learn such transformations from a training set of name transliterations in the two languages using the well-known multi-view learning framework of Canonical Correlation Analysis (Hardoon et al., 2004). By viewing the language/script specific feature vectors as two representations/views of the same semantic object, the entity whose name is written as *Stephen* in English and as स्टीफन in Hindi, we can employ the machinery of CCA to find the transformations A and B.

Given a sample of multivariate data with two views, CCA finds a linear transformation for each view such that the correlation between the projections of the two views is maximized. Consider a sample $Z = \{(x_i, y_i)\}_{i=1}^N$ of multivariate data where $x_i \in R^m$ and $y_i \in R^n$ are two views of the object. Let $X = \{x_i\}_{i=1}^N$ and $Y = \{y_i\}_{i=1}^N$. Assume that X and Y are centered[4], i.e., they have zero mean. Let a and b be two directions. We can project X onto the direction a to get $U = \{u_i\}_{i=1}^N$ where $u_i = a^T x_i$. Similarly, we can project Y onto the direction b to get the projections $V = \{v_i\}_{i=1}^n$ where $v_i = b^T y_i$. The aim of CCA is to find a pair of directions (a, b) such that the projections U and V are maximally correlated. This is achieved by solving the following optimization problem:

$$\begin{aligned} \rho &= max_{(a,b)} \frac{< Xa, Xb >}{||Xa|| ||Xb||} \\ &= max_{(a,b)} \frac{a^T XY^T b}{\sqrt{a^T XX^T a}\sqrt{b^T YY^T b}} \end{aligned}$$

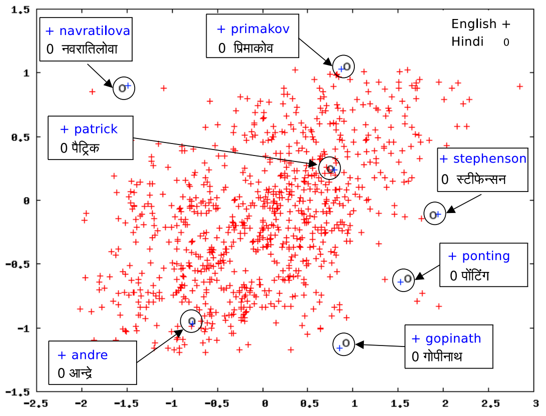
The objective function of Equation 5 can be maximized by solving the following generalized eigen value problem (Hardoon et al., 2004):

$$\begin{aligned} XY^T \left(YY^T\right)^{-1} YX^T a &= \lambda^2 XX^T a \\ \left(YY^T\right)^{-1} YX^T a &= \lambda b \end{aligned}$$

The subsequent basis vectors can be found by adding the orthogonality of bases constraint to the objective function. Although the number of basis vectors can be as high as $\min\{Rank(X), Rank(Y)\}$, in practice, only the first few basis vectors are used since the correlation of the projections is high for these vectors and small for the remaining vectors.

Let A and B be the first $d > 0$ basis vectors computed by CCA.

Figure 1: Projected names (English-Hindi).



3.2.1 Common Geometric Feature Space

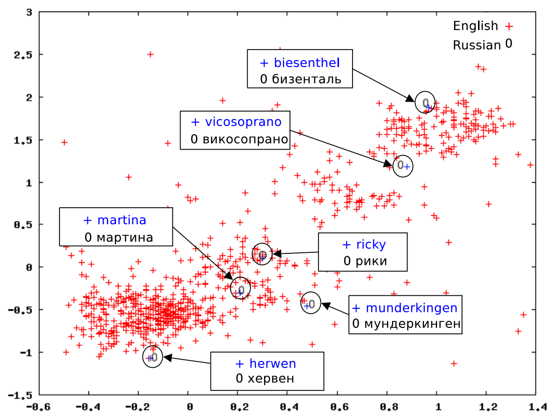
As described in Section 3.1, we represent names as points in the common geometric feature space defined by the projection matrices A and B. Figure 1

[4]If X and Y are not centered, they can be centered by subtracting the respective means.

shows a 2-dimensional common feature space computed by CCA for English (Latin script) and Hindi (Devanagari script) names. As can be seen from the figure, names that are transliterations of each other are mapped to near-by points in the common feature space.

Figure 2 shows a 2-dimensional common feature space for English (Latin script) and Russian (Cyrillic script) names. As can be seen from the figure, names that are transliterations of each other are mapped to near-by points in the common feature space.

Figure 2: Projected names (English-Russian).



