Code Switching (CS) is the alternation between languages in conversations. Wide-spread among Multi-lingual speakers. Examples,
- Spanglish = Spanish + English (Puerto-Rico, Mexico, US)
- Hinglish = Hindi + English (South-East Asia)
- Chinglish = Cantonese + English (Hong Kong)

Why Care?
A large section of humanity ignored by NLP community because they lack:
- Standardized Datasets
- Language Models
- Language Parsers
- And More!

Why Language Modeling (LM)?
- Important for downstream applications: Machine Translation, Speech Recognition, etc.
- CS data severely lacks annotation - needs unsupervised methods.

Challenges for NLP?
- Undefined grammar rules (Matrix vs. Embedded).
- Occur in informal contexts (hard to get, extremely noisy).

What Data?
- Authors collected and curated 59189 unique CS sentences. Data was gathered from a web crawl of eight Hinglish blogging websites that were returned by popular search engines (such as Google and Bing).
- All sentences were drawn from the domains of health and technology.
- Lexical language identification was performed at the word level for all sentences.
- Sentences that did not have at least one word each from both languages were discarded to channel our problem towards tackling intra-sentential code-switching.

What Next?
We are able to improve the State-of-The-Art language model for monolingual text by explicitly encoding the language information to perform this task for CS domain.

Model Architecture
- Implemented a baseline model of AWD-LSTM.
- 4 model architectures differ depending upon whether the language information is encoded or decoded.
- The distributional aspects of switching between languages captured by 16 dimensional language embedding.

Spelling Normalization is done by:
- Language Identification.
- Transliteration of romanized Hindi to its most likely Devanagari spelling using soundex encodings.

What Did We Do?
We trained and evaluated 4 different models.

We address the task of language modeling in Hinglish text with a dual objective:
- Predicting the next word.
- Predicting the language of the next word.

Underlying Theme: Simultaneous learning on multiple language tasks with shared parameterized layers improves generalization.

What Could Be Better?
- Non-standardized spellings in the romanized Hinglish text cannot always be found in the MUSE embeddings.

What Happened?
- We trained and evaluated 4 different models.

Note that these experiments are ongoing and we believe that we can improve these results with further hyperparameter tuning.

<table>
<thead>
<tr>
<th>Model/Data</th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base AWD-LSTM Model</td>
<td>10.08</td>
<td>19.73</td>
<td>20.92</td>
</tr>
<tr>
<td>Language Aware Encoder AWD-LSTM</td>
<td>10.07</td>
<td>19.00</td>
<td>20.18</td>
</tr>
<tr>
<td>Language Aware Decoder AWD-LSTM</td>
<td>11.60</td>
<td>20.72</td>
<td>22.01</td>
</tr>
<tr>
<td>Language Aware Encoder &amp; Decoder AWD-LSTM</td>
<td>9.47</td>
<td>18.51</td>
<td>19.52</td>
</tr>
</tbody>
</table>

Table 2: Perplexity scores of different models

6. Conclusion and Future Work

What Next?
- We aim to incorporate FastText word embeddings for Devanagari.
- End to End character level models is another direction to try.
- We welcome any and all feedback on any portion of this work.