Language Informed Modeling of Code-Switched Text

Khyathi Chandu **Thomas Manzini** **Sumeet Singh**

Language Technologies Institute, Carnegie Mellon University
{kchandu,tmanzini,sumeets}@andrew.cmu.edu

Abstract

We approach Code-Switching through Language Modeling (LM) on a corpus of Hinglish (Hindi + English) that we collected from blogging websites, containing 59,189 unique sentences. We implement and discuss different Language Models derived from a multi-layered LSTM architecture. Our main hypothesis is that providing language id information explicitly builds a robust language model as opposed to simple word level models by learning the switching points. We make an attempt at this in two ways: (1) factored model learning embeddings both for input word and input language, and (2) multi-task learning of predicting the language of the next word along with the word itself. We show that our highest performing model achieves a test perplexity of 19.52 on the CS corpus that we collected and processed. On this data we demonstrate that our performance is an improvement over AWD-LSTM LM (a recent State of Art on monolingual English).

1 Introduction

Code Switching (CS) is a widely studied linguistic phenomenon where two different languages are interleaved. This occurs within multilingual communities (Poplack, 1980; Myers-Scotton, 1997; Muysken, 2000; Bullock and Toribio, 2009). Typically one language (the *matrix language*) provides the grammatical structure for CS text and words are inserted from another language (the *embedded language*).

However, CS data is quite challenging to obtain because this phenomena is usually observed in informal settings. Data obtained from online sources are often noisy because of spelling, script, morphological, and grammatical variations. These sources of noise make it quite challenging to build robust NLP tools (Çetinoğlu et al., 2016). Our goal is to improve LM for Hindi-English code-mixed data (*Hinglish*) where similar challenges are apparent. The task of language modeling is very important to several downstream applications in NLP including speech recognition, machine translation etc., This task also aids in multi-task learning where parameters are shared between the language model and the other task. This is particularly important in domains such as code-switching, that lack annotated data, where the necessity to leverage unsupervised techniques are a must. In this report, we present our attempt to this problem using two techniques: (1) Factored language model that contains the factors of word and language given to a stacked LSTM based architecture, and (2) Posing this as a multi-task learning (MTL) problem with a dual objective including predicting the next word, and predicting the language of the next word. Learning to predict the language of the next word allows the model to switch points between languages at global level. In the path towards building the MTL architecture, we attempted to explicitly learn the language of the current word without predicting the language of the next word. This is one of the model in our ablation studies, which is similar to and is inspired from the factored LM.

In addition to the techniques used for monolingual language modeling, providing information about the language is a key component in CS domain. Our main goal in this paper is to examine the effect of language information in modeling CS text. We approach this systematically by experimenting with ablations of encoding and decoding language id along with the word itself. In this way, the model implicitly learns the switch points between the languages. We achieve the least perplexity score in combination with a language informed encoder and a language informed decoder among the ablation of multiple models.

The current material begins with a review of LM techniques for CS text in section 2. Then we describe our data collection and processing steps in Section 3 and model architecture in Section 4. We present a brief quantitative and qualitative discussion of our observations in Section 5. Finally in section 6 we propose promising directions for future work and conclude in section 7.

2 Related Work

Early efforts to develop computational framework for CS data include Joshi (1982) in the 1980s and Goyal et al. (2003); Sinha and Thakur (2005); Solorio and Liu (2008a,b) in the 2000s. However, these methods are quite limited in their applicability to the kind of data we see on the Internet and social media where code-switching data is usually found. This has led to an increased attention of the NLP community in the areas of Language Modeling (Li and Fung, 2013, 2014; Adel et al., 2015, 2013a,b; Garg et al., 2017), POS tagging (Vyas et al., 2014; Jamatia et al., 2015; Çetinoğlu and Çöltekin, 2016), language identification (King and Abney, 2013) prediction of code-switch points (Das and Gambäck, 2014), sentiment analysis (Rudra et al., 2016) and also certain meta level studies that include understanding metrics to characterize code-mixing (Patro et al., 2017; Guzmán et al., 2017).