3.3 Querying the Name Database

Given a database $D = \{e_i\}_{i=1}^{M}$ of single-word names in English, we first compute their language/script specific feature vectors $\phi^{(i)}$, $i = 1, \ldots, M$. We then compute the projections $\phi_s^{(i)} = A^T\phi^{(i)}$. Thus, we transform the name database D into a set of vectors $\{\phi_s^{(1)}, \ldots, \phi_s^{(M)}\}$ in R^d.

Given a query name h in Hindi, we compute its language/script specific feature vector ψ and project it on to the common feature space to get $\psi_s = B^T\psi \in R^d$. Names similar to h in the database D can be found as solutions of the k-nearest neighbor problem:

$$\begin{aligned} e_{i_k} &= argmax_{e_i \in D - \{e_{i_j}\}_{j=1}^{k-1}} K_{cross}\left(e_i, h\right) \\ &= argmax_{e_i \in D - \{e_{i_j}\}_{j=1}^{k-1}} e^{-||\phi_s^{(i)} - \psi_s||^2/2\epsilon^2} \\ &= argmin_{e_i \in D - \{e_{i_j}\}_{j=1}^{k-1}} ||\phi_s^{(i)} - \psi_s|| \end{aligned}$$

Unfortunately, computing exact k-nearest neighbors in dimensions much higher than 8 is difficult and the best-known methods are only marginally better than brute-force search (Arya et al., 1998). Fortunately, there exist very efficient algorithms for computing approximate nearest neighbors and in practice they do nearly as well as the exact nearest neighbors algorithms (Arya et al., 1998). It is also possible to control the tradeoff between accuracy and running time by specifying a maximum approximation error bound. We employ the well-known Approximate Nearest Neighbors (aka ANN) algorithm by Arya and Mount which is known to do well in practice when $d \leq 100$ (Arya et al., 1998).

3.4 Combining Single-Word Similarities

The approach described in the previous sections works only for single-word names. We need to combine the similarities at the level of individual words into a similarity function for multi-word names. Towards this end, we form a weighted bipartite graph from the two multi-word names as follows:

We first tokenize the Hindi query name into single word tokens and find the nearest English neighbors for each of these Hindi tokens using the method outlined section 3.3. We then find out all the English Words which contain one or more of the English neighbors thus fetched. Let $E = e_1e_2 \ldots e_I$ be one such multi-word English name and $H = h_1h_2 \ldots h_J$ be the multi-word Hindi query. We form a weighted bipartite graph $G = (S \cup T, W)$ with a node s_i for the ith word e_i in E and node t_j for the jth word h_j in H. The weight of the edge (s_i, t_j) is set as $w_{ij} = K_{cross}\left(e_i, h_j\right)$.

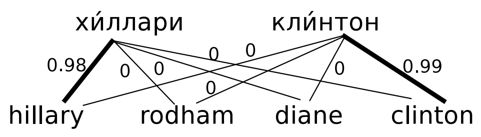
Let w be the weight of the maximum weighted bipartite matching in the graph G. We define the similarity between E and H as follows:

$$K_{cross}\left(E, H\right) = \frac{w}{|I - J| + 1}. \qquad (5)$$

The numerator of the right hand side of Equation 5 favors name pairs which have a good number of high quality matches at the individual word level whereas the denominator penalizes pairs that have disproportionate lengths.

Note that, in practice, both I and J are small and hence we can find the maximum weighted bipartite matching very easily. Further, most edge weights in

Figure 3: Combining Single-Word Similarities.



the bipartite graph are negligibly small. Therefore, even a greedy matching algorithm suffices in practice.

4 Experiments and Results

In the remainder of this section, we refer to our system by GEOM-SEARCH.

4.1 Experimental Setup

We tested our cross language name search system using six native languages, *viz.*, Russian, Hebrew, Hindi, Kannada, Tamil and Bangla. For each of these languages, we created a test set consisting of 1000 multi-word name queries and found manually the most relevant Wikipedia article for each query in the test set. The Wikipedia articles thus found and all the redirect titles that linked to them formed the gold standard for evaluating the performance of our system.

In order to compare the performance of GEOM-SEARCH with a reasonable baseline, we implemented the following baseline: We used a state-of-the art Machine Transliteration system to generate the best transliteration of each of the queries. We used the edit distance between the transliteration and the single-word English name as the similarity score. We combined single word similarities using the approach described in Section 3.4. We refer to this baseline by TRANS-SEARCH.

Note that several English Wikipedia names sometimes get the same score for a query. Therefore, we used a tie-aware mean-reciprocal rank measure to evaluate the performance (McSherry and Najork, 2008).

4.2 GEOM-SEARCH

The training and search procedure employed by GEOM-SEARCH are described below.

