CEN@Amrita: Information Retrieval on CodeMixed HindiEnglish Tweets Using Vector Space Models

Shivkaran Singh
Centre for Computational Engineering and Networking (CEN), Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, Amrita University, India, PIN: 641112
+91 84278 78973
shivkaran.ssokhey@gmail.com

Anand Kumar M
Centre for Computational Engineering and Networking (CEN), Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, Amrita University, India, PIN: 641112
m_anandkumar@cb.amrita.edu

Soman K P
Centre for Computational Engineering and Networking (CEN), Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, Amrita University, India, PIN: 641112
kp_soman@amrita.edu

ABSTRACT
One of the major challenges nowadays is Information retrieval from social media platforms. Most of the information on these platforms is informal and noisy in nature. It makes the Information retrieval task more challenging. The task is even more difficult for twitter because of its character limitation per tweet. This limitation bounds the user to express himself in condensed set of words. In the context of India, scenario is little more complicated as users prefer to type in their mother tongue but lack of input tools force them to use Roman script with English embeddings. This combination of multiple languages written in the Roman script makes the Information retrieval task even harder. Query processing for such CodeMixed content is a difficult task because query can be in either of the language and it need to be matched with the documents written in any of the language. In this work, we dealt with this problem using Vector Space Models which gave significantly better results than the other participants. The Mean Average Precision (MAP) for our system was 0.0315 which was second best performance for the subtask.

CCS Concepts
• Information Systems → Information Retrieval → Retrieval models and ranking

• Computing methodologies → Artificial Intelligence → Natural Language Processing

Keywords
CodeMixed social media, Mixed-Script, Information Retrieval, Vector-space-models, Semantics

1. INTRODUCTION
Social media has a plentitude of user generated data in numerous languages which are predominantly informal in nature. Most of these languages have their own native scripts. Some of these scripts include Arabic, Chinese, Hebrew, Greek, and Indic etc. For most of these languages, major user-generated content is transliterated into the Roman script with English embeddings. The trend in Indian social media is to use such informal text containing a mixture of multiple South-Asian languages with English embeddings. This mixture makes the Information Retrieval (IR) task very challenging. In Forum for Information retrieval (FIRE)[1] (2016), a similar task was proposed, which required Mixed Script IR on Code-Mixed Hindi-English tweets. The difference of Code-Mixed IR from MixedScript IR is subtle. In MixedScript content, query q_{ms} is written in Roman or native script [1] whereas in Code-Mixed content, query q_{cm} is a Roman transliteration of a different language. The Code-Mixed corpus provided at MSIR Subtask II had English and Roman transliterated Hindi twitter data [2]. The major issue in such corpus is several possibilities of writing the same (Hindi) word with different transliterations. For example, "कम" meaning "less" in Hindi can be spelled in Roman transliteration as *km, kam, kum, kmm* etc. These nuances make it hard for the IR system to match the query with correct document in a document set. This significantly affects the performance of IR system. Nowadays, getting information from such CodeMixed social media text is very important as it helps in many business analytics purposes. In the following sections, Section 2 explains about the information retrieval subtask, Section 3 explains the Vector Space Models which were used for information retrieval, Section 4 explains the methodology used for this work, Section 5 discusses about the results obtained and analysis of others result.

2. Task Description
The subtask II of shared task of Mixed Script IR on Code-Mixed Hindi-English tweets was to retrieve 20 most relevant tweets from a document given a query. The query as well as the document was in Roman script but with CodeMixed Hindi and English languages. The corpus had set of documents with each document containing several hundred (or thousand) tweets. The corpus was further classified based on topics and queries. Each topic had at least one query related to the topic description. **Table 1** explains about the structure of training/testing corpus provided for the subtask. The total number of topics for training and testing corpus was 10 and 3 respectively. There were several queries based on each topic (See Table 1) and there was at least one query per topic. The total number of queries for training was 23 and for testing, it was 12. A narrative on each topic was also given in the corpus describing the details about the tweets under that topic. The topic 001 (Aam Aadmi Party) has four queries under the same description (Table 1). All these four queries had separate documents with a corresponding number of tweets. Let q_i be the given query, IR task was to rank the tweets in the corresponding document from most relevant with the query to the least.

