Exploiting Named Entity Mentions Towards Code Mixed IR : Working Notes for the UB system submission for MSIR@FIRE'16

Nikhil Londhe
SUNY Buffalo
nikhillo@buffalo.edu

Rohini K. Srihari
SUNY Buffalo
rohini@buffalo.edu

ABSTRACT

A sizable percentage of online user generated content is susceptible to code switching and code mixing owing to a variety of reasons. Thus, an expected consequence is that adhoc user queries on such data are also inherently code mixed. This paper thus presents our solution for a similar scenario : information retrieval on code mixed Hindi-English tweets. We explore techniques in information extraction, clustering and query expansion as part of this work and present our results on the test dataset. Our system achieved a MAP of 0.0217 on the test set and placed third on the rankings.

CCS Concepts

•Information systems → Multilingual and cross-lingual retrieval; •Computing methodologies → Natural language processing;

1. INTRODUCTION

A large number of languages like Russian, Hindi, Arabic, etc: that although have indigenous writing scripts, are often expressed in the Roman script in online communication due a variety of socio-cultural reasons. Furthermore, it is not uncommon to see multilingual speakers switching between different languages when expressing themselves in an informal setting like social media. Hence, a large percentage of user generated content, typically on social media is *code switched* or *code mixed*, i.e., contains content in more than one language that may or may not involve multiple writing systems.

Given the nature of the content, it is thus not unexpected to see adhoc user queries on such data to be also code mixed. For example, consider Table 1, that lists some sample queries and sample target tweets that illustrate the given scenario. The given problem was more formally introduced recently by [4]. The second sub-task organized as part of the *Mixed Script Information Retrieval* track [2] deals with the same problem, described in detail as follows.

Given a topic T, specified using a name, description and narrative; a set of associated English-Hindi code mixed tweets t_T; and a set of similarly code mixed queries Q^T against the topic, the task involves returning the top k results for every such query. A summary of the training dataset is presented in Table 2. Some of the key challenges thus, can be enumerated as:

1. **Short and comparable document lengths** : Judging relevance on *curated* tweets can be especially hard given that not only the documents are short in length, thus, giving little context to work with but also that all documents are comparable in length and are definitely somewhat relevant to the query / topic to begin with.
2. **Mixed language data** : Given that both the tweets themselves as well as the queries are in one or more languages, some sort of a cross lingual indexing and querying scheme must be employed to ensure relevant results.
3. **Spelling variants** : Specifically applicable given that one of the languages (Hindi) is expressed in roman script via transliteration. Given that no standardized spelling scheme exists or is widely used, the same words can appear with different spellings.

.

The rest of this paper is organized as follows. We begin by taking a closer look at the training dataset in Section 2 and draw some inferences with regards to the data and queries that act as guiding principles for the different techniques used. Then in Section 3, we describe the different data processing, indexing and query processing techniques that we tried out as part of our initial experiments. Then in Section 4, we provide details of the runs we submitted and the exact system composition for each run. Finally, in Section 5, we present our system performance on the test dataset and present an analysis of the results. We then conclude by discussing future work & potential system improvements.

2. TRAINING DATASET

The training dataset can be summarized as follows:

1. **Topics:** Total of 10 topics
2. **Queries:** Total of 23 queries, ranging from 1 to 4 per topic
3. **Tweets:** Total of 6142 tweets split between topics/queries and ranging from 34 (Topic 9 / Q 21) to 3531 (Topic 1 / Q 4)

Based on the question types and available data, we make the following assumptions / observations:

1. Each query is an *informational* query about some *Named Entity* (NE)
2. Every query can also be completely described by a *triple* as A[-*rel*-B] where [xx] indicates an *optional* clause
3. At least one of A or B is a Named Entity

S.no	Sample Query	Example Tweets
1	delhi me election	- rohit sharma ko dilli ka chunav ladwao !! newsmaker of the day
2	india ki haar	#aapkasting thanks india ki haar ka dukh bhula diya tum logo ne
3	nirbhayaa ka rapist	asharam bapu ka kehna hai mujhe nabalig samajhkar hi chod do . #nirbhayarapistout

Table 1: Code Mixed Tweets and Queries

Topic Num	Num Tweets	Vocabulary	% OOV
1	3894	12548	66.87
2	582	3356	60.58
3	82	690	60.86
4	54	483	56.78
5	410	2556	67.03
6	532	2641	59.59
7	51	487	66.15
8	104	901	62.55
9	32	307	55.17
10	425	1900	57.65