2.1 Neural Language Models

Neural Language Models have a limitation in capturing only a finite context. This was overcome in Recurrent Neural Network (RNN) based Language Model (Mikolov et al., 2010). The RNN has an input layer x, hidden layer s (also called context layer or state) and output layer y. What allows for capturing of an infinite context is that the input to the network at time t i.e. $x(t)$ is concatenated with $s(t-1)$ i.e. the output of hidden layer at time $t-1$ to compute the current state $s(t)$ and output $y(t)$ which is probability of the next word given the context. Several word-level tasks can be tackled better at the constituent character-level (Zhang et al., 2015; Chung et al., 2016). Character-Level Language Models (Kim et al., 2016) represent a word w as a collection of the embeddings of the constituent characters $\mathbb{C}^w = [c_1, c_2, \cdots, c_l]$ where l is the length of w. $\mathbb{C}^w$ is then passed through a Convolution Neural Network(CNN) with varying filter widths to capture different lengths of character n-grams. Essentially, if there are h filters, then the feature mapping for w is $y^w = [y_1, y_2, \cdots, y_h]$. This feature mapping instead of word embeddings is fed to the RNN-LM as described above, and rest of the procedure remains the same. Adel et al. (2015) suggest using syntactic and semantic features found in CS text into a factored language model(FLM) that are indicative of a switch e.g. trigger words, trigger POS tags, brown cluster of function and content words, resulting in significant reduction in perplexity. (DLM) (Garg et al., 2017) combines two monolingual language models by using a probabilistic 'switch' between them.

2.2 Code-Switched Language Models

There has been some recent focus on adapting existing language models for CS text. (Li and Fung, 2013, 2014) use a translation model together with the language model of the matrix language to model the mixed language. The search space within the translation model is reduced by linguistic features in CS texts like inversion constraint and functional head constraint (Sankoff and Poplack, 1981).

In another approach Adel et al. (2015), use a Factored Language Model (FLM) that includes syntactic and semantic features found in CS text that are indicative of a switch e.g.

Criteria	Train	Dev	Test
# Sentences	35513	11839	11837
Avg Length of Sentences	18.90	17.58	18.22
Multilingual Index	0.8892	0.8905	0.8914
Language Entropy	0.6635	0.6639	0.6641
Integration Index	0.3304	0.3314	0.3312
Unique Unigrams	35,769	18,053	19,330
Unique Bigrams	276,552	125,108	130,947
Unique Trigrams	553,866	219,098	229,967

Table 1: Hinglish Data Statistics

trigger words, trigger POS tags, brown cluster of function and content words that result in significant reduction in perplexity.

Another recent method called Dual Language Model (DLM) (Garg et al., 2017), combines two monolingual language models by introducing a '*switch*' token common to both languages. Predicting this word in either languages acts a proxy to the probability of a switch and the next word is then predicted using the LM of the language that was switched to.

Among neural methods, (Adel et al., 2013a) use of a Recurrent Neural Network based LM to predict the language of the next word along with the actual word to model CS text. Following on these intuitions, our models are built on top of the AWD-LSTM LM (Merity et al., 2017) that was chosen due to its accessibility and high performance (recently State of the Art) on the Penn-Tree Bank and Wikitext-2 dataset (Merity et al., 2016). Extensive work has been done on this model through investigation on relative importance of hyper-parameters (Merity et al., 2018).

3 Data Collection and Analysis

To the best of our knowledge, there is no standard dataset to evaluate LM in code-switched texts. In this section, we will describe about our data collection and a brief analysis of it.