[1] https://msir2016.github.io/

Table 1. Corpus Description

Topic	Topic Description	Queries	#tweets
001 Aam Aadmi Party	The tweets under this topic are related to the Aam Aadmi Party which is a political party in Delhi Government	q1: aam aadmi party	710
		q2: aam aadmi party dilli me	1071
		q3: aam aadmi ki party	1583
		q4: aap ki rally	3529

The IR system should return top-20 most relevant tweets to the given query. The CodeMixed nature of the tweets makes the IR task hard to process as semantic search for such transliterated queries and documents is still an unsolved problem [2].

3. Vector Space Models

Vector-Space-Models (VSMs) are used to represent documents as a vector (of terms) that occurs within a collection [5]. The given query is also represented in the same document space. The query is also called as *pseudo-document.* As the document is represented as a vector of terms that occur in the document hence it is necessary to identify the terms present in the document. The terms are basically the vocabulary of collection of documents. If there are more than one document then each document will be a huge vector and it will be convenient to organize these vectors into a matrix. This matrix is called *term-document* matrix. The row vectors are referred as terms and column vectors are referred as documents. A document is used as a context to understand the term. If we take document as phrases, sentences, paragraphs, chapters etc. we get a *word-context* matrix. Similarly we can also have a *pair-pattern* matrices [3].

To imagine the representation of term-document matrix, think of a multiset from set theory. A multiset is a set but it allows multiple instances of the same element. For example, $M = \{x, x, x, y, y, z\}$ is a multiset containing elements x, y and z. Just like sets, order of elements in multiset could be anything. That means, multiset $M_1 = \{y, z, y, x, x, x\}$ is same as multiset $M_2 = \{x, y, z, x, x, y\}$. Multisets are also called as *bags* and we can represent these bags as a vectors with vector component denoting the frequency of the elements of multiset i.e. $V = < 3, 2, 1 >$ is vector representation of the bag M in which 3 is the frequency of x and 2 is the frequency of y etc. Using the same analogy, we can imagine a document as a bag and set of documents as set of bags aligned as columns in a matrix, say X. This matrix, X, is term document matrix with columns representing a bag and rows representing a unique member. A particular element x_{ij} in the matrix corresponds to the frequency of i^{th} term in the j^{th} document (or bag). To capture the whole intuition, let's assume 3 documents as:

Doc1: We stayed very closely connected.

Doc2: Charger stayed connected with phone.

Doc3: His phone charger closely resembled mine.

The term document matrix of frequency for above three documents could be:

Table 2. Term-Document matrix

	Doc1	Doc2	Doc3
We	**1**	**0**	**0**
stayed	**1**	**1**	**0**
very	**1**	**0**	**0**
closely	**1**	**0**	**1**
Connected	**1**	**1**	**0**
Charger	**0**	**1**	**1**
with	**0**	**1**	**0**
phone	**0**	**1**	**1**
His	**0**	**0**	**1**
resembled	**0**	**0**	**1**
mine	**0**	**0**	**1**

In the above matrix, terms are the rows and columns are documents. It has 3 documents (Doc1, Doc2 & Doc3) and 11 unique terms (tokens in this case) with dimension 11x3. In a similar way a given query could be represented as bag of words and estimating the relevance of query with the documents in such a manner is called *bag of words hypothesis* in Information retrieval. This hypothesis states that a column vector in a term-document matrix captures the meaning of the corresponding document (to some extent). It should be observed that the column vector which correspond to a document in a collection tell us about the frequency of the words in the document with loss of actual order of the words. The vector may not capture the structure of a document as it is but it works surprisingly well with the search engines. We can compare the column (document) vectors to compute the similarity among them. This similarity can be computed using *euclidean distance* if we are assuming columns (documents) as points in the document space. If we are assuming columns (documents) as vectors in documents space, we can use *cosine similarity* to measure the similarity by the angle between the vectors. Larger the cosine, more semantically related the documents are. If $doc1$ and $doc2$ are two document vectors, then cosine of angle θ between them is computed as:

$$\cos(doc1, doc2) = \frac{dot(doc1, doc2)}{||doc1||.||doc2||}$$

Where $||doc1||$ and $||doc2||$ are the length (or norm) of the vectors. The basic intuition behind using cosine similarity is that it captures the idea that the angle between the vectors is important, length of the vector is not (See Figure 1). The cosine is 1 when vectors are same or they point in the same direction ($\theta\ zero$). The cosine value varies from 0-1, zero being not similar and one being exactly similar.

