DA-IICT in FIRE 2015 Shared Task on Mixed Script Information Retreival

Devanshu Jain
Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar, Gujarat, India
devanshu.jain919@gmail.com

ABSTRACT
This paper aims to describe the methodology followed by Team Watchdogs in their submission for the shared task on Mixed Script Information Retrieval (MSIR) in FIRE 2015. I participated in the subtask 1 (Query Word Labelling) and 2 (Mixed-script Ad hoc retrieval). For subtask 1, Machine Learning approach using CRF classifier was used to classify the tokens as one of the possible languages using n-gram and word2vec features. The method achieved a weighted F-measure of 0.805. For subtask 2, DFR similarity measure was used on the back-transliterated documents and queries (to Hindi with Vowel Signs replaced with actual vowels). The technique resulted in a NDCG@10 score of 0.7160.

Categories and Subject Descriptors
H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

Keywords
Information Retrieval, Mixed-Script Data, Natural Language Processing

1. INTRODUCTION
With the Internet becoming increasingly accessible, a linguistically diverse population has come online. It has been observed that such non-English population usually uses its own language written in Roman script ('Transliteration') to generate web-content like tweets, blogs, etc. Moreover, these people switch back and forth between languages mid-sentence, a behaviour termed as 'Code Switching'. This shared task aims to develop methods to retrieve content across scripts.

Subtask1: Query Word Labelling aims to detect the language of a token in a code-switched sentence. In addition to language detection, the subtask also requires to detect the named entities (people, organisation, etc.), punctuations and mixed words (i.e. words that belong to more than one language). The dataset, provided by the organisers, consisted of a list of annotated tweets. The distribution of all the labels in the dataset is provided in the table 1.

Subtask2: Mixed Script Ad hoc Retrieval aims to retrieve the documents containing relevant information for the query given to the system. The caveat is that the query as well as documents can be either in Hindi or English or both. So, retrieval needs to be done across script. The toy dataset, provided for experiment, consisted of 229 documents and 5 queries.

Table 1: Frequency of Label Tags in Training Data

Label	Frequency	Description of Tag
NE	2203	Named Entities
MIX	148	Mix of 2 languages
hi	4453	Hindi
en	17938	English
kn	1623	Kannada
ta	3153	Tamil
te	6475	Telugu
gu	890	Gujarati
mr	1960	Marathi
bn	3545	Bengali
ml	1160	Malyalam
O	8	Words of Foreign Language
X	7436	Punctuation, Numbers, Emoticons, etc.

Section 2 and 3 describes the methodology followed for the subtasks 1 and 2 respectively, in detail. Tools used to tackle these subtasks have aslo been mentioned. Section 4 specifies the results achieved by these methods.

2. SUBTASK 1: QUERY WORD LABELLING
2.1 Methodology
Before training, the following pre-processing was done on the data. The MIX tokens (i.e. tokens which are derived from two language) were not labelled in a consistent manner. For example, some words were labelled as: MIX_hi-en and some were labelled as: MIX_en-hi. Such instances were relabelled in a consistent way.

The problem was identified as sequence tagging. To tackle the problem, CRF was used. Two separate CRF models were trained for this subtask: one to identify the language and another to identify the named entity.

For training the Language Identification model, following features were used:

1. Character and Word N-Grams: For including the context features, individual tokens in the sentence were included within a token-window of 3 on each side of the word in consideration. For example, if the sentence is:

Table 2: Word N-Gram Features around the word ya

Feature	Value
w[-3]	ke
w[-2]	mat
w[-1]	maano
w[0]	ye
w[1]	birth
w[2]	se
w[3]	ab

Table 3: Character N-Gram Features for the word maano

Feature	Value
2_gram[-1][0]	ma
2_gram[-1][1]	aa
2_gram[-1][2]	an
2_gram[-1][3]	no
3_gram[-1][0]	maa
3_gram[-1][1]	aan
3_gram[-1][2]	ano
4_gram[-1][0]	maan
4_gram[-1][1]	aano

admin ke mat maano ye birth se ab tak single h

and the token in consideration is : ye, then the features used are as in table 2. Furthermore 2, 3 and 4 character n-grams of each of those words are also included as features. So, for w[-1] i.e. maano, generated features will be as in table 3