4.2.1 CCA Training

We learnt the linear transformations A and B that project the language/script specific feature vectors to the common feature space using the approach discussed in Section 3.2. The learning algorithm requires a training set consisting of pairs of single-word names in English and the respective native language. We used approximately $15,000$ name pairs for each native language.

A key parameter in CCA training is the number of dimensions of the common feature space. We found the optimal number of dimensions using a tuning set consisting of $1,000$ correct name pairs and $1,000$ incorrect name pairs for each native language. We found that $d = 50$ is a very good choice for each native language.

Another key aspect of training is the choice of language/script specific features. For the six languages we experimented with and also for English, we found that character bigrams formed a good set of features. We note that for languages such as Chinese, Japanese, and Korean, unigrams are the best choice. Also, for these languages, it may help to syllabify the English name.

4.2.2 Search

As a pre-processing step, we extracted a list of 1.3 million unique words from the Wikipedia titles. We computed the language/script specific feature vector for each word in this list and projected the vector to the common feature space as described in Section 3.1. The low-dimensional embeddings thus computed formed the input to the ANN algorithm.

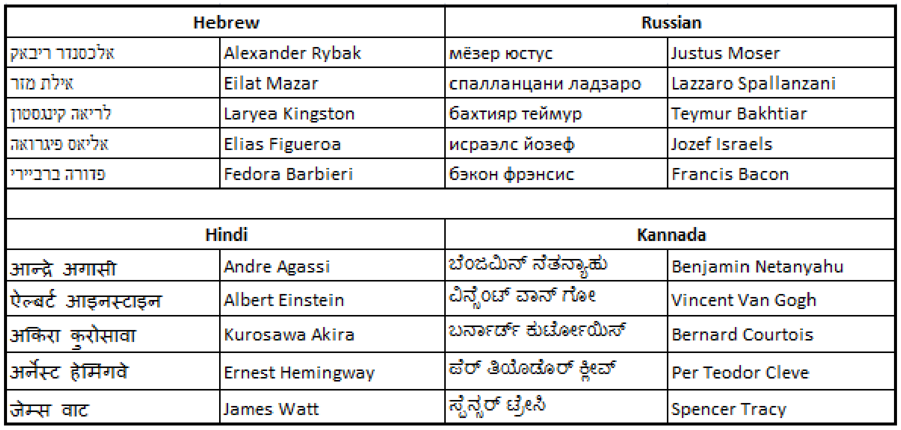
We tokenized each query in the native language into constituent words. For each constituent, we first computed the language/script specific feature vector, projected it to the common feature space, and found the k-nearest neighbors using the ANN algorithm. We used k=100 for all our experiments.

After finding the nearest neighbors and the corresponding similarity scores, we combined the scores using the approach described in Section 3.4.

4.3 TRANS-SEARCH

The training and search procedure employed by TRANS-SEARCH are described below.

Figure 4: Top scoring English Wikipedia page retrieved by GEOM-SEARCH



Hebrew		Russian	
אלכסנדר ריבאק	Alexander Rybak	мёзер юстус	Justus Moser
אילת מזר	Eilat Mazar	спалланцани ладзаро	Lazzaro Spallanzani
לריאה קינגסטון	Laryea Kingston	бахтияр теймур	Teymur Bakhtiar
אליאס פיגרואה	Elias Figueroa	исраэлс йозеф	Jozef Israels
פדורה ברביירי	Fedora Barbieri	бэкон фрэнсис	Francis Bacon
Hindi		**Kannada**	
आन्द्रे अगासी	Andre Agassi	ಬೆಂಜಮಿನ್ ನೆತನ್ಯಾಹು	Benjamin Netanyahu
ऐल्बर्ट आइनस्टाइन	Albert Einstein	ವಿನ್ಸೆಂಟ್ ವಾನ್ ಗೋ	Vincent Van Gogh
अकिरा कुरोसावा	Kurosawa Akira	ಬರ್ನಾರ್ಡ್ ಕುರ್ಟೋಯಿಸ್	Bernard Courtois
अर्नेस्ट हेमिंगवे	Ernest Hemingway	ಪೆರ್ ತಿಯೊಡೊರ್ ಕ್ಲೀವ್	Per Teodor Cleve
जेम्स वाट	James Watt	ಸ್ಪೆನ್ಸರ್ ಟ್ರೇಸಿ	Spencer Tracy

4.3.1 Transliteration Training

We used a state-of-the-art CRF-based transliteration technique for transliterating the native language names (Khapra and Bhattacharyya, 2009). We used CRF++, an open-source CRF training tool, to train the transliteration system. We used exactly the same features and parameter settings as described in (Khapra and Bhattacharyya, 2009). As in the case of CCA, we use around $15,000$ single word name pairs in the training.

