Language Identification and Named Entity Recognition in Hinglish Code Mixed Tweets

Kushagra Singh
IIIT Delhi

Indira Sen
IIIT Delhi

Ponnurangam Kumaraguru
IIIT Delhi

{kushagra14056,indira15021,pk}@iiitd.ac.in

Abstract

While growing code-mixed content on Online Social Networks (OSNs) provides a fertile ground for studying various aspects of code-mixing, the lack of automated text analysis tools render such studies challenging. To meet this challenge, a family of tools for analyzing code-mixed data such as language identifiers, parts-of-speech (POS) taggers, chunkers have been developed. Named Entity Recognition (NER) is an important text analysis task which is not only informative by itself, but is also needed for downstream NLP tasks such as semantic role labeling. In this work, we present an exploration of automatic NER of code-mixed data. We compare our method with existing off-the-shelf NER tools for social media content, and find that our systems outperforms the best baseline by 33.18 % (F_1 score).

1 Introduction

Code-switching or code-mixing occurs when "lexical items and grammatical features from two languages appear in one sentence" (Muysken, 2000). [1] It is frequently seen in multilingual communities and is of interest to linguists due to its complex relationship with societal factors (Sridhar and Sridhar, 1980).

With the rise of Web 2.0, the volume of text on online social networks (OSN) such as Twitter, Facebook, Reddit has grown. It is estimated that around 240 Million Indian users, alone, are active on Twitter [2]. A significant fraction of these users are bilingual, or even trilingual, and their tweets can be monolingual in English or their vernacular, or code-mixed. Past research has looked at multiple dimensions of this behaviour such as its relationship to emotion expression (Rudra et al., 2016) and identity. Code-mixing or multilingualism of tweets poses a significant problem to both the OSNs' underlying text mining algorithms as well as researchers trying to study online discourse, since most existing tools for analyzing OSN text content caters to monolingual data. For example, Twitter's abuse detection systems fail to flag code-mixed tweets as offensive. [3] Recent efforts to build tools for code-mixed content include language identifiers (Solorio and Liu, 2008), parts-of-speech(POS) taggers (Vyas et al., 2014), and chunking (Sharma et al., 2016). A natural extension of these set of automated natural language processing (NLP) tools is a Named Entity Recognizer (NER) for code-mixed social media data, which we present in this paper. Additionally, as language tags are an essential feature for NLP tasks, including NER, we also present a neural network based language identifier.

Our main contributions are:

1. Building a token-level language identification system for Hindi-English (Hi-En) code mixed tweets, described in detail in Section 3.

2. Building an NER for En-Hi code-mixed tweets, which we explain in Section 4. We also show, in Section 5, that our NER performs better than existing baselines.

2 Related Work

In this section we briefly describe the approaches for automatic language identification and extrac-

[1] Many researchers use code-mixing and code-switching interchangeably, which we follow in this work

[2] https://www.statista.com/statistics/381832/twitter-users-india/

[3] http://timesofindia.indiatimes.com/india/to-avoid-social-media-police-indian-trolls-go-vernacular/articleshow/60139671.cms

tion of named entities.

Language Identification for code-mixed content has been previously explored in Barman et al. (2014). Particularly close to our work is the use of deep-learning approaches for detecting token-level language tags for code-mixed content (Jaech et al., 2016).

We particularly focus on efforts to building NERs for social media content and, NERs for Indian languages and code-mixed corpora. Social Media text, including and especially tweets, have subtle variations from written and spoken text. These include slacker grammatical structure, spelling variations, ad-hoc abbreviations and more. See Ritter et al. (2011) for detailed differences between tweets and traditional textual sources. Monolingual NER for tweets include (Ritter et al., 2011; Li et al., 2012). We build on these approaches to account for code-mixing and use the former as a baseline to test our method against.

3 Language Identification using Transliteration

We build a token level language identification model (LIDF) for code-mixed tweets using monolingual corpora available for both the languages, supplemented by a small set of annotated code-mixed tweets. We hypothesize that words or character sequences of different languages encode different structures. Therefore, we aim to capture this character structure for both languages. Subsequently, given a token, we try to see which language fits better. Our LIDF algorithm comprises of multiple steps, described in Figure 1. Each of the steps have been explained in detail below.

3.1 Roman-Devanagari transliteration

We restrict ourselves to instances of En-Hi code-mixing where the Hindi component is written in Roman script. [4] Therefore, any model trained on Hindi corpora (which will be in Devanagari) is not directly usable. To make use of such a corpus, we first transliterate words written in Roman to Devanagari script.

