IIIT-H System Submission for FIRE2014 Shared Task on Transliterated Search

Irshad Ahmad Bhat  Vandan Mujadia  Aniruddha Tammewar  
Riyaz Ahmad Bhat  Manish Shrivastava

Language Technologies Research Centre, 
International Institute of Information Technology, Hyderabad

FIRE2014 Shared Task on Transliterated Search
1 Introduction

2 Query Word Labeling
   • Description
   • Data
   • Methodology
     • Token Level Language Identification
     • Transliteration
   • Results

3 Hindi Song Lyrics Retrieval
   • Description
   • Data
   • Methodology
   • Results
Task Description

- **Shared Task on Transliterated Search:**
  - **Subtask-I: Query word labeling**
    - **Goal:** Token level language identification of query words in code-mixed queries and the transliteration of identified Indian language words into their native scripts.
    - **Approach:** Modeled both the language identification and transliteration of a query word as a classification problem.
  - **Subtask-II: Mixed-script Ad hoc retrieval for Hindi Song Lyrics.**
    - **Goal:** Retrieve a ranked list of songs from a corpus of Hindi song lyrics given an input query in Devanagari or transliterated Roman script.
    - **Approach:** Query expansion using edit distance, pruning using language modeling and re-ranking based on relevance.
Task Description

- **Shared Task on Transliterated Search:**
  - **Subtask-I: Query word labeling**
    - **Goal:** Token level language identification of query words in code-mixed queries and the transliteration of identified Indian language words into their native scripts.
    - **Approach:** Modeled both the language identification and transliteration of a query word as a classification problem.
  - **Subtask-II: Mixed-script Ad hoc retrieval for Hindi Song Lyrics.**
    - **Goal:** Retrieve a ranked list of songs from a corpus of Hindi song lyrics given an input query in Devanagari or transliterated Roman script.
    - **Approach:** Query expansion using edit distance, pruning using language modeling and re-ranking based on relevance.
Task Description

- **Shared Task on Transliterated Search:**
  - **Subtask-I:** Query word labeling
    - **Goal:** Token level language identification of query words in code-mixed queries and the transliteration of identified Indian language words into their native scripts.
    - **Approach:** Modeled both the language identification and transliteration of a query word as a classification problem.
  - **Subtask-II:** Mixed-script Ad hoc retrieval for Hindi Song Lyrics.
    - **Goal:** Retrieve a ranked list of songs from a corpus of Hindi song lyrics given an input query in Devanagari or transliterated Roman script.
    - **Approach:** Query expansion using edit distance, pruning using language modeling and re-ranking based on relevance.
Task Description

- Shared Task on Transliterated Search:
  - Subtask-I: Query word labeling
    - **Goal:** Token level language identification of query words in code-mixed queries and the transliteration of identified Indian language words into their native scripts.
    - **Approach:** Modeled both the language identification and transliteration of a query word as a classification problem.
  - Subtask-II: Mixed-script Ad hoc retrieval for Hindi Song Lyrics.
    - **Goal:** Retrieve a ranked list of songs from a corpus of Hindi song lyrics given an input query in Devanagari or transliterated Roman script.
    - **Approach:** Query expansion using edit distance, pruning using language modeling and re-ranking based on relevance.
Shared Task on Transliterated Search:

- **Subtask-I: Query word labeling**
  - **Goal**: Token level language identification of query words in code-mixed queries and the transliteration of identified Indian language words into their native scripts.
  - **Approach**: Modeled both the language identification and transliteration of a query word as a classification problem.

- **Subtask-II: Mixed-script Ad hoc retrieval for Hindi Song Lyrics.**
  - **Goal**: Retrieve a ranked list of songs from a corpus of Hindi song lyrics given an input query in Devanagari or transliterated Roman script.
  - **Approach**: Query expansion using edit distance, pruning using language modeling and re-ranking based on relevance.
Task Description

- **Shared Task on Transliterated Search:**
  - **Subtask-I: Query word labeling**
    - **Goal:** Token level language identification of query words in code-mixed queries and the transliteration of identified Indian language words into their native scripts.
    - **Approach:** Modeled both the language identification and transliteration of a query word as a classification problem.
  - **Subtask-II: Mixed-script Ad hoc retrieval for Hindi Song Lyrics.**
    - **Goal:** Retrieve a ranked list of songs from a corpus of Hindi song lyrics given an input query in Devanagari or transliterated Roman script.
    - **Approach:** Query expansion using edit distance, pruning using language modeling and re-ranking based on relevance.
Task Description

- **Shared Task on Transliterated Search:**
  - **Subtask-I: Query word labeling**
    - **Goal:** Token level language identification of query words in code-mixed queries and the transliteration of identified Indian language words into their native scripts.
    - **Approach:** Modeled both the language identification and transliteration of a query word as a classification problem.
  - **Subtask-II: Mixed-script Ad hoc retrieval for Hindi Song Lyrics.**
    - **Goal:** Retrieve a ranked list of songs from a corpus of Hindi song lyrics given an input query in Devanagari or transliterated Roman script.
    - **Approach:** Query expansion using edit distance, pruning using language modeling and re-ranking based on relevance.
Language Identification (LID) of query words in code-mixed queries

- Code-mixing - A socio-linguistic phenomenon
- prominent among multi-lingual speakers
- switch back and forth between two or more languages or language-varieties
- spoken and written communication
- sudden rise due to increase in social networking channels
- Why LID? Pre-requisite for various NLP tasks
- ∴ Performance of any NLP task $\propto$ amount and level of code-mixing
- e.g. Parsing, MT, ASR, IR & IE, Semantic Processing, etc.
Language Identification (LID) of query words in code-mixed queries

- Code-mixing - A socio-linguistic phenomenon
  - prominent among multi-lingual speakers
  - switch back and forth between two or more languages or language-varieties
  - spoken and written communication
  - sudden rise due to increase in social networking channels
  - Why LID? Pre-requisite for various NLP tasks
  - ∴ Performance of any NLP task ∝ amount and level of code-mixing
  - e.g. Parsing, MT, ASR, IR & IE, Semantic Processing, etc.
Language Identification (LID) of query words in code-mixed queries

- Code-mixing - A socio-linguistic phenomenon
- prominent among multi-lingual speakers
  - switch back and forth between two or more languages or language-varieties
  - spoken and written communication
- sudden rise due to increase in social networking channels
- Why LID? Pre-requisite for various NLP tasks
  - Performance of any NLP task \( \propto \) amount and level of code-mixing
  - e.g. Parsing, MT, ASR, IR & IE, Semantic Processing, etc.
Language Identification (LID) of query words in code-mixed queries