3.1 Data Collection

Curating a reasonable dataset for CS text is an important challenge for researchers in this domain. To the knowledge of the authors, there is no benchmark CS corpus for language modeling as there is for English (Merity et al., 2016; Marcus et al., 1994). CS is commonly observed in informal settings and in casual conversations. Hence the two potential source choices to gather data include social media (such as Twitter and Facebook) and blogging websites. We decided to go with the latter due to comparatively lesser noise and availability of more descriptive text. Our CS LM data was collected after having crawled eight blogging Hinglish websites[1] that were returned by popular search engines (such as Google and Bing) with simple code-switched queries in the domains of health and technology. These CS texts cover several different topics, primarily technical reviews of electronic and general e-commerce products as well as several health related articles. These texts were all tokenized at the sentence level and lexical language identification was performed. All the 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. This resulted in a total of 59,189 unique sentences. The data needed extensive cleaning due to a lot of hyperlinking text.

3.2 Data Analysis

To estimate the quality and extent of mixing and frequency of switching in our data, we measured Multilingual index (M-Index),

[1] Hinglish blogging websites:
www.hinglishpedia.com
www.qureshiinfotech.com
www.hindimehelp.com
www.pakkasolutionhindi.com
www.myhelplive.blogspot.com
www.seekhoweb.com
www.onlinesikhe.com

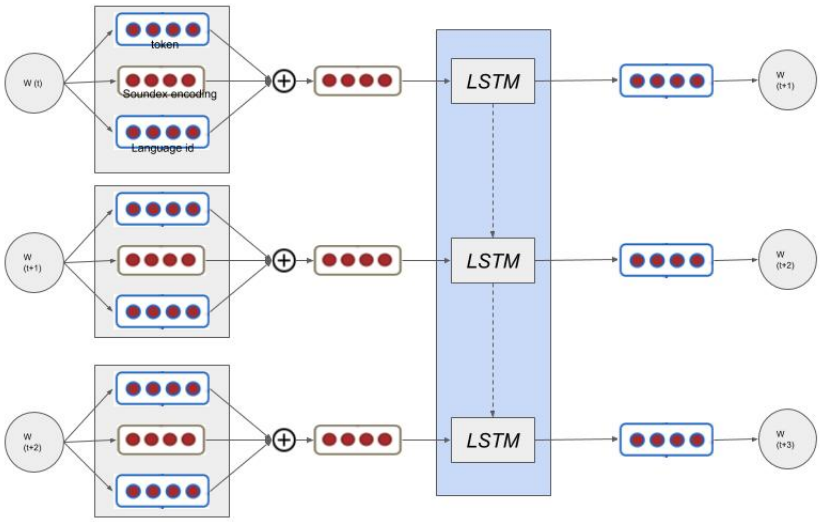

Figure 1: Architecture of our baseline Factored LM for Hinglish data

Language Entropy and Integration index (I-index) that were introduced in the domain of CS by (Guzmán et al., 2017). A multilingual index of 1 indicates that there is equal extent of mixing from both the participating languages. As we can observe, the mixing is nearly in the range of 0.8 which indicates that both Hindi and English are participating in the ratio of 4 is to 5 respectively. The metric itself does not reveal about the which is the embedded language and which is the matrix language. These metrics along with other n-gram statistics over our data are presented in Table 1. Note that the CS metrics for each of the train, validate and test splits of the data are almost the same, indicating a similar extent of mixing in them.

3.3 Pre-processing

One of the important characteristics of code-switched text in Hinglish when written in Roman script is the non-standard representations of the words. This is very commonly observed since there are strict guidelines that monitor the correctness of an approximately phonetically represented word. There is no standard one to one correspondence of syllables in Hindi when written in Roman script, that universally everyone would follow in informal settings. Hence the same word could be written in multiple representations based on the idiosyncrasies of the individual and perception of Romanization of a syllable to that specific person.

For both the Factored LM and language informed encoder models, we perform lexical language identification for each of the words and annotate them with this information.

The non-standardized representations of the words are dealt in the following two ways:

- Soundex Encodings: We use *soundex encodings* of the words as another factor. This way, the idiolectic representational variations with an additional 'h' for aspirated sounds and single vs multiple vowels for long and short syllables etc, would be mapped to the same space. This is not a solution to normalizing the representational variations but we hypothesize that this feature helps in learning context around variants of the same word better.