3.1 Term-Weighing

Generally, most frequent terms will have lower information than the less frequent or surprising terms. To capture this idea, most efficient way is to use $tf - idf$ (term frequency-inverse document frequency). An element in a term-document matrix gets a higher weight when a term in corresponding document is very frequent (tf) that means the term is rare in collection of documents(df). Hence the weight of a particular terms appearance is computed as:

$$W_{mn} = tf \times idf$$

Where W_{mn} is the weight of the term m in document n. It is demonstrated in [4] that using $tf - idf$ functions brings significant improvement over raw frequency.

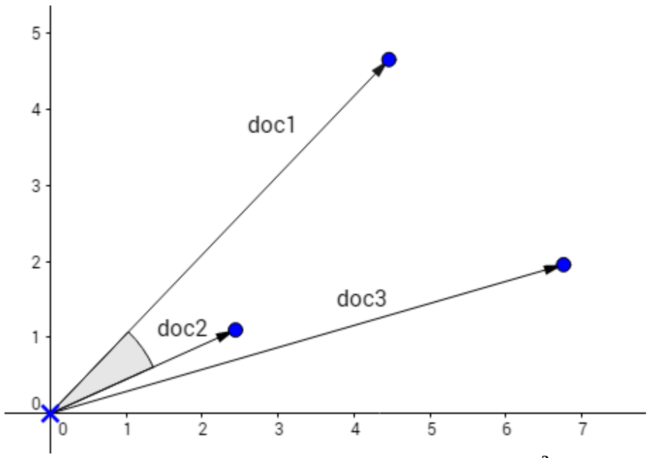


Figure. 1 Angle between document vectors[2]

So far we have talked about measuring document similarity but VSMs can also be used for query processing. A query q can be treated as a pseudo document and similarity measures of each document in the collection with pseudo document (query) can be computed. There are several other similarity measures available as Jensen-Shannon, recall, precision, Jaccard, harmonic mean etc. The use of these similarity measures depends upon the relative frequency of adjacent words with respect to the target word.

4. Methodology

The subtask II in FIRE was Mixed Script Information Retrieval on Code-Mixed Hindi-English tweets. There were total 23 files containing tweets for training and 12 files for testing. Each file had a corresponding query. Given a query $\boldsymbol{q_i}$, the information retrieval task was to compute the similarity between the query and tweets in each file and return the top 20 most relevant tweets. As explained in the last section, this query processing task can be successfully executed using VSMs.

Each file was treated as a collection of documents and each tweet within the collection is referred as document. The dataset comprised of Hindi-English code-mixed tweets. As twitter data is generally noisy and requires some preprocessing, it was subjected to some preprocessing modules. The preprocessing in our implementation included tokenizing, removing stop words, stripping punctuations, stripping repetitions (hiiiiiii→ hi) etc. The major issue in tokenizing twitter data is to capture the key attributes of tweets such as: hashtags (#aap), @ mentions (@timesnow), URLs, symbol, emoticons etc. These attribute were captured using regular expressions. A sample tweet after capturing these nuances, stripping punctuation and tokenizing appeared as:

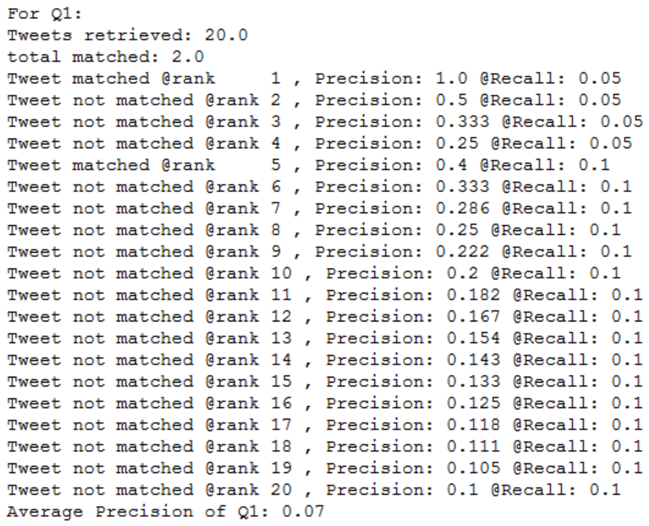
Original	@respectshraddie shhhhh :(salman ko jail hojaegi :(#badday
Preprocessed	'@respectshraddie', 'shh', ':(', 'salman', 'jail', 'hojaegi', ':(' , '#badday'