- Code-mixing - A socio-linguistic phenomenon
- prominent among multi-lingual speakers
- switch back and forth between two or more languages or language-varieties
- spoken and written communication
- sudden rise due to increase in social networking channels
- Why LID? Pre-requisite for various NLP tasks
  - Performance of any NLP task $\propto$ amount and level of code-mixing
  - e.g. Parsing, MT, ASR, IR & IE, Semantic Processing, etc.
Language Identification (LID) of query words in code-mixed queries

- Code-mixing - A socio-linguistic phenomenon
- prominent among multi-lingual speakers
- switch back and forth between two or more languages or language-varieties
- spoken and written communication
- sudden rise due to increase in social networking channels
- Why LID? Pre-requisite for various NLP tasks
- ∴ Performance of any NLP task $\propto$ amount and level of code-mixing
- e.g. Parsing, MT, ASR, IR & IE, Semantic Processing, etc.
Language Identification (LID) of query words in code-mixed queries

- Code-mixing - A socio-linguistic phenomenon
- prominent among multi-lingual speakers
- switch back and forth between two or more languages or language-varieties
- spoken and written communication
- sudden rise due to increase in social networking channels

Why LID? Pre-requisite for various NLP tasks

\[ \text{Performance of any NLP task} \propto \text{amount and level of code-mixing} \]

- e.g. Parsing, MT, ASR, IR & IE, Semantic Processing, etc.
Language Identification (LID) of query words in code-mixed queries

- Code-mixing - A socio-linguistic phenomenon
- prominent among multi-lingual speakers
- switch back and forth between two or more languages or language-varieties
- spoken and written communication
- sudden rise due to increase in social networking channels
- Why LID? Pre-requisite for various NLP tasks
  - Performance of any NLP task $\propto$ amount and level of code-mixing
  - e.g. Parsing, MT, ASR, IR & IE, Semantic Processing, etc.
Language Identification (LID) of query words in code-mixed queries

- Code-mixing - A socio-linguistic phenomenon
- prominent among multi-lingual speakers
- switch back and forth between two or more languages or language-varieties
- spoken and written communication
- sudden rise due to increase in social networking channels
- Why LID? Pre-requisite for various NLP tasks
- ∴ Performance of any NLP task ∝ amount and level of code-mixing
  - e.g. Parsing, MT, ASR, IR & IE, Semantic Processing, etc.
Language Identification (LID) of query words in code-mixed queries

- Code-mixing - A socio-linguistic phenomenon
- prominent among multi-lingual speakers
- switch back and forth between two or more languages or language-varieties
- spoken and written communication
- sudden rise due to increase in social networking channels
- Why LID? Pre-requisite for various NLP tasks
- ∵ Performance of any NLP task ∝ amount and level of code-mixing
- e.g. Parsing, MT, ASR, IR & IE, Semantic Processing, etc.
Description

- Back transliteration of Indic words to their native scripts.
  - Challenge - Enormous noise/variation in transliterated form particularly in social media.
  - Importance - Retrieval of relevant documents in native script for a Roman transliterated query.
- Example queries and their expected system output

<table>
<thead>
<tr>
<th>Input query</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>sachin tendulkar number of centuries</td>
<td>sachin\H tendulkar\H number\E of\E centuries\E</td>
</tr>
<tr>
<td>palak paneer recipe</td>
<td>palak\H=\pa/\lanak\E paneer\H=\pa/\na/iimatra\ra\E</td>
</tr>
<tr>
<td>mungeri lal ke hasseen sapney</td>
<td>mungeri\H=\mu/\pa\nu/\na\mamata\i\ra lal\E=\la/\lan ke\H=\ka \hasseen\E=\nu\mamata\i\ra sapney\H=\pa/\na\mamata\i\ra \la\E=\pa/\na\mamata\i\ra \</td>
</tr>
<tr>
<td>iguazu water fall argentina</td>
<td>iguazu\E water\E fall\E argentina\E</td>
</tr>
</tbody>
</table>

Table 1: Input query with desired outputs, where L is Hindi and has to be labeled as H
Description

- Back transliteration of Indic words to their native scripts.
  - Challenge - Enormous noise/variation in transliterated form particularly in social media.
  - Importance - Retrieval of relevant documents in native script for a Roman transliterated query.
- Example queries and their expected system output

<table>
<thead>
<tr>
<th>Input query</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>sachin tendulkar number of centuries</td>
<td>sachin\H tendulkar\H number\E of\E centuries\E</td>
</tr>
<tr>
<td>palak paneer recipe</td>
<td>palak\H=पालक\ recipe\E</td>
</tr>
<tr>
<td>mungeri lal ke hasseen sapney</td>
<td>mungeri\H=मुगरी lal\H=लाल ke\H=क hasseen\H=हस्सीन sapney\H=सपने</td>
</tr>
<tr>
<td>iguazu water fall argentina</td>
<td>iguazu\E water\E fall\E argentina\E</td>
</tr>
</tbody>
</table>

Table 1: Input query with desired outputs, where L is Hindi and has to be labeled as H
Description

- Back transliteration of Indic words to their native scripts.
  - Challenge - Enormous noise/variation in transliterated form particularly in social media.
  - Importance - Retrieval of relevant documents in native script for a Roman transliterated query.

- Example queries and their expected system output

<table>
<thead>
<tr>
<th>Input query</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>sachin tendulkar number of centuries</td>
<td>sachin\H tendulkar\H number\E of\E centuries\E</td>
</tr>
<tr>
<td>palak paneer recipe</td>
<td>palak\H=\palak\H paneer\H=\paneer\H=\recipe\E</td>
</tr>
<tr>
<td>mungeri lal ke hasseen sapney</td>
<td>mungeri\H=\mugging\E lal\H=\lal ke\H=\ke hasseen\H=\hasseen\E sapney\H=\napney</td>
</tr>
<tr>
<td>iguazu water fall argentina</td>
<td>iguazu\E water\E fall\E argentina\E</td>
</tr>
</tbody>
</table>

Table 1: Input query with desired outputs, where L is Hindi and has to be labeled as H
Description

- Back transliteration of Indic words to their native scripts.
  - Challenge - Enormous noise/variations in transliterated form particularly in social media.
  - Importance - Retrieval of relevant documents in native script for a Roman transliterated query.