- Transliteration: Along the similar lines of standardizing the multiple representations of the same word, the words that are identified as Hindi at the lexical level are transliterated into their Devanagari representation and the most likely Devanagari spelling in a pre-existing Devanagri dictionary is chosen according to soundex encoding.

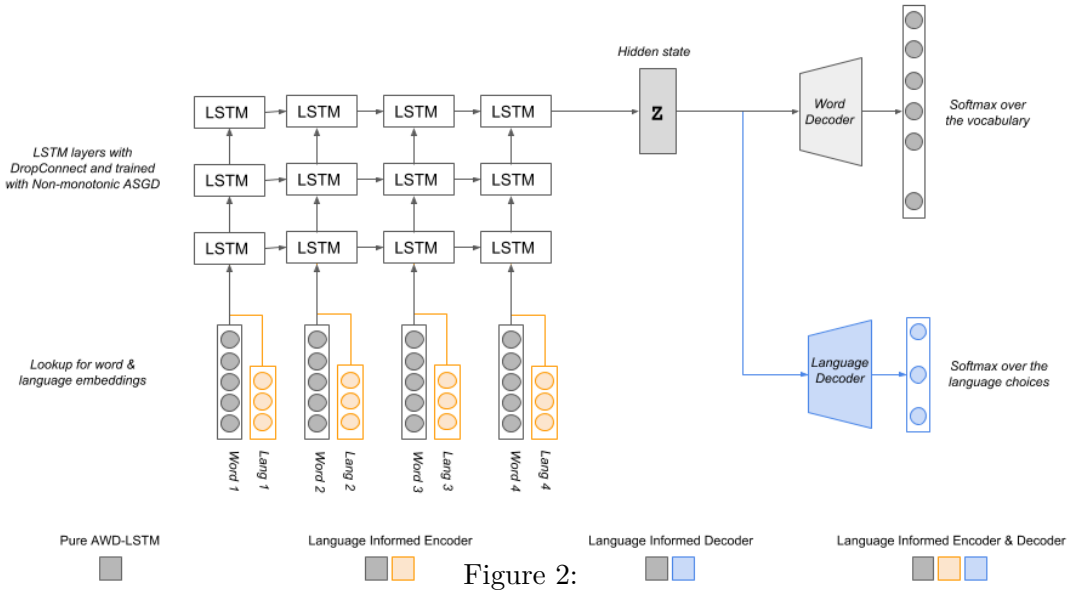

Figure 2: Various CS LM models that we explored in this work

4 Models and Experiments

In this section, we describe the process of step by step building of our final model to perform language modeling for code-switched text that performs comparatively better than the remaining models. The path to this is roughly along the lines of incorporating the language information in the model along with the word information to improve the performance overall.

- With this intuition, we start with a factored LM that takes both language and words as factors to an LSTM based architecture of a language model.
- We then move onto improving the core of the LSTM by adapting the state-of-the-art model (Merity et al., 2018).
- We improve this model by incorporating lexical level language information before encoding the input and decoding the output for the next word. This is similar to an MTL setting where the two tasks are the predicting the next word and predicting the language of the next word. Both the losses back propagate. Note that only the loss at the word decoder is used to calculate perplexity of the model. We also perform ablations with the combinations of giving language information of the current word and predicting the language of the next word.

4.1 Stage 1: Factored Language Model

Each word in the vocabulary V is represented as embeddings of 3 factors: token, soundex encoding, language information. We build an LSTM based LM with long and short term dependencies between switch points and partially normalized tokens.

The model trained uses 3 embedding layers: one learned from pure vocabulary (dimension of 128), one learned from the language label (dimension of 2) and one learned from the soundex labels (dimension of 64) at training time these embeddings are concatenated and fed into an single layered LSTM. A linear decoding layer is then used to transition from the LSTM's representation into the vocabulary space (Figure 1). The model is trained using the Categorical Cross Entropy Loss and the ADAM optimizer (Kingma and Ba, 2014).