Tokenization was done for all the queries too. The document vector (column) size, as well as the query vectors, were in same vector space. The preprocessing was performed for each file (collection) over each document (tweet). After preprocessing over each file (collection), it was fed to Information Retrieval system. The similarity scores for each tweet in a collection given a query were computed and results were saved in a list. The top 20 tweets related to the given query were retrieved from the index values of top 20 similarity scores in the list.

There was a provision of submitting three systems per team. We submitted two systems. One system was same as explained above. In second system, we manually removed some Hindi stop words like $ka, jo, ke\ ne, to, ve, le$ etc. It didn't reflected any better results. All the implementations were done in Python 2.7. Related code will be made available at author's Github page.

5. Result and Analysis

The result were declared roughly after two weeks of the submission. There were total 7 teams and our system performed well as compared to others [6]. The evaluation was done by calculating Mean Average Precision (MAP) which is a standard measure for comparing search algorithm. The results for Q1 for system 1 and system 2 can be seen in **Figure 2.** And **Figure 3.**



```
For Q1:
Tweets retrieved: 20.0
total matched: 2.0
Tweet matched @rank     1 , Precision: 1.0 @Recall: 0.05
Tweet not matched @rank 2 , Precision: 0.5 @Recall: 0.05
Tweet not matched @rank 3 , Precision: 0.333 @Recall: 0.05
Tweet not matched @rank 4 , Precision: 0.25 @Recall: 0.05
Tweet matched @rank     5 , Precision: 0.4 @Recall: 0.1
Tweet not matched @rank 6 , Precision: 0.333 @Recall: 0.1
Tweet not matched @rank 7 , Precision: 0.286 @Recall: 0.1
Tweet not matched @rank 8 , Precision: 0.25 @Recall: 0.1
Tweet not matched @rank 9 , Precision: 0.222 @Recall: 0.1
Tweet not matched @rank 10 , Precision: 0.2 @Recall: 0.1
Tweet not matched @rank 11 , Precision: 0.182 @Recall: 0.1
Tweet not matched @rank 12 , Precision: 0.167 @Recall: 0.1
Tweet not matched @rank 13 , Precision: 0.154 @Recall: 0.1
Tweet not matched @rank 14 , Precision: 0.143 @Recall: 0.1
Tweet not matched @rank 15 , Precision: 0.133 @Recall: 0.1
Tweet not matched @rank 16 , Precision: 0.125 @Recall: 0.1
Tweet not matched @rank 17 , Precision: 0.118 @Recall: 0.1
Tweet not matched @rank 18 , Precision: 0.111 @Recall: 0.1
Tweet not matched @rank 19 , Precision: 0.105 @Recall: 0.1
Tweet not matched @rank 20 , Precision: 0.1 @Recall: 0.1
Average Precision of Q1: 0.07
```

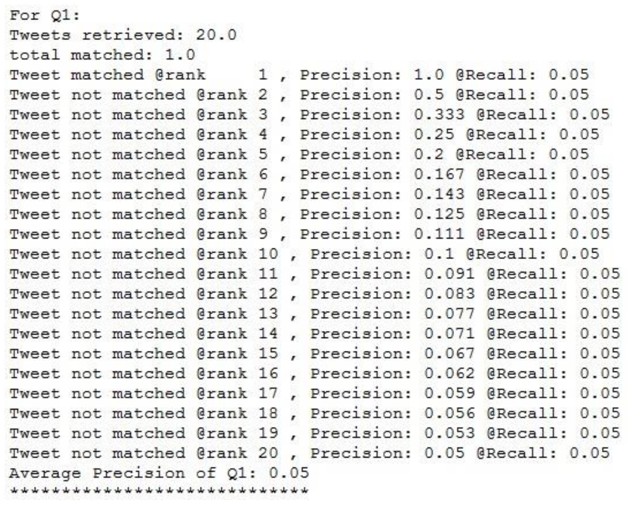
Figure. 2 System-1 results

The results of top three performers in the subtask are given in Table 3.