- Example queries and their expected system output

<table>
<thead>
<tr>
<th>Input query</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>sachin tendulkar number of centuries</td>
<td>sachin\H tendulkar\H number\E of\E centuries\E</td>
</tr>
<tr>
<td>palak paneer recipe</td>
<td>palak\H=\pa\la\ka\ka\ka paneer\H=\pa\na\iimatra\ra recipe\E</td>
</tr>
<tr>
<td>mungeri lal ke haseen sapney</td>
<td>mungeri\H=\ma\umatra\anusvara\ga /ematra\ra /lal ke\H=\ka /ematra haseen\H=\ha\sa\iimatra\na sapney\H=\sa\pa\na \ematra</td>
</tr>
<tr>
<td>iguazu water fall argentina</td>
<td>iguazu\E water\E fall\E argentina\E</td>
</tr>
</tbody>
</table>

Table 1: Input query with desired outputs, where L is Hindi and has to be labeled as H.
Description

- Back transliteration of Indic words to their native scripts.
  - Challenge - Enormous noise/variation in transliterated form particularly in social media.
  - Importance - Retrieval of relevant documents in native script for a Roman transliterated query.
- Example queries and their expected system output

<table>
<thead>
<tr>
<th>Input query</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>sachin tendulkar number of centuries</td>
<td>sachin\H tendulkar\H number\E of\E centuries\E</td>
</tr>
<tr>
<td>palak paneer recipe</td>
<td>palak\H=पळक paneer\H=पनीर recipe\E</td>
</tr>
<tr>
<td>mungeri lal ke haseen sapney</td>
<td>mungeri\H=मुंगरी lal\H=लाल ke\H=क haseen\H=हसीन sapney\H=सपने</td>
</tr>
<tr>
<td>iguazu water fall argentina</td>
<td>iguazu\E water\E fall\E argentina\E</td>
</tr>
</tbody>
</table>

Table 1: Input query with desired outputs, where L is Hindi and has to be labeled as H
Data

- **Word Query Labeling** is meant for 6 language-pairs:
  - Hindi-English (H-E)
  - Gujarati-English (G-E)
  - Bengali-English (B-E)
  - Tamil-English (T-E)
  - Kannada-English (K-E)
  - Malayalam-English (M-E).

- **Data released contain the following:**
  - Monolingual corpora of English, Hindi and Gujarati.
  - Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
  - Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
  - A development set of 1000 transliterated code-mixed queries for each language pair.
  - A separate test set of \(\sim 1000\) queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:
- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:
- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:

- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:

- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:
- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:
- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:
- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:
- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:
- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:
- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:
- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:
- Monolingual corpora of English, Hindi, and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali, and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English, and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:
- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:
- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:
- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:
- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:
- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:
- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Data

- Word Query Labeling is meant for 6 language-pairs:
  - Hindi-English (H-E)
  - Gujarati-English (G-E)
  - Bengali-English (B-E)
  - Tamil-English (T-E)
  - Kannada-English (K-E)
  - Malayalam-English (M-E).

- Data released contain the following:
  - Monolingual corpora of English, Hindi and Gujarati.
  - Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
  - Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
  - A development set of 1000 transliterated code-mixed queries for each language pair.
  - A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:
- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:
- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:

- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:

- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of ~1000 queries for the evaluation of results.
Word Query Labeling is meant for 6 language-pairs:
- Hindi-English (H-E)
- Gujarati-English (G-E)
- Bengali-English (B-E)
- Tamil-English (T-E)
- Kannada-English (K-E)
- Malayalam-English (M-E).

Data released contain the following:
- Monolingual corpora of English, Hindi and Gujarati.
- Word lists with corpus frequencies for English, Hindi, Bengali and Gujarati.
- Word transliteration pairs for Hindi-English, Bengali-English and Gujarati-English.
- A development set of 1000 transliterated code-mixed queries for each language pair.
- A separate test set of \(\sim\)1000 queries for the evaluation of results.
Query word labeling is a similar problem to Document-level Language Identification task [1]. Query word labeling is a token level language identification problem while Document language identification is about deciphering the language a document is written in. More complex than Document-level Language Identification:

\[ \#\text{features}_{\text{Document-level}} > \#\text{features}_{\text{Word-level}} \]

Features available for Query word labeling are mostly restricted to word level like:
- word morphology
- syllable structure
- phonemic (letter) inventory
- \( n \)-gram models best suited for the task [2], [3], [5], [7], [6]
Token Level Language Identification

- Query word labeling is a similar problem to Document-level Language Identification task [1]
- Query word labeling is a token level language identification problem while Document language identification is about deciphering the language a document is written in.
- More complex than Document-level Language Identification
  \[ \#\text{features}_{\text{Document-level}} > \#\text{features}_{\text{Word-level}} \]
- Features available for Query word labeling are mostly restricted to word level like:
  - word morphology
  - syllable structure
  - phonemic (letter) inventory
- \( n \)-gram models best suited for the task [2], [3], [5], [7], [6]
Query word labeling is a similar problem to Document-level Language Identification task [1].

Query word labeling is a token level language identification problem while Document language identification is about deciphering the language a document is written in.

More complex than Document-level Language Identification

- \#features_{Document-level} > \#features_{Word-level}

Features available for Query word labeling are mostly restricted to word level like:
  - word morphology
  - syllable structure
  - phonemic (letter) inventory

\(n\)-gram models best suited for the task [2], [3], [5], [7], [6]
Token Level Language Identification

- Query word labeling is a similar problem to Document-level Language Identification task [1].
- Query word labeling is a token level language identification problem while Document language identification is about deciphering the language a document is written in.
- \[ \#\text{features}_{\text{Document-level}} > \#\text{features}_{\text{Word-level}} \]
- Features available for Query word labeling are mostly restricted to word level like:
  - word morphology
  - syllable structure
  - phonemic (letter) inventory
- \( n \)-gram models best suited for the task [2], [3], [5], [7], [6].
Query word labeling is a similar problem to Document-level Language Identification task [1].

Query word labeling is a token level language identification problem while Document language identification is about deciphering the language a document is written in.

More complex than Document-level Language Identification

$\therefore \ #\text{features}_{\text{Document-level}} > #\text{features}_{\text{Word-level}}$

Features available for Query word labeling are mostly restricted to word level like:

- word morphology
- syllable structure
- phonemic (letter) inventory

$n$-gram models best suited for the task [2], [3], [5], [7], [6]
Query word labeling is a similar problem to Document-level Language Identification task [1]

Query word labeling is a token level language identification problem while Document language identification is about deciphering the language a document is written in.