Table 2: Training Dataset summary

4. When not a NE, A or B, are most likely a *noun*
5. NEs would usually be represented by a fixed number of variants
6. Nouns on the other hand, could be represented by any number of synonyms that could be drawn from both languages (*chunaav* versus **election** for example)
7. The ordering of the clause (Noun-*rel*-NE versus NE-*rel*-Noun) is dependent on the underlying language (*chunaav* **in delhi** versus *dilli mein* **election**)

Apart from the training dataset, no relevance judgments were provided, binary or nuanced. The narrative with each topic provided some insight into what was considered relevant and non-relevant, but there were no clues to differentiate between two tweets given they both referenced the same number of query tokens. Thus, we need to derive some notion of *aboutness* or information content of each tweet to determine relevance. Thus, a possible search strategy can be formulated as:

1. Determine NE or NEs within the query and the tweets - this could be a fixed list or processed from Wikipedia or other Knowledge Bases (KBs)
2. Find tweets that match as many tokens or as much of the remaining query as possible
3. Find some proxy for tweet aboutness and use it to determine relevance

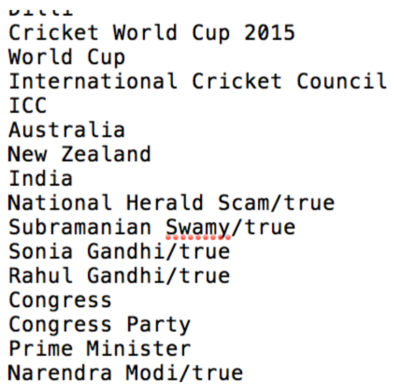
Using these three guiding principles, we now present the different techniques we tried out and how they contributed to our final system run.

3. INDEXING EXPERIMENTS

In this section, we describe the different experiments we conducted and the various system components that comprise our final runs.

3.1 Apache Solr

We used Apache Solr - an open source, scalable text search engine written in Java and built over Apache Lucene as the back-end for our system. It is fairly straightforward to index any sort of data in Solr after defining a schema. We present



```
Cricket World Cup 2015
World Cup
International Cricket Council
ICC
Australia
New Zealand
India
National Herald Scam/true
Subramanian Swamy/true
Sonia Gandhi/true
Rahul Gandhi/true
Congress
Congress Party
Prime Minister
Narendra Modi/true
```

Figure 1: Examples of extracted NEs

our full schema definition in Table 3. Thus, apart from the data already provided, we augment it using two fields : *NE* (see Section 3.2) and *cluster_id* (see Section 3.3).

3.2 NE tagger

We implemented the NE tagger in two parts : (a) a regex based extractor that generates a list of known NEs and (b) a simple string matching based tagger that are explained as follows. For the NE extractor, we used the topic description files and automatically extracted longest possible sequence of tokens wherein the sequence is delineated by a token that begins with a capital letter. We then sorted the list extracted as thus and eliminated duplicates and subsequences. However, for every sub-sequence that we eliminated, we tagged the parent entity as being capable of being present as a sub-string. For example, the entity *Narendra Modi* could be present as a whole or as individual tokens namely *Narendra* or *Modi*. At the end of this step, we had thus generated a list of NEs (of interest at least) and a boolean flag indicating sub-sequence occurrence. A snapshot of the extracted lists from the training dataset is shown in Figure 1.

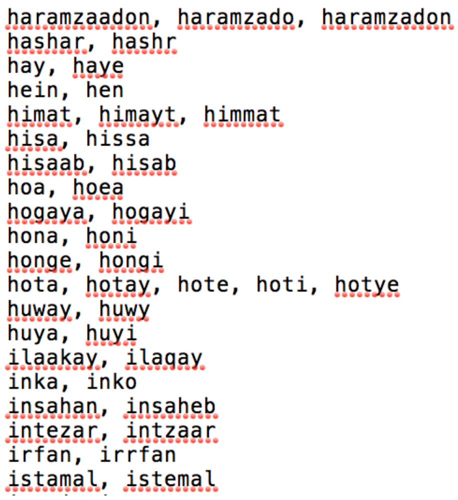
The actual tagger followed a similar philosophy where we started with the above list and tagged each tweet as follows. We simply created a mapping from token to entity, wherein the token was either defined as the full entity or any of its sub-tokens if it could be present partially. For example, in the above case, for *Narendra Modi*, we created two mappings as *Narendra* $\rightarrow$ *Narendra Modi* and *Modi* $\rightarrow$ *Narendra Modi*. For each tweet, we then performed a look-up and added the matching NE to the index if a match was found. Further, we also added a Solr plugin for the modified Levenshtein distance [7] to match NEs with spelling variants occurring due to transliteration across the two languages.