We train a model T, that takes a token written in Roman characters and generates its Devanagari equivalent (given token ***school***, T generates स्कूल). We follow an approach used by (Rosca and Breuel, 2016) for English-Arabic transliteration and, and train an Attention Sequence to Sequence (Seq2Seq) model (Bahdanau et al., 2014). Given an input sequence of Roman characters, T generates a sequence of Devanagari characters.

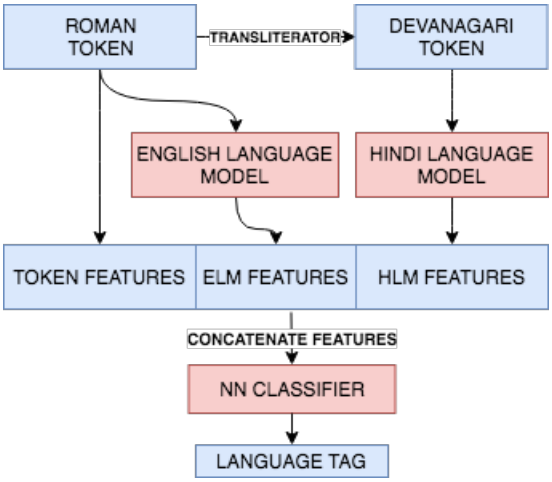

Figure 1: Different steps of our language identification pipeline. We extract features using language models trained on monolingual corpora, and train a classifier based on these features.

We train T using Roman-Devanagari transliteration pairs mined by (Gupta et al., 2012) and (Khapra et al., 2014). After combining the two sets and removing duplicates, we were left with 41,383 unique pairs.

We explore multiple Seq2Seq models, experimenting with different RNN cells, encoder and decoder depths, output activation functions and the number of RNN cells in each layer. We evaluate their performance using the character error rate metric, comparing with LITCM (Bhat et al., 2015) as a baseline. We report the performance of our top five models in Table 1, all of which perform better than the baseline.

Our best model comprises of a 2 layer bidirectional RNN encoder and a 2 layer RNN decoder, each layer comprising of 128 GRU units with ReLU activation. We use the Adagrad optimizer to train T, adding a dropout of 0.5 after each layer and use the early stopping technique to prevent over fitting. Larger and deeper networks perform relatively poorer since they start overfitting quickly.

We observe that Hindi words which have multiple spellings when written in Roman script are all transliterated to the same Hindi token. Therefore, T could also be used for normalization and

[4] This is followed in previous studies since the quantity of code-mixed content with non-Roman Hindi is negligible.

Model	Depth	# Units	Cell	CER
LITCM	–	–	–	18.88
Seq2Seq	3	256	LSTM	17.23
Seq2Seq	2	64	GRU	17.09
Seq2Seq	3	128	GRU	16.99
Seq2Seq	2	128	LSTM	16.67
Seq2Seq	**2**	**128**	**GRU**	**16.54**

Table 1: Performance of different transliteration models. Reported CER is in percentage and was calculated after a 5-fold cross validation. Depth was kept the same for both encoding and decoding layers.

we hope to investigate this in more detail in future.

3.2 Extracting features using monolingual corpora

For both languages, we learn the structure of the language at a character level. We do this by training a model which for a token of length n, learns to predict the last character given the first $n-1$ characters as input. More formally, for each token $\{c_1, c_2, ..., c_n\}$, the model learns to predict c_n given the input sequence $\{c_1, c_2, ..., c_{n-1}\}$. We model this as a sequence classification problem using LSTM RNNs (Hochreiter and Schmidhuber, 1997). We use ELM to refer to the English language model, and HLM to refer to the Hindi language model.

The same architecture is used for both ELM and HLM, as shown in Figure 1. Both comprise of two RNN layers with 128 LSTM cells each, using ReLU activation. The output of the second RNN layer at the last time step is connected to a fully connected (FC) layer with softmax activation. The size of this FC layer is equal to the character vocabulary V of the language. We take a softmax over the FC layer to predict c_n. The normalized outputs from the FC layer can be thought of as a probability distribution over V, the i^{th} normalized output being equal to $P(V_i|c_1, c_2, ..., c_{n-1})$, where V_i is the i^{th} character in the vocabulary. We refer to the normalized outputs from the FC layer of ELM and HLM as P_{ELM} and P_{HLM} respectively, and use them as features for our language detection classifier.