More complex than Document-level Language Identification

\[ \#\text{features}_{\text{Document-level}} > \#\text{features}_{\text{Word-level}} \]

Features available for Query word labeling are mostly restricted to word level like:

- word morphology
- syllable structure
- phonemic (letter) inventory

\( n \)-gram models best suited for the task [2], [3], [5], [7], [6]
Token Level Language Identification

- Query word labeling is a similar problem to Document-level Language Identification task [1]
- Query word labeling is a token level language identification problem while Document language identification is about deciphering the language a document is written in.
- More complex than Document-level Language Identification

  \[ \#\text{features}_{\text{Document-level}} > \#\text{features}_{\text{Word-level}} \]

- Features available for Query word labeling are mostly restricted to word level like:
  - word morphology
  - syllable structure
  - phonemic (letter) inventory

- \( n \)-gram models best suited for the task [2], [3], [5], [7], [6]
Token Level Language Identification

- Query word labeling is a similar problem to Document-level Language Identification task [1]
- Query word labeling is a token level language identification problem while Document language identification is about deciphering the language a document is written in.
- More complex than Document-level Language Identification
- \[ \# \text{features}_{\text{Document-level}} > \# \text{features}_{\text{Word-level}} \]
- Features available for Query word labeling are mostly restricted to word level like:
  - word morphology
  - syllable structure
  - phonemic (letter) inventory
- \( n \)-gram models best suited for the task [2], [3], [5], [7], [6]
Token Level Language Identification

- Query word labeling is a similar problem to Document-level Language Identification task [1]
- Query word labeling is a token level language identification problem while Document language identification is about deciphering the language a document is written in.
- More complex than Document-level Language Identification

\[ \# \text{features}_{\text{Document-level}} > \# \text{features}_{\text{Word-level}} \]

- Features available for Query word labeling are mostly restricted to word level like:
  - word morphology
  - syllable structure
  - phonemic (letter) inventory
- \( n \)-gram models best suited for the task [2], [3], [5], [7], [6]
Query Word Classification

- Language Identification as a classification problem
  - For each query word, predict its class from a finite set of classes. In our case classes labels are:
    - English
    - Any of the six Indian languages: Bengali, Hindi, Gujarati, Marathi, Malayalam and Tamil
    - Ambiguous
    - Named Entity
    - Other

- Features for classification
  - Letter-based n-gram posterior probabilities
  - Use of Dictionaries
Query Word Classification

- Language Identification as a classification problem
- For each query word, predict its class from a finite set of classes. In our case classes labels are:
  - English
  - Any of the six Indian languages: Bengali, Hindi, Gujarati, Marathi, Malayalam and Tamil
  - Ambiguous
  - Named Entity
  - Other
- Features for classification
  - Letter-based n-gram posterior probabilities
  - Use of Dictionaries
Query Word Classification

- Language Identification as a classification problem
- For each query word, predict its class from a finite set of classes. In our case, classes labels are:
  - English
  - Any of the six Indian languages: Bengali, Hindi, Gujarati, Marathi, Malayalam and Tamil
  - Ambiguous
  - Named Entity
  - Other

- Features for classification
  - Letter-based n-gram posterior probabilities
  - Use of Dictionaries
Language Identification as a classification problem

For each query word, predict its class from a finite set of classes. In our case, classes labels are:

- English
- Any of the six Indian languages: Bengali, Hindi, Gujarati, Marathi, Malayalam and Tamil
- Ambiguous
- Named Entity
- Other

Features for classification

- Letter-based n-gram posterior probabilities
- Use of Dictionaries
Query Word Classification

- Language Identification as a classification problem
- For each query word, predict its class from a finite set of classes. In our case classes labels are:
  - English
  - Any of the six Indian languages: Bengali, Hindi, Gujarati, Marathi, Malayalam and Tamil
  - Ambiguous
  - Named Entity
  - Other

- Features for classification
  - Letter-based n-gram posterior probabilities
  - Use of Dictionaries
Query Word Classification

- Language Identification as a classification problem
- For each query word, predict its class from a finite set of classes. In our case classes labels are:
  - English
  - Any of the six Indian languages: Bengali, Hindi, Gujarati, Marathi, Malayalam and Tamil
  - Ambiguous
  - Named Entity
  - Other

- Features for classification
  - Letter-based n-gram posterior probabilities
  - Use of Dictionaries
Query Word Classification

- Language Identification as a classification problem
- For each query word, predict its class from a finite set of classes. In our case classes labels are:
  - English
  - Any of the six Indian languages: Bengali, Hindi, Gujarati, Marathi, Malayalam and Tamil
  - Ambiguous
  - Named Entity
  - Other

Features for classification
- Letter-based n-gram posterior probabilities
- Use of Dictionaries
Query Word Classification

- Language Identification as a classification problem
- For each query word, predict its class from a finite set of classes. In our case classes labels are:
  - English
  - Any of the six Indian languages: Bengali, Hindi, Gujarati, Marathi, Malayalam and Tamil
  - Ambiguous
  - Named Entity
  - Other
- Features for classification
  - Letter-based n-gram posterior probabilities
  - Use of Dictionaries
Query Word Classification

- Language Identification as a classification problem
  - For each query word, predict its class from a finite set of classes. In our case classes labels are:
    - English
    - Any of the six Indian languages: Bengali, Hindi, Gujarati, Marathi, Malayalam and Tamil
    - Ambiguous
    - Named Entity
    - Other

- Features for classification
  - Letter-based n-gram posterior probabilities
  - Use of Dictionaries
Query Word Classification

- Language Identification as a classification problem
- For each query word, predict its class from a finite set of classes. In our case classes labels are:
  - English
  - Any of the six Indian languages: Bengali, Hindi, Gujarati, Marathi, Malayalam and Tamil
  - Ambiguous
  - Named Entity
  - Other

- Features for classification
  - Letter-based n-gram posterior probabilities
  - Use of Dictionaries
Train separate letter-based smoothed $n$-gram LMs for each language in a language pair

- $N$-gram LMs
  - Compute the conditional probability corresponding to $k^1$ classes $c_1, c_2, \ldots, c_k$ as:
    \[ p(c_i|w) = p(w|c_i) \times p(c_i) \] (1)

- Prior distribution $p(c)$ of a class is estimated from the respective training sets shown below.

<table>
<thead>
<tr>
<th>Language</th>
<th>Data Size</th>
<th>Average Token Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hindi</td>
<td>32,909</td>
<td>9.19</td>
</tr>
<tr>
<td>English</td>
<td>94,514</td>
<td>4.78</td>
</tr>
<tr>
<td>Gujarati</td>
<td>40,889</td>
<td>8.84</td>
</tr>
<tr>
<td>Tamil</td>
<td>55,370</td>
<td>11.78</td>
</tr>
<tr>
<td>Malayalam</td>
<td>12,811</td>
<td>13.18</td>
</tr>
<tr>
<td>Bengali</td>
<td>29,324</td>
<td>11.08</td>
</tr>
<tr>
<td>Kannada</td>
<td>87973</td>
<td>12.74</td>
</tr>
</tbody>
</table>

$k = 2$ for each LP
Posterior Probabilities

- Train separate letter-based smoothed $n$-gram LMs for each language in a language pair
- $N$-gram LMs
  - Compute the conditional probability corresponding to $k^1$ classes $c_1$, $c_2$, ..., $c_k$ as:
    \[ p(c_i|w) = p(w|c_i) \cdot p(c_i) \] (1)
  
- Prior distribution $p(c)$ of a class is estimated from the respective training sets shown below.