4.2 Stage 2: Improving the core of LSTM architecture

In Stage 1, the model has a single layer LSTM where the input representation from the concatenation of individual factors are fed as input. In the second stage, we improve the inner LSTM model itself based on

SOTA for WikiText-2 (Merity et al., 2018). We have built a word level language model based on (Merity et al., 2017). Although the huge number of parameters in neural models enable them to learn high degree of non-linearities present in the data, they affect the generalization capabilities of the model. In spite of techniques such as batch normalization and dropout that are generally used to regularize training in neural networks, the improvement in performance has not been substantial in the case of sequential units such as recurrent models. The nuances that this model addresses are as follows:
First, coming to the problems of a default usage of dropout to the recurrent units that inhibits maintenance of longer term dependencies. One solution for this is to use the same dropout mask at each time step, known as Locked Dropout. Another solution is the application of dropout for specific network units such as certain gates or states rather than the entire unit.

Second kind of regularization techniques involve different kinds of normalization that directly impacts the training process due to the induction of more trainable parameters.

Finally an effective optimization algorithm is needed in synergy with when the dropout is applied. This paper uses ASGD (Averaged Stochastic Gradient Descent) that returns an average across the iterations that have crossed a specified loss threshold value, which is decided using a non-monotonic criterion.

The distinguishing factors with this model are the regularization and optimization strategies that are applied on the LSTM based models built upon their core form.

4.3 Stage 3: Language Informed Reading and Decoding

This is built upon the idea of Stage 1, where we factorize the language id of the input word and decode the next word. This technique is incorporated with the model described in Stage 2 with AWD-LSTM. In addition to this, we pose this as a multi-task learning problem, with shared layers except for the decoder. The 2 tasks are the following:

1. Predict the next word.
2. Predict the language of the next word.

There are a number of ways to frame the desire for humans to switch between languages (Skiba, 1997; Moreno et al., 2002), however, we view the human desire as out of scope for this work. Instead, our focus is on how we can incorporate linguistic information while training a statistical model for code-switched text. We discuss two main choices as to where we can introduce this information: either at the input stage or at the decoding stage of an RNN language model.

Given a CS sentence $\boldsymbol{X_{cs}} = (\boldsymbol{x^1}, \boldsymbol{x^2}$ ... , $\boldsymbol{x^n})$ which has lexical level language sequence $\boldsymbol{L_{cs}} = (\boldsymbol{l^1}, \boldsymbol{l^2}$ ... , $\boldsymbol{l^n})$, our model has to predict the word at the next time step. Note that this vector $\boldsymbol{l^i}$ is the language of the $\boldsymbol{ith}$ lexical item and is represented as a vector of length sixteen which is trained in concert with the model. This allows our model to encode the distributional properties of the language switching. We experimented with encoding and decoding the word and language embeddings for this task. $\boldsymbol{\theta_{E_X}}$, $\boldsymbol{\theta_{E_L}}$, $\boldsymbol{\theta_{D_X}}$ and $\boldsymbol{\theta_{D_L}}$ are the parameters for the word encoder, language encoder, word decoder and language decoder respectively.

We identify four different model architectures (Figure 2) that could be useful in training code switched language models. In the first model, our baseline, we have a sequence of words and we are trying to predict the following word. This model is identical to running a traditional RNN language model on CS text.

For our baseline model we adapt the state-of-the-art language model, the AWD-LSTM, for this domain. This model is a 3 layered stacked LSTM trained via Averaged SGD with tied weights between the embedding and the softmax layer. There are several other important elements of this model, all of which are detailed in (Merity et al., 2017). The next word in this model is given by:

$$\boldsymbol{z} = \textit{Encoder}(\boldsymbol{X_{cs}}, \boldsymbol{\theta_E})$$

In our second model we extend our baseline such that we have a sequence of words

Model/Data	Train	Dev	Test
Base AWD-LSTM Model	10.08	19.73	20.92
Language Aware Encoder AWD-LSTM	10.07	19.00	20.18
Language Aware Decoder AWD-LSTM	11.60	20.72	22.01
Language Aware Encoder & Decoder AWD-LSTM	9.47	18.51	19.52