3.3 Clustering

Finally, we attempted to mine cross-lingual equivalents and spelling variants for different words. Much akin to the work of query expansion by [5], we first explored the usage

S.no	Field name	Type	Description
1	id	String	Unique identifier for each tweet, UUID generated from raw text
2	text	Text	Raw text for the tweet, stored after tokenization and minimal processing
3	qnum	Integer	Query Number
4	topicid	Integer	Topic Number
5	NE	Text	List of Named Entities referenced within this tweet
6	cluster_id	Integer	Union of cluster ids for all constituent tokens within the tweet

Table 3: Solr Schema definition



```
haramzaadon, haramzado, haramzadon
hashar, hashr
hay, haye
hein, hen
himat, himayt, himmat
hisa, hissa
hisaab, hisab
hoa, hoea
hogaya, hogayi
hona, honi
honge, hongi
hota, hotay, hote, hoti, hotye
huway, huwy
huya, huyi
ilaakay, ilaqay
inka, inko
insahan, insaheb
intezar, intzaar
irfan, irrfan
istamal, istemal
```

Figure 2: Examples of generated synonyms

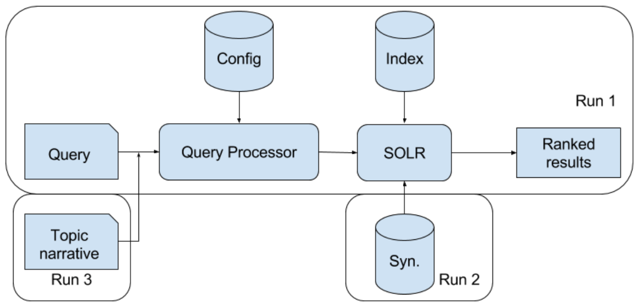


Figure 3: System Diagram

of Brown Clustering [3], a hierarchical clustering algorithm that exploits word distributional similarities. We used the freely available python implementation [6] and tested with a few cluster sizes. We found a value of 100 to be a reasonable balance between unrelated words (10-50) to fragmented clusters (500-1000). We also found that for the given setting, words with similar spellings (e.g. *jaise*, *jaisey*, *jaisay*) or inflectional variants (e.g. *ka*, *ke*, *ko*, etc.) were assigned to the same cluster. As an unintended side-effect however, the clustering also assigned similarly spelled English words with the same POS labels (i.e. verbs like cooking, cooling etc) to the same cluster.

Thus, we automatically generated a synonym list using the above results using a two step process. First, we removed all words that were found in a standard English dictionary. This was done to remove any false positives. Secondly, we deemed a set of words to be equivalents if their edit distance was less than a pre-determined threshold of three. The said threshold was manually determined by trial and error over a small test set. All such matching pairs thus generated were then combined into a single synonym file as partially shown in Figure 2.

Having described the individual components that comprise our system, we now turn our attention to presenting the overall system in the following section.

4. SYSTEM DESCRIPTION

The basic system architecture is shown in Figure 3. It includes the following components as introduced earlier: Solr (and its associated index), Query Processor (that includes NE tagger) and the extracted synonyms. Two additional inputs are (a) the search configuration that defines the different weights, choice of scoring mechanism etc: and (b) the topic narrative files that are additionally ingested as described below. Note that the diagram shows three different configurations, additive in nature for each of the three runs submitted.

Before we describe the three runs, we present some additional details on the query processor and Solr configuration. In continuation with the schema as presented in Table 3, given: - Query Q of n tokens as $q_1, q_2, \dots, q_n$ - Predetermined query weights $W_f = w_1, w_2, \dots, w_k$ for fields $f_1, f_2, \dots, f_k$ as stored within the configuration

The processor then partitions or generates field level parses of the query as $Q_f = q_1^f, q_2^f, \dots, q_k^f$ and passes the final query to Solr as $Q_f \cdot W_f$.