<table>
<thead>
<tr>
<th>Language</th>
<th>Data Size</th>
<th>Average Token Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hindi</td>
<td>32,909</td>
<td>9.19</td>
</tr>
<tr>
<td>English</td>
<td>94,514</td>
<td>4.78</td>
</tr>
<tr>
<td>Gujarati</td>
<td>40,889</td>
<td>8.84</td>
</tr>
<tr>
<td>Tamil</td>
<td>55,370</td>
<td>11.78</td>
</tr>
<tr>
<td>Malayalam</td>
<td>12,8118</td>
<td>13.18</td>
</tr>
<tr>
<td>Bengali</td>
<td>29,3240</td>
<td>11.08</td>
</tr>
<tr>
<td>Kannada</td>
<td>579736</td>
<td>12.74</td>
</tr>
</tbody>
</table>

\[ k = 2 \text{ for each LP} \]
Posterior Probabilities

- Train separate letter-based smoothed $n$-gram LMs for each language in a language pair
- $N$-gram LMs
  - Compute the conditional probability corresponding to $k^1$ classes $c_1, c_2, \ldots, c_k$ as:
    \[
    p(c_i|w) = p(w|c_i) \times p(c_i)
    \]  
  
- Prior distribution $p(c)$ of a class is estimated from the respective training sets shown below.

<table>
<thead>
<tr>
<th>Language</th>
<th>Data Size</th>
<th>Average Token Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hindi</td>
<td>32,909</td>
<td>9.19</td>
</tr>
<tr>
<td>English</td>
<td>94,514</td>
<td>4.78</td>
</tr>
<tr>
<td>Gujarati</td>
<td>40,889</td>
<td>8.84</td>
</tr>
<tr>
<td>Tamil</td>
<td>55,370</td>
<td>11.78</td>
</tr>
<tr>
<td>Malayalam</td>
<td>12,8118</td>
<td>13.18</td>
</tr>
<tr>
<td>Bengali</td>
<td>29,3240</td>
<td>11.08</td>
</tr>
<tr>
<td>Kannada</td>
<td>579736</td>
<td>12.74</td>
</tr>
</tbody>
</table>

$k = 2$ for each LP
Posterior Probabilities

- Train separate letter-based smoothed \( n \)-gram LMs for each language in a language pair
- \( N \)-gram LMs
  - Compute the conditional probability corresponding to \( k \) classes \( c_1, c_2, \ldots, c_k \) as:
    \[
    p(c_i|w) = \frac{p(w|c_i) \cdot p(c_i)}{p(w)}
    \]  
  - Prior distribution \( p(c) \) of a class is estimated from the respective training sets shown below.

<table>
<thead>
<tr>
<th>Language</th>
<th>Data Size</th>
<th>Average Token Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hindi</td>
<td>32,9091</td>
<td>9.19</td>
</tr>
<tr>
<td>English</td>
<td>94,514</td>
<td>4.78</td>
</tr>
<tr>
<td>Gujarati</td>
<td>40,889</td>
<td>8.84</td>
</tr>
<tr>
<td>Tamil</td>
<td>55,370</td>
<td>11.78</td>
</tr>
<tr>
<td>Malayalam</td>
<td>12,8118</td>
<td>13.18</td>
</tr>
<tr>
<td>Bengali</td>
<td>29,3240</td>
<td>11.08</td>
</tr>
<tr>
<td>Kannada</td>
<td>579,736</td>
<td>12.74</td>
</tr>
</tbody>
</table>

\( k = 2 \) for each LP
LM $p(w)$ is implemented as an $n$-gram model using the IRSTLM-Toolkit[4] with Kneser-Ney smoothing as:

$$p(w) = \prod_{i=1}^{n} p(l_i|l_{i-j}^{i-1})$$

where $l$ is a letter and $j^2$ is a parameter indicating the amount of context used.

\[ \Rightarrow j = 4 \quad \text{5-gram model} \]
Lib-linear SVM classifier

Trained separate SVM classifiers for each language pair

- Low dimensional feature vectors:
  - Posterior probabilities from both the language models in a language pair
  - Presence of a word in English dictionaries as a boolean feature. We use python’s PyEnchant-package with the following dictionaries:
    - en GB: British English
    - en US: American English
    - de DE: German
    - fr FR: French
Trained separate SVM classifiers for each language pair

- Low dimensional feature vectors:
  - Posterior probabilities from both the language models in a language pair
  - Presence of a word in English dictionaries as a boolean feature. We use python’s PyEnchant-package with the following dictionaries:
    - en_GB: British English
    - en_US: American English
    - de_DE: German
    - fr_FR: French
Trained separate SVM classifiers for each language pair

- Low dimensional feature vectors:
  - Posterior probabilities from both the language models in a language pair
  - Presence of a word in English dictionaries as a boolean feature. We use python’s PyEnchant-package with the following dictionaries:
    - en_GB: British English
    - en_US: American English
    - de_DE: German
    - fr_FR: French
Trained separate SVM classifiers for each language pair

- Low dimensional feature vectors:
  - Posterior probabilities from both the language models in a language pair
  - Presence of a word in English dictionaries as a boolean feature. We use python’s PyEnchant-package with the following dictionaries:
    - en_GB: British English
    - en_US: American English
    - de_DE: German
    - fr_FR: French
Lib-linear SVM classifier

Trained separate SVM classifiers for each language pair

- Low dimensional feature vectors:
  - Posterior probabilities from both the language models in a language pair
  - Presence of a word in English dictionaries as a boolean feature. We use python’s PyEnchant-package with the following dictionaries:
    - en_GB: British English
    - en_US: American English
    - de_DE: German
    - fr_FR: French
Trained separate SVM classifiers for each language pair

- Low dimensional feature vectors:
  - Posterior probabilities from both the language models in a language pair
  - Presence of a word in English dictionaries as a boolean feature. We use python’s PyEnchant-package with the following dictionaries:
    - en_GB: British English
    - en_US: American English
    - de_DE: German
    - fr_FR: French
Trained separate SVM classifiers for each language pair

- Low dimensional feature vectors:
  - Posterior probabilities from both the language models in a language pair
  - Presence of a word in English dictionaries as a boolean feature. We use python’s PyEnchant-package with the following dictionaries:
    - en_GB: British English
    - en_US: American English
    - de_DE: German
    - fr_FR: French
Trained separate SVM classifiers for each language pair

- Low dimensional feature vectors:
  - Posterior probabilities from both the language models in a language pair
  - Presence of a word in English dictionaries as a boolean feature. We use python’s PyEnchant-package with the following dictionaries:
    - en_GB: British English
    - en_US: American English
    - de_DE: German
    - fr_FR: French
Back Transliteration of Indic Words

Transliteration of Indic words from Roman to the respective native scripts

- Learn a classification model that can predict a phonetically equivalent letter sequence from target script for each letter sequence in a source script.