Table 2: Perplexity scores of different models

and their language IDs and we are trying to predict the following word. This model can be seen as a factored language model operating with code switched data. So, the next word in this model is given by:

$$Decoder(Encoder(\boldsymbol{X_{cs}}, \boldsymbol{\theta_{E_X}}), \boldsymbol{\theta_{D_X}})$$

In our third model we take a sequence of words as an input and attempt to predict both the language and the value of the following word. The next word in this model is given by:

$$Decoder(Encoder(\boldsymbol{X_{cs}}, \boldsymbol{\theta_{E_X}}) \bigoplus Encoder(\boldsymbol{L_{cs}}, \boldsymbol{\theta_{E_X}})), \boldsymbol{\theta_{D_X}})$$

In our fourth model we take a sequence of words and their corresponding language IDs as input and attempt to predict both the language and value of the subsequent word. In our third and fourth models we operate with two loss values being calculated for (one for the word error, and one for the language error multiplied by 0.1) and gradients for both losses are propagated through the network and are used to update the weights.

5 Results and Error Analysis

We trained 4 different models based on the description in Section 4. The results of these experiments are presented in Table 2. We observe that the Language Aware Encoding and Decoding with the AWD-LSTM gives the least perplexity. This aligns with our hypothesis that providing language information of the current word before encoding and enabling the model to decode the language of the next word allows the model to learn a higher level context of switch points between the languages.

5.1 Qualitative Analysis

We also did a qualitative analysis of these LMs with the sentences they generate, some generation examples of the baseline AWD-LSTM are shown in table 3. We observe them to be quite consistent in generating meaningful code-mixed n-grams of length 10 and above.

However, the sentences that got generated were not very coherent. Part of the reason for this is training sentences themselves did not have proper punctuation and end-markers, which are issues to take care of during pre-processing. Notice that the in the first two examples, the first word is connective (in specific a conjunction), and hence expects their respective sub-ordinate and co-ordinate parts of the sentences that are missing.

Also, as seen in the second word of the last sentence, which is the transliterated form of the word for *'mei'* in Hindi (which is the Hindi word for 'in'). In this context, the word that is closest in lexical form should have been *'main'* (which is the Hindi word for 'I'). Using transliteration as a proxy to standardize the representations of the words is the reason for the model to make such errors. Character level models have been working well for spell corrections and normalization and incorporating character level convolution as input along with the word embeddings could potentially help solve this problem.

In this perspective, we have attempted character level CNN based model along with an LSTM at the top but this did not perform very well. Further experimentation of this model with MTL set up by decoding the language is work in progress. Let $\boldsymbol{c_1}$, $\boldsymbol{c_2}$ ... $\boldsymbol{c_p}$ be the padded sequence of characters of a word. Each of the characters has a 50 dimensional embedding (let it be $\boldsymbol{e}$). Let $\boldsymbol{c_{i:i+j}}$ be

अगर आप android lollipop और android phone से connected हैं *gloss*: if you android lollipop and android phone to connected are *trans*: if you are connected to android lollipop and android phone
और आप us android phone के बारे मे share कर रहे हैं *gloss*: and you that android phone regarding share doing are *trans*: and you are sharing regarding that android phone
आज मे अपने blog के लिए बहुत ही फायडिमैंड option लेकर आया hun जिस पर सभी high quality backlink का post लिखी *gloss*: today I my blog for very beneficial option bring here on which all high quality backlink 's post written *trans*: today I bring a very beneficial option for my blog on which all high quality backlink's post written

Table 3: Sample Code-Mixed generation by AWD-LSTM along with gloss and translation (text in red are Hindi words that weren't identified correctly in the LID step.)

the concatenation of characters from $\boldsymbol{i}$ to $\boldsymbol{j}$. Next we take 200 filters of size $\boldsymbol{sXe}$, where s is the window size which in our case is 3. The feature in the CNN layer is computed and activated using a Relu which gives a feature map of size $\boldsymbol{p - s + 1}$ over which mean pooling is performed. After the end of 3 convolution layers, each word is represented in a 200 dimensional hidden convoluted character space which is concatenated with the corresponding 200 dimensional word embedding. This latent representation is used to decode the next word.