Secondly, as mentioned in Section 1, we also needed to figure out some notion of relevance. Given the lack of any additional information, we chose a simple voting based scheme. Solr supports a variety of different ranking models and thus, we configured four distinct Solr instances each with a different relevance scheme as enumerated below:

1. **Lucene Similarity:** This is an implementation of the classic tf-idf similarity that uses $\sqrt{tf}$ and $1 + log(\frac{df}{N+1})$ as normalization factors
2. **Okapi similarity:** A probabilistic relevance model as introduced by [8]
3. **Language model:** Uses Jelinek-Mercer smoothing on a language model as proposed by [9]
4. **DFR similarity:** The Divergence From Randomness models as suggested by [1]. We used the Inverse Expected document frequency (I(ne)) model with Laplacian smoothing and Zipfian normalization in our experiments.

For the first three models, we used the default settings and for DFR, we manually experimented with a few queries and returned results to settle on the selected model. For a given query, the system executes it in parallel on the four

Query no	Topic	Run 1	Run 2	Run 3	Avg MAP
1	1	0.0	0.0	0.0	0.0
2		0.0	0.0	0.0	0.0
3		0.0	0.0	0.0	0.0
4		0.0	0.0	0.0	0.0
5	2	0.0	0.0	0.0	0.0
6		0.003	0.003	0.003	0.003
7		0.177	0.1	0.1	0.126
8		0.019	0.008	0.019	0.015
9	3	0.0	0.0	0.0	0.0
10		0.007	0.022	0.007	0.012
11		0.05	0.05	0.05	0.05
12		0.004	0.009	0.004	0.006
Average		0.0217	0.016	0.0152	0.0176

Table 4: Test Dataset Summary and Results

subsystems, normalizes the returned scores and combines the results into a single ranked list.

We enumerate the three runs as follows:

1. **Run 1 - Named Entity boosts :** In the first run, we performed two levels of query matching - one was boosting the documents based on their NE matches from the query, i.e., the query was parsed to extract NEs and each document (tweet) that matched the given NE was provided a small numeric boost. The second level of boosting utilized phrase matching, i.e., documents that more closely matched the input query phrase were ranked higher than those that did not.
2. **Run 2 - Synonym expansion:** We merely expanded the given query based on these synonyms over the ranking mechanism presented for Run 1.
3. **Run 3 - Narrative based weighting:** For the final run, we extracted NEs from the provided topic narratives and assigned positive or negative boosts based on the associated word usage "relevant" and "not relevant". These additional weights were applied over the scheme presented in Run 2.

5. RESULTS & CONCLUSIONS

We present a summary of the test dataset and the results from each of the runs in Table 4. We make the following observations:

- Overall, the best system performance was our baseline system. Synonym expansion performs second best and narrative based weighting performs the worst.
- In hindsight, the techniques used by us perhaps improved recall at the cost of precision
- The only queries where synonym expansion works better than not having any synonyms is perhaps where there is a spelling mismatch between query and text (queries 10 and 12).
- Thus, we could have used "pessimistic" expansion - only use it if adequate results not available.
- When multiple NEs are present (queries 1-3), our system gets confused and returns non relevant results.

As part of future improvements, we thus need define a better notion of relevance and aboutness at a tweet level, specifically ascertain the information content of each tweet. In summary, our work showed three important results : (a) Named Entities have a very important role in IR on tweets and system performance could be improved by using more sophisticated NE taggers (b) simple clustering on topic wise tweets can give considerable insight into the constituent words - enough to derive spelling and inflectional variants and (c) in the given setting, precision is very sensitive to small changes and hence, typical recall improving techniques should be used as a "last resort".

6. REFERENCES

[1] G. Amati, C. Joost, and V. Rijsbergen. Probabilistic models for information retrieval based on divergence from randomness. 2003.

[2] S. Banerjee, K. Chakma, S. K. Naskar, A. Das, P. Rosso, S. Bandyopadhyay, and M. Choudhury. Overview of the Mixed Script Information Retrieval (MSIR) 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, 2016.

[3] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[4] K. Chakma and A. Das. Cmir: A corpus for evaluation of code mixed information retrieval of hindi-english tweets. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 2016.

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

[6] P. Liang. *Semi-supervised learning for natural language*. PhD thesis, Massachusetts Institute of Technology, 2005.

[7] N. Londhe, V. Gopalakrishnan, R. K. Srihari, and A. Zhang. Mess: A multilingual error based string similarity measure for transliterated name variants. In *Proceedings of the 7th Forum for Information Retrieval Evaluation*, pages 47–50. ACM, 2015.

[8] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, 109:109, 1995.

[9] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM, 2001.