- Transliteration of the said 6 Indian languages is carried out in the following manner:
  - Convert Indic words in training data to WX for readability.
  - WX is a transliteration scheme for representing Indian languages in ASCII.
  - In WX, every consonant and every vowel has a single mapping into Roman script, which means there is no loss of information while conversion.
Transliteration of Indic words from Roman to the respective native scripts

- Learn a classification model that can predict a phonetically equivalent letter sequence from target script for each letter sequence in a source script.

- Transliteration of the said 6 Indian languages is carried out in the following manner:
  - Convert Indic words in training data to WX for readability.
Back Transliteration of Indic Words

Transliteration of Indic words from Roman to the respective native scripts

- Learn a classification model that can predict a phonetically equivalent letter sequence from target script for each letter sequence in a source script.

- Transliteration of the said 6 Indian languages is carried out in the following manner:
  - Convert Indic words in training data to WX for readability.
    - WX is a transliteration scheme for representing Indian languages in ASCII.
    - In WX every consonant and every vowel has a single mapping into Roman, that means there is no loss of information while conversion.
Back Transliteration of Indic Words

Transliteration of Indic words from Roman to the respective native scripts

- Learn a classification model that can predict a phonetically equivalent letter sequence from target script for each letter sequence in a source script.

- Transliteration of the said 6 Indian languages is carried out in the following manner:
  - Convert Indic words in training data to WX for readability.
    - WX is a transliteration scheme for representing Indian languages in ASCII.
    - In WX every consonant and every vowel has a single mapping into Roman, that means there is no loss of information while conversion.
Back Transliteration of Indic Words

Transliteration of Indic words from Roman to the respective native scripts

- Learn a classification model that can predict a phonetically equivalent letter sequence from target script for each letter sequence in a source script.

- Transliteration of the said 6 Indian languages is carried out in the following manner:
  - Convert Indic words in training data to WX for readability.
    - WX is a transliteration scheme for representing Indian languages in ASCII.
    - In WX every consonant and every vowel has a single mapping into Roman, that means there is no loss of information while conversion.
Transliteration of Indic words from Roman to the respective native scripts

- Learn a classification model that can predict a phonetically equivalent letter sequence from target script for each letter sequence in a source script.

- Transliteration of the said 6 Indian languages is carried out in the following manner:
  - Convert Indic words in training data to WX for readability.
    - WX is a transliteration scheme for representing Indian languages in ASCII.
    - In WX every consonant and every vowel has a single mapping into Roman, that means there is no loss of information while conversion.
Learn a transliteration model using ID3 Decision trees from the transformed training data of each language.

- The models are character based, mapping each character in Roman script to WX based on their context of previous 3 and next 3 characters.
- Training data available only for Hindi, Bengali and Gujarati.

Use the transliteration model to predict the equivalent of Romanized word in WX.

- Use Indic converter to convert WX to native script.
- For Telugu, Tamil and Malayalam, use Hindi WX transliteration model to predict WX forms.
  - Use Indic converter to convert WX to Devanagari.
  - Use Unicode encoding tables of these languages to extract the corresponding letters. Mapping Hindi Hexadecimal encoding to the encoding of other Indian languages is trivial.
Learn a transliteration model using ID3 Decision trees from the transformed training data of each language.

- The models are character based, mapping each character in Roman script to WX based on their context of previous 3 and next 3 characters.
- Training data available only for Hindi, Bengali and Gujarati.

Use the transliteration model to predict the equivalent of Romanized word in WX.

- Use Indic converter to convert WX to native script.
- For Telugu, Tamil and Malayalam, use Hindi WX transliteration model to predict WX forms.

- Use Indic converter to convert WX to Devanagari.
- Use Unicode encoding tables of these languages to extract the corresponding letters. Mapping Hindi hexadecimal encoding to the encoding of other Indian languages is trivial.
Learn a transliteration model using ID3 Decision trees from the transformed training data of each language.

- The models are character based, mapping each character in Roman script to WX based on their context of previous 3 and next 3 characters.
- Training data available only for Hindi, Bengali and Gujarati.

- Use the transliteration model to predict the equivalent of Romanized word in WX.
- Use Indic converter to convert WX to native script.
- For Telugu, Tamil and Malayalam, use Hindi WX transliteration model to predict WX forms.

- Use Indic converter to convert WX to Devanagari.
- Use Unicode encoding tables of these languages to extract the corresponding letters. Mapping Hindi Hexadecimal encoding to the encoding of other Indian languages is trivial.
Learn a transliteration model using ID3 Decision trees from the transformed training data of each language.

- The models are character based, mapping each character in Roman script to WX based on their context of previous 3 and next 3 characters.
- Training data available only for Hindi, Bengali and Gujarati.

Use the transliteration model to predict the equivalent of Romanized word in WX.

- Use Indic converter to convert WX to native script.
- For Telugu, Tamil and Malayalam, use Hindi WX transliteration model to predict WX forms.

- Use Indic converter to convert WX to Devanagari.
- Use Unicode encoding tables of these languages to extract the corresponding letters. Mapping Hindi Hexadecimal encoding to the encoding of other Indian languages is trivial.
Learn a transliteration model using ID3 Decision trees from the transformed training data of each language.

- The models are character based, mapping each character in Roman script to WX based on their context of previous 3 and next 3 characters.
- Training data available only for Hindi, Bengali and Gujarati.

Use the transliteration model to predict the equivalent of Romanized word in WX.

Use Indic converter to convert WX to native script.

For Telugu, Tamil and Malayalam, use Hindi WX transliteration model to predict WX forms.

- Use Indic converter to convert WX to Devanagari.

- Use Unicode encoding tables of these languages to extract the corresponding letters. Mapping Hindi Hexadecimal encoding to the encoding of other Indian languages is trivial.
Learn a transliteration model using ID3 Decision trees from the transformed training data of each language.

- The models are character based, mapping each character in Roman script to WX based on their context of previous 3 and next 3 characters.
- Training data available only for Hindi, Bengali and Gujarati.

Use the transliteration model to predict the equivalent of Romanized word in WX.

Use Indic converter to convert WX to native script.

For Telugu, Tamil and Malayalam, use Hindi WX transliteration model to predict WX forms.

- Use Indic converter to convert WX to Devanagari.
- Use Unicode encoding tables of these languages to extract the corresponding letters. Mapping Hindi hexadecimal encoding to the encoding of other Indian languages is trivial.
Learn a transliteration model using ID3 Decision trees from the transformed training data of each language.

- The models are character based, mapping each character in Roman script to WX based on their context of previous 3 and next 3 characters.
- Training data available only for Hindi, Bengali and Gujarati.

Use the transliteration model to predict the equivalent of Romanized word in WX.

Use Indic converter to convert WX to native script.