2. Dictionary for Hindi, Bengali and Gujarati: Dataset provided by IIT-Kharagpur consisting of following transliteration pairs: Hindi-English, Bangla-English, Gujarati-English, was used to determine the language of a word written in English script as shown in the algorithm 1.

3. Word2Vector Tweet Clustering: A feature vector was constructed for every word in the dataset using skip-gram implementation of word2vec with negative sampling. Then, these feature vectors were clustered using kMeans algorithm into 9 clusters (because there were 9 languages). Every word in consideration was assigned a cluster ID and this was used as a feature for generating a model for language detection module.

The main hypothesis is that using word2vec feature vectors' clusters as features during and dictionary mentions should improve the system's performance.

For training the Named Entity Recognition model, following additional features were also included apart from those mentioned above:

1. isFullCapitalised (Boolean): This feature tells whether the whole word is capitalised or not.

2. isFirstCapitalised (Boolean): This feature tells whether the first letter of the word is capitalised or not.

3. numCapitalised (Integer): This feature tells the number of capital letters in the word.

Algorithm 1 Algorithm for labelling

```
1: procedure label(token, ld-model, ne-model)
2:     ld-tag = getLabel(ld-model, token)
3:     ne-tag = getLabel(ne-model, token)
4:     final-tag = ne-tag
5:     if final-tag = O then
6:         final-tag = ld-tag
7:         (dict-tag,dict-freq)=getTagWithMaxFreqFromDict()
8:         if dict-tag ≠ O then
9:             final-tag = dict-tag
10:        end if
11:    end if
12: end procedure
```

4. isDot (Boolean): This feature tells whether the dot (.) character is present in the word or not.

5. numDot (Integer): This feature tells the number of dot characters in the word.

6. isDigit (Boolean): This feature tells the presence of a digit in the word.

7. numDigit (Integer): This feature tells the number of a digit in the word.

8. isSpecialChar (Boolean): This feature tells the presence of any special character like (, -, etc. in the word.

9. numSpecialChar (Integer): This feature tells the number of special characters in the word.

Capitalisation is often used for mention of important named entities. Dot character (.) is often used with abbreviations which, in most cases, is used to refer to a named entity. Digits and special characters are helpful in detecting the punctuations.

The procedure for labelling the token is explained in the algorithm 1. The constant O in line 5 is returned when the classifier can not identify any appropiate tag for the given token. So, if the token is not a named-entity, then it is tagged as one of the language tags using the corresponding model. The method "getTagWithMaxFreqFromDict()" in line 7 determines the language which has the most occurrences of the token.

2.2 External Tools Used

Following tools were used for this subtask:

1. CRFSuite was used to train the models for language detection and named-entity recognition based on the training data and for tagging the test files.

2. Deeplearning Word2Vec API was used for obtaining the word2vec model for each word of the training as well as test files. Number of iterations to train was set to 50 and feature vector size of each word was set to 100.

3. JavaML library's implementation of kMeans algorithm was used for clustering the words' feature vectors as obtained from DeepLearning Word2Vec API.