Table 3

Team Name	No. of runs	Runs (Mean Average Precision)		
		Run 1	Run 2	Run 3
Amrita_CEN	1	0.0377	NIL	NIL
CEN@Amrita*	2	0.0315	0.016	NIL
UB	3	0.0217	0.016	0.015

[2] https://www.math10.com/en/geometry/geogebra/geogebra.html



```
For Q1:
Tweets retrieved: 20.0
total matched: 1.0
Tweet matched @rank     1 , Precision: 1.0 @Recall: 0.05
Tweet not matched @rank 2 , Precision: 0.5 @Recall: 0.05
Tweet not matched @rank 3 , Precision: 0.333 @Recall: 0.05
Tweet not matched @rank 4 , Precision: 0.25 @Recall: 0.05
Tweet not matched @rank 5 , Precision: 0.2 @Recall: 0.05
Tweet not matched @rank 6 , Precision: 0.167 @Recall: 0.05
Tweet not matched @rank 7 , Precision: 0.143 @Recall: 0.05
Tweet not matched @rank 8 , Precision: 0.125 @Recall: 0.05
Tweet not matched @rank 9 , Precision: 0.111 @Recall: 0.05
Tweet not matched @rank 10 , Precision: 0.1 @Recall: 0.05
Tweet not matched @rank 11 , Precision: 0.091 @Recall: 0.05
Tweet not matched @rank 12 , Precision: 0.083 @Recall: 0.05
Tweet not matched @rank 13 , Precision: 0.077 @Recall: 0.05
Tweet not matched @rank 14 , Precision: 0.071 @Recall: 0.05
Tweet not matched @rank 15 , Precision: 0.067 @Recall: 0.05
Tweet not matched @rank 16 , Precision: 0.062 @Recall: 0.05
Tweet not matched @rank 17 , Precision: 0.059 @Recall: 0.05
Tweet not matched @rank 18 , Precision: 0.056 @Recall: 0.05
Tweet not matched @rank 19 , Precision: 0.053 @Recall: 0.05
Tweet not matched @rank 20 , Precision: 0.05 @Recall: 0.05
Average Precision of Q1: 0.05
*****************************
```

Figure. 3 System-2 results

6. Conclusion

The shared task on CodeMixed Information retrieval was indeed a unique task. It captured the latest trend in social media. We used Vector Space Models (VSMs) of semantics to compute the similarity between the tweets and given query. The performance of our system was ranked 2 among all the participants. But the Mean Average Precision (MAP) value was very low in terms of performance. That suggests, CodeMixed IR task is a difficult task and existing algorithms do not perform as expected and require sufficient attention to perform well for such data.

Acknowledgements

The authors would like to thank the organizers of Forum for Information Retrieval Evaluation (FIRE) for organizing this event. The authors would also like to thank the organizers of shared task on Mixed Script Information Retrieval (MSIR) for organizing the much coveted task for Indian social media.

REFERENCES

[1] Gupta, P., Bali, K., Banchs, E. R., Choudhury, M., and Rosso, P. 2014. Query expansion for mixed-script information retrieval. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pp. 677-686. ACM.

[2] Chakma, K., and Das, A. 2016. CMIR: A Corpus for Evaluation of Code Mixed Information Retrieval of Hindi-English Tweets. *Computación y Sistemas* 20.3 (2016): 425-434.

[3] Turney, P. D., and Pantel, P. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37.1 (2010): 141-188.

[4] Salton, G., and Buckley, B. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24.5 (1988): 513-523.

[5] Soman, K. P., Loganathan, R., and Ajay, V. 2009. *Machine learning with SVM and other kernel methods*. PHI Learning Pvt. Ltd.

[6] Banerjee, S., Chakma K., Naskar, S. K., Das, A., Rosso, P., Bandyopadhyay, S., and Choudhury, M. 2016. Overview of the Mixed Script Information Retrieval at FIRE. In *Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation, Kolkata, India, December 7-10, 2016*, CEUR Workshop Proceedings. CEUR-WS.org.