For Telugu, Tamil and Malayalam, use Hindi WX transliteration model to predict WX forms.

- Use Indic converter to convert WX to Devanagari.
- Use Unicode encoding tables of these languages to extract the corresponding letters. Mapping Hindi Hexadecimal encoding to the encoding of other Indian languages is trivial.
Learn a transliteration model using ID3 Decision trees from the transformed training data of each language.

- The models are character based, mapping each character in Roman script to WX based on their context of previous 3 and next 3 characters.
- Training data available only for Hindi, Bengali and Gujarati.

Use the transliteration model to predict the equivalent of Romanized word in WX.

- Use Indic converter to convert WX to native script.
- For Telugu, Tamil and Malayalam, use Hindi WX transliteration model to predict WX forms.
  - Use Indic converter to convert WX to Devanagari.
  - Use Unicode encoding tables of these languages to extract the corresponding letters. Mapping Hindi Hexadecimal encoding to the encoding of other Indian languages is trivial.
<table>
<thead>
<tr>
<th>Language Pair</th>
<th>BengaliEnglish</th>
<th>GujaratiEnglish</th>
<th>HindiEnglish</th>
<th>KannadaEnglish</th>
<th>MalayalamEnglish</th>
<th>TamilEnglish</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>0.835</td>
<td>0.986</td>
<td>0.83</td>
<td>0.939</td>
<td>0.895</td>
<td>0.983</td>
</tr>
<tr>
<td>LR</td>
<td>0.83</td>
<td>0.868</td>
<td>0.749</td>
<td>0.926</td>
<td>0.963</td>
<td>0.987</td>
</tr>
<tr>
<td>LF</td>
<td>0.833</td>
<td>0.923</td>
<td>0.787</td>
<td>0.932</td>
<td>0.928</td>
<td>0.985</td>
</tr>
<tr>
<td>EP</td>
<td>0.819</td>
<td>0.078</td>
<td>0.718</td>
<td>0.804</td>
<td>0.796</td>
<td>0.991</td>
</tr>
<tr>
<td>ER</td>
<td>0.907</td>
<td>1</td>
<td>0.887</td>
<td>0.911</td>
<td>0.934</td>
<td>0.98</td>
</tr>
<tr>
<td>EF</td>
<td>0.861</td>
<td>0.145</td>
<td>0.794</td>
<td>0.854</td>
<td>0.86</td>
<td>0.986</td>
</tr>
<tr>
<td>TP</td>
<td>0.011</td>
<td>0.28</td>
<td>0.074</td>
<td>0</td>
<td>0.095</td>
<td>0</td>
</tr>
<tr>
<td>TR</td>
<td>0.181</td>
<td>0.243</td>
<td>0.357</td>
<td>0</td>
<td>0.102</td>
<td>0</td>
</tr>
<tr>
<td>TF</td>
<td>0.021</td>
<td>0.261</td>
<td>0.122</td>
<td>0</td>
<td>0.098</td>
<td>0</td>
</tr>
<tr>
<td>LA</td>
<td>0.85</td>
<td>0.856</td>
<td>0.792</td>
<td>0.9</td>
<td>0.891</td>
<td>0.986</td>
</tr>
<tr>
<td>EQMF All(NT)</td>
<td>0.383</td>
<td>0.387</td>
<td>0.143</td>
<td>0.429</td>
<td>0.383</td>
<td>0.714</td>
</tr>
<tr>
<td>EQMF—NE(NT)</td>
<td>0.479</td>
<td>0.413</td>
<td>0.255</td>
<td>0.555</td>
<td>0.525</td>
<td>0.714</td>
</tr>
<tr>
<td>EQMF—Mix(NT)</td>
<td>0.383</td>
<td>0.387</td>
<td>0.143</td>
<td>0.437</td>
<td>0.492</td>
<td>0.714</td>
</tr>
<tr>
<td>EQMF—Mix and NE(NT)</td>
<td>0.479</td>
<td>0.413</td>
<td>0.255</td>
<td>0.563</td>
<td>0.675</td>
<td>0.714</td>
</tr>
<tr>
<td>EQMF All</td>
<td>0.004</td>
<td>0.007</td>
<td>0.001</td>
<td>0</td>
<td>0.008</td>
<td>0</td>
</tr>
<tr>
<td>EQMF—NE</td>
<td>0.004</td>
<td>0.007</td>
<td>0.001</td>
<td>0</td>
<td>0.008</td>
<td>0</td>
</tr>
<tr>
<td>EQMF—Mix</td>
<td>0.004</td>
<td>0.007</td>
<td>0.001</td>
<td>0</td>
<td>0.008</td>
<td>0</td>
</tr>
<tr>
<td>EQMF—Mix and NE</td>
<td>0.004</td>
<td>0.007</td>
<td>0.001</td>
<td>0</td>
<td>0.008</td>
<td>0</td>
</tr>
<tr>
<td>ETPM</td>
<td>72/288</td>
<td>259/911</td>
<td>907/2004</td>
<td>0/751</td>
<td>90/852</td>
<td>0/0</td>
</tr>
</tbody>
</table>

Table: Subtask-I: Token Level Results

LP, LR, LF: Token level precision, recall and F-measure for the Indian language in the language pair.
ETPM: Exact transliterated pair match
Description

- **Hindi Song Lyrics Retrieval** - A Information Retrieval plus linguistic phenomenon
  - also prominent among multi-lingual specific Indian speaker
  - switch back and forth between language scripts
  - rise due to increase in multi script same language content

- **Shared Task** - Multi-script Ad hoc retrieval for Hindi Song Lyrics

- **Why?**
  - To improve retrieval and relevance of IR systems
  - To increase search space
Hindi Song Lyrics Retrieval - A Information Retrial plus linguistic phenomenon
- also prominent among multi-lingual specific Indian speaker
- switch back and forth between language scripts
- rise due to increase in multi script same language content

Shared Task - Multi-script Ad hoc retrieval for Hindi Song Lyrics

Why? -
- To improve retrieval and relevance of IR systems
- To increase search space
Hindi Song Lyrics Retrieval - A Information Retrieval plus linguistic phenomenon

- also prominent among multi-lingual specific Indian speaker
- switch back and forth between language scripts
- rise due to increase in multi script same language content

Shared Task - Multi-script Ad hoc retrieval for Hindi Song Lyrics

Why? -

:: To improve retrieval and relevance of IR systems
:: To increase search space
**Description**

- **Hindi Song Lyrics Retrieval** - A Information Retrieval plus linguistic phenomenon
  - also prominent among multi-lingual specific Indian speaker
  - switch back and forth between language scripts
  - rise due to increase in multi script same language content