3. SUBTASK 2: MIXED SCRIPT AD-HOC RETRIEVAL

3.1 Methodology

Before indexing the documents, all the Roman words in documents as well as queries were transliterated back to Devanagri script. It has been observed that while transliterating Devanagri words to Roman, there are more spelling variations than in the case when transliterating from Roman to Devanagri. Then, the documents were indexed in the following 4 ways:

1. Run 1: Texts were tokenised at white spaces. Then, a Hindi Stemmer was used to stem these tokens to take into account the multiple variations of the token. For example, if the token is ख़रीदारों, then after stemming, it becomes खरीदार.
2. Run 2: All the white spaces and vowel signs were removed from the texts. For example, if the token is बॉलीवुड, then after removing all the vowel signs, it becomes बलवड. Then, character-level n-Grams were created for the texts where n ranged from 2 to 6.
3. Run 3: All the white spaces were removed from the texts and vowel signs were replaced by actual vowels. For example, if the token is बॉलीवुड, then after replacing all the vowel signs with the actual vowels, it becomes बऑलईवउड. Then, character-level n-Grams were created for the documents where n ranged from 2 to 6.
4. Run 4: Texts were tokenised at white spaces. Then, a Hindi Stemmer was used to stem these tokens. Furthermore, word-level n-Grams (called Shingles in Lucene vocabulary) were created for the documents where n ranged from 2 to 6.

Further, DFR similarity measure was used to find the most relevant documents for a particular query. Within the DFR, following settings were used:

1. Limiting form of Bose-Einstein model as a basic model of information content.
2. Laplace Law of Succession as first normalization.
3. Dirichlet Priors as second normalisation.

The main hypothesis are:

1. Indexing character level n-grams of texts should produce better results as compared to word level n-grams. The main reason for this is that character level n-grams are able to capture much more granular information and hence are able to account for minor spelling variations more effectively.
2. Indexing using word level n-grams should produce better results than indexing individual words.
3. System that replaces vowel signs with actual vowels should perform better than the one just removing them. This would prevent the loss of information as happening in the latter. The loss can result in some ambiguity. For example, दुखी and देखो - when vowel signs are removed, they both result in दख. However, when vowel signs are replaced by vowels, they both result in different words - दउखई and दएखओ, respectively.

3.2 External Tools Used

Following tools were used for this subtask:

1. Google Transliterator was used for transliterating the documents and queries back to Hindi language.
2. Apache Lucene was used to index the documents and search for the relevant documents according to the queries.

4. RESULTS AND DISCUSSION

4.1 Subtask 1

Three runs were submitted for the subtask. The methods deployed in each run has been described in table 4.

Table 4: Subtask 1 Runs

Run	Vocabulary Feature	Word2Vec Clustering Feature	Dictionary Feature
Run 1	✓	✓	✓
Run 2	✓	✓	×
Run 3	✓	×	×

The overall results achieved by deploying the aforementioned methods achieved results as described in table 5.

Table 5: Subtask 1 Results

Measure	Run 1	Run 2	Run 3
Tokens Accuracy	0.689	0.817	0.756
Average F-measure*	0.575	0.622	0.524
Weighted F-measure**	0.701	0.804	0.734

*: It was calculated as an average of f-measures of all the valid tags in the test-set.
**: It was a weighted average (weight by the frequency of a tag) of f-measures of all the valid tags in the test-set.

I had hypothesised that the use of dictionary and word2vec features will result in the improvement of the system's performance. Although, the use of word2vec features resulted in appreciable improvement in system's performance (almost 8% accuracy improvement), yet it was surprising to see that the use of dictionary in determining the tag actually decreased the system's performance. The main reason for this is that transliteration pairs were available for only 3 languages: Hindi, Gujarati and Bangla. Dictionary for rest of the 6 languages were ignored which may have caused the poor results.

A more granular specification of results (for language identification only) is given in table 6.

The system had a poor performance in identifying Gujarati words. One of the reasons for this is lack of sufficient mentions of the Gujarati words in the training dataset. One interesting observation was that many common Gujarati words

Table 6: Subtask 1 Strinct F-measures for Language Identification

Language	Run 1	Run 2	Run 3
Bengali	0.7613	0.8525	0.7205
English	0.6984	0.8511	0.8403
Gujarati	0.1582	0	0
Hindi	0.5522	0.8131	0.6995
Kannada	0.7324	0.7483	0.594
Malayalam	0.6287	0.6219	0.4644
Marathi	0.7074	0.8308	0.6354
Tamil	0.8249	0.8639	0.7346
Telugu	0.4603	0.5083	0.2418

like maru, karwu, pachi, etc. were tagged as being Hindi words. A high resemblance of Hindi and Gujarati language exacerbated the uneven distribution of labels in the dataset.