In another qualitative analysis we observe the t-sne plots of the Hindi and English words from the embedding layer separately. For the English terms, the content words like `infographics`, `click`, `cyber`, `blueborne` are related to blogs and tend to be grouped closer. While in case of the Hindi terms, verbs like *banaya*(-made),*aaoge*(-come), *karte hain*(are doing) and function words are grouped closer. This is understandable because of the matrix language being in Hindi. So, the Hindi words made up bulk of the syntax contained in these sentences with English content words sprinkled in between. Also, because the English content words are separated by a long distance, they do not influence the meanings of one another in the context. For the same reason we did not find any noticeable correlation between the meanings of Hindi and English word embeddings because they played very different roles in sentences.

5.2 Challenges: Discussion

Robustness of the language model also depends on the diversity of context in which the words co-occur. Since most of the articles belong to the topics of e-commerce, latest technology and health, this may be affected. Hence, we plan to use pre-trained word embeddings based on large monolingual corpora after aligning the embedding spaces of both the participating languages such as MUSE embeddings (Conneau et al., 2017). However, due to the non-standardized spellings in the *romanized* Hinglish text, most words that are incorrectly transliterated are not found in the MUSE embeddings. This implies that the errors from transliteration are propagated through the subsequent parts of model. To avoid this, we plan to extend this work by using character encodings in future.

6 Future Work

There are a number of different avenues that could be explored to improve this work. While we aimed primarily to incorporate language information along with the word input in this work, further work could be a done to incorporate other factors such as parts of speech, & root words etc,. Additionally, our baseline model achieved its high results as a result of significant hyperparameter tuning to the Wikitext2 dataset. This tuning was not performed on this data and could represent a significant avenue for improvement. Lastly, and arguably most importantly, the collection of additional CS data would be a significant contribution to this work. Much of the work involved in this project was to properly clean, parse, and represent the CS data that was scraped from the online sources discussed above. These sources remain limited in topic and variation and additional sources of CS data would be the best way to improve how well our model

(a) English Word Embeddings

(b) *Romanized* Hindi Embeddings

Figure 3: English and Hindi Embeddings

can generalize. We would also explore the effect of including sentence level features composed of corresponding code-switching metrics at the sentence level which conditions the model to generate CS text.

Additionally, the probabilities given by the language decoder could be tuned towards using as bias to the word decoder model to predict the next word. These probabilities can be multiplied with the probabilities of the vocab which can be used as bias term for predicting the word. This cross product as bias term reduces the probabilities of the Hindi word if the language probability of English is higher and vice versa.

7 Conclusion

We hypothesize that incorporating the information of language aids in building more robust language models for code-switched text. In this context, we substantiate our hypothesis by experimenting with different combinations of providing the language of the current word as input and decoding the language of the next word along with th word itself. We conclude that we are able to improve the State-of-The-Art language model for monolingual text by both explicitly providing the language information and decoding the language of the next word to perform this task for CS domain. We treat this problem as a multi-task learning problem where the same embedding and LSTM layers are shared. These two comparable tasks are predicting the next word and predicting the language of the next word. So far, our best test perplexity is 18.51 on development and 19.52 on test sets. This is in comparison to the baseline model which is 19.73 and 20.92 on dev and tests respectively. We are of the opinion that further research can be done to not only improve perplexity, but to also improve the quality of the training and testing dataset. Language models are a core element in multiple generation tasks, from speech recognition to machine translation, we believe that this work will support future research into the development of such NLP tools for CS domain.