4.3.2 Search

The preprocessing step for TRANS-SEARCH is the same as that for GEOM-SEARCH. We transliterated each constituent of the query into English and find all single-word English names that are at an edit distance of at most 3. We computed the similarity score as described in Section 3.4.

4.4 Evaluation

We evaluated the performance of GEOM-SEARCH and TRANS-SEARCH using a tie-aware mean reciprocal rank (MRR). Table 4 compares the average time per query and the MRR of the two systems.

GEOM-SEARCH performed significantly better than the transliteration based baseline system for all the six languages. On an average, the relevant English Wikipedia page was found in the top 2 results produced by GEOM-SEARCH for all the six native languages. Clearly, this shows that GEOM-SEARCH is highly effective as a cross-langauge name search system. The good results also validate our claim that cross-language name search can improve the multi-lingual user experience of ESL/EFL users.

Table 4: MRR and average time per query (in seconds) for the two systems.

Language	GEOM		TRANS	
	Time	MRR	Time	MRR
Hin	0.51	0.686	2.39	0.485
Tam	0.23	0.494	2.16	0.291
Kan	1.08	0.689	2.17	0.522
Ben	1.30	0.495	–	–
Rus	0.15	0.563	1.65	0.476
Heb	0.65	0.723	–	–

5 Conclusions

GEOM-SEARCH, a geometry-based cross-language name search system for Wikipedia, improves the multilingual experience of ESL/EFL users of Wikipedia by allowing them to formulate queries in their native languages. Further, it is easy to integrate a Machine Translation system with GEOM-SEARCH. Such a system would find the relevant English Wikipedia page for a query using GEOM-SEARCH and then translate the relevant Wikipedia pages to the native language using the Machine Translation system.

6 Acknowledgement

We thank Jagadeesh Jagarlamudi and Shaishav Kumar for their help.

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 955–962, Morristown, NJ, USA. Association for Computational Linguistics.

Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923.

Éric Gaussier, Jean-Michel Renders, Irina Matveeva, Cyril Goutte, and Hervé Déjean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *ACL*, pages 526–533.

Dan Goldwasser and Dan Roth. 2008. Transliteration as constrained optimization. In *EMNLP*, pages 353–362.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: HLT*, pages 771–779, Columbus, Ohio, June. Association for Computational Linguistics.

David R. Hardoon, Sándor Szedmák, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664.

Liang Jin, Nick Koudas, Chen Li, and Anthony K. H. Tung. 2005. Indexing mixed types for approximate retrieval. In *VLDB*, pages 793–804.

Mitesh Khapra and Pushpak Bhattacharyya. 2009. Improving transliteration accuracy using word-origin detection and lexicon lookup. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*. Association for Computational Linguistics.

Alexandre Klementiev and Dan Roth. 2006. Named entity transliteration and discovery from multilingual comparable corpora. In *HLT-NAACL*.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Jin-Shea Kuo, Haizhou Li, and Ying-Kuei Yang. 2006. Learning transliteration lexicons from the web. In *ACL*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Haizhou Li, A Kumaran, Vladimir Pervouchine, and Min Zhang. 2009. Report of news 2009 machine transliteration shared task. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*. Association for Computational Linguistics.

Frank McSherry and Marc Najork. 2008. Computing information retrieval performance measures efficiently in the presence of tied scores. In *ECIR*, pages 414–421.

Jeff Pasternack and Dan Roth. 2009. Learning better transliterations. In *CIKM*, pages 177–186.

Sujith Ravi and Kevin Knight. 2009. Learning phoneme mappings for transliteration without parallel data. In *NAACL-HLT*.

Hanan Samet. 2006. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann, August.

Tarek Sherif and Grzegorz Kondrak. 2007. Substring-based transliteration. In *ACL*.

Richard Sproat, Tao Tao, and ChengXiang Zhai. 2006. Named entity transliteration with comparable corpora. In *ACL*.

Raghavendra Udupa, K. Saravanan, Anton Bakalov, and Abhijit Bhole. 2009a. "they are out there, if you know where to look": Mining transliterations of oov query terms for cross-language information retrieval. In *ECIR*, pages 437–448.

Raghavendra Udupa, K. Saravanan, A. Kumaran, and Jagadeesh Jagarlamudi. 2009b. Mint: A method for effective and scalable mining of named entity transliterations from large comparable corpora. In *EACL*, pages 799–807.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-language applications. In *SIGIR*, pages 365–366.