For training ELM, we use the News Crawl dataset provided as a part of the WMT 2014 translation task.[5] As a preprocessing step, we remove all non-alphabetic characters (such as numbers and punctuations), and convert all uppercase alphabets to lowercase. After preprocessing, we were left with a total of 98,565,179 tokens, 189,267 of which were unique. For HLM, we use the IIT Bombay Hindi corpus (Kunchukuttan et al., 2017). We follow the same preprocessing steps, except for converting to lowercase (since there is no concept of case in Hindi). This yielded 59,494,325 tokens, of which 161,020 were unique.

The input sequence is encoded into a sequence of one hot vectors before feeding it to the network. We use categorical cross-entropy as the loss function, optimizing the model using gradient descent (Adagrad). 20% of the unique tokens are held out and used to validate the performance of our model. Once again, we use the early stopping technique and add a dropout of 0.5 after each layer to avoid over-fitting.

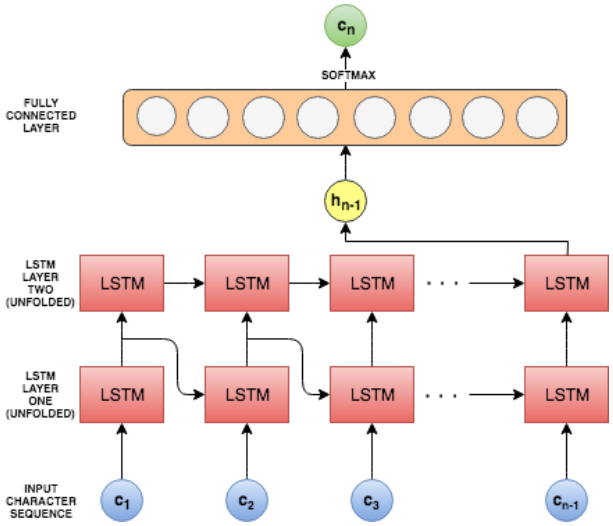

Figure 2: Architecture for ELM and HLM. The figure shows one cell in each LSTM layer unrolled over time.

3.3 Predicting language tag using P_{ELM} and P_{HLM}

Given a word $\{c_1, c_2, ..., c_n\}$ we first transliterate it to Devanagari using T, generating $\{c'_1, c'_2, ..., c'_k\}$. Then by passing $\{c_1, c_2, ..., c_{n-1}\}$ and $\{c'_1, c'_2, ..., c'_{k-1}\}$ through ELM and HLM respectively, we obtain P_{ELM} and P_{HLM}. Our hypothesis is that we can differentiate P_{ELM} of a Hindi word from the P_{ELM} of an English word,

[5] http://statmt.org/wmt14/translation-task.html

since the character sequence structure of a Hindi word is different from that of English words (which are used to train ELM). Similarly, we can differentiate P_{HLM} of an English word from the P_{HLM} of a Hindi word.

We use a set of tweets curated by Sakshi Gupta and Radhika (2016) which are annotated for language at a token level (each token is either English, Hindi or Rest) to train a three class classifier using (i) P_{ELM} (ii) P_{HLM}, (iii) ratio of non-alphabetic characters in W, (iv) ratio of capitalization in W and (v) Binary feature indicating whether W is title-case as features. The last three features help identify the *Rest* tokens. Our 3-class classifier is a fully connected neural network with 2 hidden layers using ReLU activation. On 5-fold cross validation, our model achieves an average F_1 score of 0.934 and an average accuracy of 0.961 across the three classes. This is a slight improvement over the model proposed by Sharma et al. (2016), which had an accuracy of 0.938 on the same dataset (as reported by reported by Sakshi Gupta and Radhika (2016)).

4 Named Entity Recognition

Named entity recognition typically comprises of two components, (i) entity segmentation and (ii) entity classification. Both these components can either be modeled separately as done by Ritter et al. (2011), or they can be combined and tackled together like the model proposed by Finkel et al. (2005). We adopt the latter approach, modeling both components together as a sequence labeling task. Our hypothesis is that named entities can be identified using features extracted from words surrounding it. We explore models using Conditional Random Fields and LSTM RNNs using handcrafted features described below.

4.1 Features

Our hand-crafted features are described below.

- ***Token based features*** : The current token T, T after stripping all characters which are not in the Roman alphabet (T_{clean}), and converting all characters in T_{clean} to lowercase (T_{norm}) generates three different features. We create $T_{wordhsape}$ by replacing all uppercase letters in T with X, all lowercase letters with x, all numerals with o and leave all other characters as they are. For example, **Adam123** becomes **Xxxxooo**. We also use token length T_L as as feature.

- ***Affixes*** : Prefixes and suffixes of length 1 to 5 extracted from T, padded with whitespace if needed. These help in identifying phrases that are not entities. For example, an English token ending in ***ing*** is highly unlikely to be a named entity.