The answer to why in spite of having a sufficiently large number of mentions of Telugu words, the results for them were not as good still remains unknown.

4.2 Subtask 2

Four runs were submitted: one for each of the ways of indexing the documents as described in section 3.1. Table 7 specifies the overall results achieved by the methods. Table 8 specifies more specific results for the case of cross script retrieval.

One of the objective of the experiment was to determine which indexing technique produces better results - word or character level n-grams. As can be observed in the tables, the use of character level n-grams outperformed the use of word level n-grams.

The run where vowel signs are replaced by actual vowels performed much better than the case when they were completely removed, which proves our hypothesis, as stated earlier.

The hypothesis that indexing the word n-grams would produce better results than indexing individual stemmed words was proven wrong by the experiments' results. The reason for which is still not clear.

Table 7: Subtask 2 Overall Results

Measure	Run 1	Run 2	Run 3	Run 4
NDCG@1	0.6700	0.5267	0.6967	0.5633
NDCG@5	0.5922	0.5424	0.6991	0.5124
NDCG@10	0.6057	0.5631	0.7160	0.5173
MAP	0.3173	0.2922	0.3814	0.2360
MRR	0.4964	0.3790	0.5613	0.3944
Recall	0.3962	0.4435	0.4921	0.2932

5. FUTURE WORK

Currently, the system does not handle the mixed words (i.e. words formed by fusion of multiple languages). An effective algorithm needs to be formed to do so. A word2vec model of every language can be created separately. This model can be a list of feature vector of each word of that language. Then similarity of a word's feature vector to the model can be used to do this. This similarity can be calculated by averaging the hamming distance of the feature vector to every vector in the model of that particular language. It can also be used for language identification.

Table 8: Subtask 2 Cross Script Results

Measure	Run 1	Run 2	Run 3	Run 4
NDCG@1	0.4233	0.1833	0.3333	0.2900
NDCG@5	0.3264	0.2681	0.3864	0.2684
NDCG@10	0.3721	0.3315	0.4358	0.2997
MAP	0.2804	0.2168	0.3060	0.2047
MRR	0.4164	0.2757	0.4233	0.3244
Recall	0.3774	0.4356	0.5058	0.2914

Graph-Based N-gram Language Identification for short texts has been used by some people to identify the language in the code switched data. The method was used early in the system but it produced poor results when validated using 10-fold cross validation. The reason for this still needs to be found.

6. REFERENCES

[1] Transliteration Pairs for Hindi-English, Bangla-English and Gujarati-English. http://cse.iitkgp.ac.in/resgrp/cnerg/qa/fire13translit/index.html

[2] F. S. F. I. K. C. Czajkowski, K. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proceedings of the Eighteenth International Conference on Machine Learning.

[3] Parth Gupta et al. Query Expansion for Mixed-script Information Retrieval, in Proceedings of SIGIR 2014.

[4] Monojit Choudhury et. al. Overview of FIRE 2014 Track on Transliterated Search.

[5] Crfsuite: a fast implementation of conditional random fields (crfs). http://www.chokkan.org/software/crfsuite/

[6] Google Transliterator https://developers.google.com/transliterate/v1/getting_started

[7] Apache Lucene https://lucene.apache.org/

[8] Deeplearning4j's implementation of Word2vec http://deeplearning4j.org/word2vec.html

[9] E. Tromp and M. Pechenizkiy Graph-Based N-gram Language Identification on Short Texts Proceedings of the 20th Machine Learning conference of Belgium and The Netherlands, 2011.