- **Shared Task** - Multi-script Ad hoc retrieval for Hindi Song Lyrics

- **Why?** -
  - To improve retrieval and relevance of IR systems
  - To increase search space
Hindi Song Lyrics Retrieval - A Information Retrieval plus linguistic phenomenon
- also prominent among multi-lingual specific Indian speaker
- switch back and forth between language scripts
- rise due to increase in multi script same language content

Shared Task - Multi-script Ad hoc retrieval for Hindi Song Lyrics

Why?
- To improve retrieval and relevance of IR systems
- To increase search space
Hindi Song Lyrics Retrieval - A Information Retrieval plus linguistic phenomenon

- also prominent among multi-lingual specific Indian speaker
- switch back and forth between language scripts
- rise due to increase in multi script same language content

Shared Task - Multi-script Ad hoc retrieval for Hindi Song Lyrics

Why? -

:: To improve retrieval and relevance of IR systems
:: To increase search space
Description

- Hindi Song Lyrics Retrieval - A Information Retrieval plus linguistic phenomenon
  - also prominent among multi-lingual specific Indian speaker
  - switch back and forth between language scripts
  - rise due to increase in multi script same language content

- Shared Task - Multi-script Ad hoc retrieval for Hindi Song Lyrics

- Why? -
  - To improve retrieval and relevance of IR systems
  - To increase search space
Description

- **Hindi Song Lyrics Retrieval** - A Information Retrial plus linguistic phenomenon
  - also prominent among multi-lingual specific Indian speaker
  - switch back and forth between language scripts
  - rise due to increase in multi script same language content

- **Shared Task** - Multi-script Ad hoc retrieval for Hindi Song Lyrics

- **Why?** -
  - To improve retrieval and relevance of IR systems
  - To increase search space
Documents (≈60000) contain lyrics both in Devanagari and Roman scripts

Data Normalization -

- Cleaning of unwanted content and specific word handling (i.e. jahaa.N, jahaan, mann, D, etc.)
- Converted all documents in uniform Roman script
Documents (≈60000) contain lyrics both in Devanagari and Roman scripts

Data Normalization - -

- Cleaning of unwanted content and specific word handling (i.e. jahaa.N, jahaan, mann, D, etc.)
- Converted all document in uniform Roman script
Data and Data Normalization

- Documents (≈60000) contain lyrics both in Devanagari and Roman scripts
- Data Normalization -
  - Cleaning of unwanted content and specific word handling (i.e. jahaa.N, jahaan, mann, D, etc.)
  - Converted all document in uniform Roman script
Documents (≈60000) contain lyrics both in Devanagari and Roman scripts

Data Normalization -

- Cleaning of unwanted content and specific word handling (i.e. jahaa.N, jahaan, mann, D, etc.)
- Converted all document in uniform Roman script
Posting list and Relevancy

- Build index from the scratch on unified roman scripted song data
- Use conventional TF-IDF metric
- Parse song lyric document for relevancy measure
- Title of the song ■ First line of song ■ First line of stanzas ■ Each line of chorus ■ etc.
Posting list and Relevancy

- Build index from the scratch on unified roman scripted song data
- Use conventional TF-IDF metric
- Parse song lyric document for relevancy measure
- Title of the song
- First line of song
- First line of stanzas
- Each line of chorus
- etc.
Posting list and Relevancy

- Build index from the scratch on unified roman scripted song data
- Use conventional TF-IDF metric
- Parse song lyric document for relevancy measure
  - Title of the song
  - First line of song
  - First line of stanzas
  - Each line of chorus
  - etc.
Posting list and Relevancy

- Build index from the scratch on unified roman scripted song data
- Use conventional TF-IDF metric
- Parse song lyric document for relevancy measure
- Title of the song ✗ First line of song ✗ First line of stanzas ✗ Each line of chorus ✗ etc.
Query Expansion

- Includes identifying script of seed query and expanding it in terms of spelling variation

**Why?** -

∵ To improve the recall of the retrieval system

**How?** -

∵ Edit Distance + Language Modelings (To rank and limit generated query).
Query Expansion

- Includes identifying script of seed query and expanding it in terms of spelling variation

  Why? -

  ∴ To improve the recall of the retrieval system

- How? -

  ∴ Edit Distance + Language Modelings (To rank and limit generated query)
Query Expansion

- Includes identifying script of seed query and expanding it in terms of spelling variation

- Why? -
  ∴ To improve the recall of the retrieval system

- How? -
  ∴ Edit Distance + Language Modelings (To rank and limit generated query).
Query Expansion

- Includes identifying script of seed query and expanding it in terms of spelling variation

  Why? -
  
  ∴ To improve the recall of the retrieval system

  How? -
  
  ∴ Edit Distance + Language Modelings (To rank and limit generated query).
Query Expansion

- Includes identifying script of seed query and expanding it in terms of spelling variation

Why? -

∵ To improve the recall of the retrieval system

How? -

∵ Edit Distance + Language Modelings (To rank and limit generated query).
System flow
## Results

### Table: Subtask-II Results

<table>
<thead>
<tr>
<th>TEAM</th>
<th>NDCG@1</th>
<th>NDCG@5</th>
<th>NDCG@5</th>
<th>Map</th>
<th>MRR</th>
<th>RECALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>bits-run-2</td>
<td>0.7708</td>
<td>0.7954</td>
<td>0.6977</td>
<td>0.6421</td>
<td>0.8171</td>
<td>0.6918</td>
</tr>
<tr>
<td>iiiith-run-1</td>
<td>0.6429</td>
<td>0.5262</td>
<td>0.5105</td>
<td>0.4346</td>
<td>0.673</td>
<td>0.5806</td>
</tr>
<tr>
<td>bit-run-2</td>
<td>0.6452</td>
<td>0.4918</td>
<td>0.4572</td>
<td>0.3578</td>
<td>0.6271</td>
<td>0.4822</td>
</tr>
<tr>
<td>dcu-run-2</td>
<td>0.4143</td>
<td>0.3933</td>
<td>0.371</td>
<td>0.2063</td>
<td>0.3979</td>
<td>0.2807</td>
</tr>
</tbody>
</table>
Thank You!
Questions?
Timothy Baldwin and Marco Lui.  
Language identification: The long and the short of the matter. 

Ted Dunning.  
Statistical identification of language. 
Computing Research Laboratory, New Mexico State University, 1994.

Heba Elfardy and Mona T Diab.  
Token level identification of linguistic code switching. 
In COLING (Posters), pages 287–296, 2012.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo.  
Iristlm: an open source toolkit for handling large scale language models. 
In Interspeech, pages 1618–1621, 2008.

Ben King and Steven P Abney.  
Labeling the languages of words in mixed-language documents using weakly supervised methods.  

Marco Lui, Jey Han Lau, and Timothy Baldwin.  
Automatic detection and language identification of multilingual documents.  

Dong Nguyen and A Seza Dogruoz.  
Word level language identification in online multilingual communication.  