- ***Character based features*** : Binary features indicating (i) whether T is title case, (ii) whether T has an uppercase letter, (iii) whether T has all uppercase characters, (iv) whether T has a non-alphanumeric character and (v) whether T is a hashtag. We also calculate the fraction of characters in T which are ASCII

- ***Language based features*** : The language which T belongs to, as predicted by the model proposed in section 3. For the LSTM model, we also use P_{ELM} and P_{HLM} generated for T.

- ***Syntactic features*** : POS tag for T as predicted by the model trained by Owoputi et al. (2013), POS tag and chunk tag for T and as predicted by the shallow parser trained by Sharma et al. (2016)

- ***Tweet capitalization features*** : From the tweet that T belongs to, we extract (i) fraction of characters that are uppercase, (ii) fraction of characters that are lowercase, (iii) fraction of tokens that are title case.[6]

4.2 Proposed Models

Our LSTM model (Figure 3) comprises of two bidirectional RNN layers using LSTM cells and ReLU activation. The input at the time step t is F_t, i.e. the feature vector for the token at position t in the tweet. F_t is generated by concatenating the extracted features for the token at position t. T, T_{clean}, T_{norm}, $T_{wordhsape}$ and affixes are passed through embedding layers which are randomly initialized and learnt during the training phase. All real-valued features are encoded into one-hot vectors of length 10, using percentile binning.

The output (h_t) of the second RNN layer at each time step is passed to a separate fully connected

[6]Using the output of the T_cap classifier trained by Ritter et al reduces accuracy.

Type	Lample	Ritter	LSTM	CRF	N
PER	38.73	38.45	65.47	**72.23**	1644
LOC	38.35	46.84	67.53	**72.50**	744
ORG	16.99	13.54	50.94	**70.35**	375
All	33.77	36.88	64.64	**72.06**	2763

Table 2: Performance (F_1 scores) of different models on segmentation and classification. N is the total number of entities in the entire dataset. Reported numbers are in percentages. Results of CRF and LSTM are on 5-fold cross validation.

layer FC_t. We take a softmax over the output of FC_t to predict the label (L_t) for the token at position t. While training, we use the Adagrad optimizer and add a dropout of 0.5 after each layer.

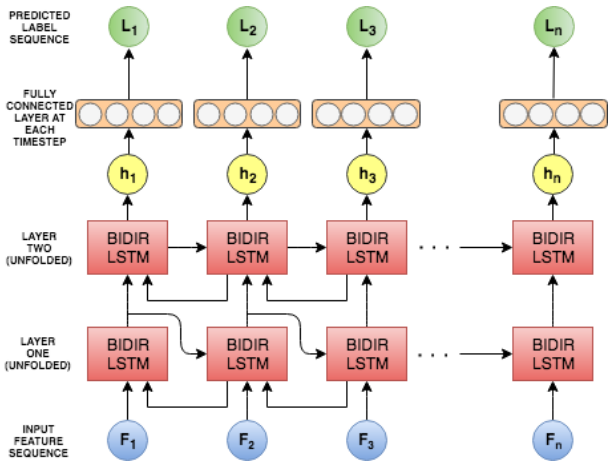

Figure 3: LSTM NER architecture. The figure shows one cell in each LSTM layer unrolled over time

Our CRF model is the standard generalized CRF proposed by Lafferty et al. (2001), which allows a flow of information across the sequence in both directions. We add L_1 and L_2 regularization to prevent over-fitting, and do an extensive grid search to come up with optimum values for these constants.

5 Experiments and Results

5.1 Data collection and annotation

We randomly sampled 50,000 tweets from the code mixed tweet dataset collected by (Patro et al., 2017). On this set, we ran our language detection algorithm and filtered tweets which had at least five Hi tokens. The filtered data also had tweets containing Roman tokens belonging to languages other than English and Hindi (like transliterated Telugu), such tweets were removed during the annotation process.

The final dataset comprised of 2079 tweets (35,374 tokens). 13,860 (39.18 %) of the tokens were En, 11,391 (32.2 %) Hi and 10,123 (28.61 %) $Rest$. Each tweet was annotated at a token level for three classical named entity types $Person$, $Location$ and $Organisation$, using the IOB format. The annotation process was carried out by three linguists proficient in both English and Hindi. The final label was decided based on majority vote and any instance (around 2%) where all three annotators disagreed was resolved through discussion. In all, the annotators identified 2763 entity phrases (3751 tokens) which included 1,644 $Person$ entities, 744 $Location$ entities and 375 $Organisation$ entities.