References

Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz. 2015. Syntactic and semantic features for code-switching factored language models. *IEEE Transactions on Audio, Speech, and Language Processing*, 23(3):431–440.

Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz. 2013a. Recurrent neural network language modeling for code switching conversational speech. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8411–8415. IEEE.

Heike Adel, Ngoc Thang Vu, and Tanja Schultz. 2013b. Combination of recurrent neural networks and factored language models for code-switching language modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 206–211.

Barbara E Bullock and Almeida Jacqueline Ed Toribio. 2009. *The Cambridge handbook of linguistic code-switching*. Cambridge University Press.

Özlem Çetinoğlu and Çağrı Çöltekin. 2016. Part of speech annotation of a turkish-german

code-switching corpus. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 120–130.

Özlem Çetinoğlu, Sarah Schulz, and Ngoc Thang Vu. 2016. Challenges of computational processing of code-switching. *arXiv preprint arXiv:1610.02213*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Amitava Das and Björn Gambäck. 2014. Identifying languages at the word level in code-mixed indian social media text.

Saurabh Garg, Tanmay Parekh, and Preethi Jyothi. 2017. Dual language models for code mixed speech recognition. *arXiv preprint arXiv:1711.01048*.

P Goyal, Manav R Mital, A Mukerjee, Achla M Raina, D Sharma, P Shukla, and K Vikram. 2003. A bilingual parser for hindi, english and code-switching structures. In *10th Conference of The European Chapter*, page 15.

Gualberto Guzmán, Joseph Ricard, Jacqueline Serigos, Barbara E Bullock, and Almeida Jacqueline Toribio. 2017. Metrics for modeling code-switching across corpora. *Proc. Interspeech 2017*, pages 67–71.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 239–248.

Aravind K Joshi. 1982. Processing of sentences with intra-sentential code-switching. In *Proceedings of the 9th conference on Computational linguistics-Volume 1*, pages 145–150. Academia Praha.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Ben King and Steven Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1110–1119.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ying Li and Pascale Fung. 2013. Improved mixed language speech recognition using asymmetric acoustic model and language model with code-switch inversion constraints. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7368–7372. IEEE.

Ying Li and Pascale Fung. 2014. Code switch language modeling with functional head constraint. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 4913–4917. IEEE.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *arXiv preprint arXiv:1803.08240*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Eva M Moreno, Kara D Federmeier, and Marta Kutas. 2002. Switching languages, switching palabras (words): An electrophysiological study of code switching. *Brain and language*, 80(2):188–207.

Pieter Muysken. 2000. *Bilingual speech: A typology of code-mixing*, volume 11. Cambridge University Press.

Carol Myers-Scotton. 1997. *Duelling languages: Grammatical structure in codeswitching*. Oxford University Press.

Jasabanta Patro, Bidisha Samanta, Saurabh Singh, Abhipsa Basu, Prithwish Mukherjee, Monojit Choudhury, and Animesh Mukherjee.

2017. All that is english may be hindi: Enhancing language identification through automatic ranking of the likeliness of word borrowing in social media. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2264–2274.

Shana Poplack. 1980. Sometimes i' ll start a sentence in spanish y termino en espanol: toward a typology of code-switching1. *Linguistics*, 18(7-8):581–618.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do hindi-english speakers do on twitter? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1131–1141.

David Sankoff and Shana Poplack. 1981. A formal grammar for code-switching. *Research on Language & Social Interaction*, 14(1):3–45.

R Mahesh K Sinha and Anil Thakur. 2005. Machine translation of bi-lingual hindi-english (hinglish) text. *10th Machine Translation summit (MT Summit X), Phuket, Thailand*, pages 149–156.

Richard Skiba. 1997. Code switching as a countenance of language interference. *The internet TESL journal*, 3(10):1–6.

Thamar Solorio and Yang Liu. 2008a. Learning to predict code-switching points. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 973–981. Association for Computational Linguistics.

Thamar Solorio and Yang Liu. 2008b. Part-of-speech tagging for english-spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1051–1060. Association for Computational Linguistics.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.