5.2 Baselines and Results

We compare our proposed models (LSTM and CRF) with two baseline systems, (i) a state of the art English NER model proposed by Lample et al. (2016) and (ii) a state of the art NER model for tweets proposed by Ritter et al. (2011). Named Entities do not have belong to one particular language though the same NE might have different forms in different languages, such as Cologne in English, is Kln in German. For the purpose of this study, we do not assign NEs any language tags and leave the detection and mapping of multiple NE forms as future work.

The results are summarized in Table 2 (segmentation) and Table 3 (segmentation and classification). All metrics are calculated on a phrase level, no partial credit is awarded. An incorrectly identified boundary is penalized as both, a false positive and a false negative. For computing Table 2, we generalize entity tags (B-PER, B-LOC, B-ORG become B-ENT, I-PER, I-LOC, I-ORG become I-ENT). As expected, both these systems fare poorly on our data at both entity segmentation and entity classification. We believe this is due to the high number of out of vocabulary tokens (belonging to Hindi) in the data.

6 Discussion

In this paper, we present a Named Entity Recognition tool specifically targeting Hindi-English code-mixed content. To build our NER model, we

Model	Precision	Recall	F_1 score
Lample	36.55	46.59	40.97
Ritter	64.64	34.24	44.77
LSTM	74.45	64.87	69.33
CRF	**84.95**	**69.91**	**76.70**

Table 3: Performance of different models at entity segmentation. All numbers are in percentages.

also present a unique semi-supervised language identifier which exploits the character-level differences in languages. We validate the performance of our NER against off-the-shelf NER for Twitter and observe that our model outperforms them.

In future, we plan to explore building other downstream NLP tools such as Semantic Role Labeling or Entity-specific Sentiment Analyzers which make use of NER for code-mixed data.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the first workshop on computational approaches to code switching*. pages 13–23.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2015. Iiit-h system submission for fire2014 shared task on transliterated search. In *Proceedings of the Forum for Information Retrieval Evaluation*. ACM, New York, NY, USA, FIRE '14, pages 48–53. https://doi.org/10.1145/2824864.2824872.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '05, pages 363–370. https://doi.org/10.3115/1219840.1219885.

Kanika Gupta, Monojit Choudhury, and Kalika Bali. 2012. Mining hindi-english transliteration pairs from online hindi lyrics. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2012)*. pages 2459–2465.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Aaron Jaech, George Mulcaire, Mari Ostendorf, and Noah A Smith. 2016. A neural model for language identification in code-switched tweets. In *Proceedings of The Second Workshop on Computational Approaches to Code Switching*. pages 60–64.

Mitesh M. Khapra, Ananthakrishnan Ramanathan, Anoop Kunchukuttan, Karthik Visweswariah, and Pushpak Bhattacharyya. 2014. When transliteration met crowdsourcing : An empirical study of transliteration via crowdsourcing using efficient, non-redundant and fair quality control. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA), Reykjavik, Iceland.

Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. 2017. The IIT bombay english-hindi parallel corpus. *CoRR* abs/1710.02855. http://arxiv.org/abs/1710.02855.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '01, pages 282–289. http://dl.acm.org/citation.cfm?id=645530.655813.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *CoRR* abs/1603.01360. http://arxiv.org/abs/1603.01360.

Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun, and Bu-Sung Lee. 2012. Twiner: named entity recognition in targeted twitter stream. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 721–730.

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

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

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 likeliness of word borrowing in social media. *CoRR* abs/1707.08446. http://arxiv.org/abs/1707.08446.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1524–1534.

Mihaela Rosca and Thomas Breuel. 2016. Sequence-to-sequence neural network models for transliteration. *CoRR* abs/1610.09565. http://arxiv.org/abs/1610.09565.

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 *EMNLP*. pages 1131–1141.

Piyush Bansal Sakshi Gupta and Mamidi Radhika. 2016. Resource creation for hindi-english code mixed social media text .

Arnav Sharma, Sakshi Gupta, Raveesh Motlani, Piyush Bansal, Manish Shrivastava, Radhika Mamidi, and Dipti M Sharma. 2016. Shallow parsing pipeline for hindi-english code-mixed social media text. In *Proceedings of NAACL-HLT*. pages 1340–1345.

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

Shikaripur N Sridhar and Kamal K Sridhar. 1980. The syntax and psycholinguistics of bilingual code mixing. *Canadian Journal of Psychology/Revue canadienne de psychologie* 34(4):407.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *EMNLP*. volume 14, pages 974–979.