Part-of-speech Tagging of Code-mixed Social Media Content: Pipeline, Stacking and Joint Modelling

Utsab Barman, Joachim Wagner and Jennifer Foster
ADAPT Centre, National Centre for Language Technology
School of Computing, Dublin City University, Dublin, Ireland
{ubarman,jwagner,jfoster}@computing.dcu.ie

Abstract

Multilingual users of social media sometimes use multiple languages during conversation. Mixing multiple languages in content is known as code-mixing. We annotate a subset of a trilingual code-mixed corpus (Barman et al., 2014) with part-of-speech (POS) tags. We investigate two state-of-the-art POS tagging techniques for code-mixed content and combine the features of the two systems to build a better POS tagger. Furthermore, we investigate the use of a joint model which performs language identification (LID) and part-of-speech (POS) tagging simultaneously.

1 Introduction

Automatic processing of code-mixed social media content is an emerging topic in NLP (Solorio et al., 2014; Choudhury et al., 2014). Code-mixing is a linguistic phenomenon where language switching occurs at a sentence boundary (inter-sentential), or within a sentence (intra-sentential) or within a word (word-level). This phenomenon can be observed among multilingual speakers and in many languages. Additionally, non-English speakers often use Roman script to write something in social media. This is known as Romanisation. The following comment taken from a Facebook group of Indian students is an example of trilingual code-mixed content:

> Original: *Yaar tu to*, GOD *hain*. **tui** JU **te ki korchis**? Hail u man!
>
> Translation: Buddy you are GOD. What are you doing in JU? Hail u man!

Three languages are present in this comment: English, Hindi (italics) and Bengali (bold). Bengali and Hindi words are written in romanised forms. These phenomena (code-mixing and Romanisation) can occur simultaneously and increase the ambiguity of words. For example, in the previous comment, 'to' could be mistaken as an English word but it is a romanised Hindi word. Moreover, the romanised form of a native word may vary according to the user's preference. In such situations automatic processing is challenging.

POS tagging in code-mixed data (Solorio and Liu, 2008; Vyas et al., 2014) is an interesting problem because of its word-level ambiguity. Traditional NLP systems trained in one language perform poorly on such multilingual code-mixed data. In this paper, we present a data set manually annotated with part of speech and language[1]. We implement and explore two state-of-the-art methods for POS tagging in code-mixed data, i.e. (1) a stacked system (Solorio and Liu, 2008)[2] and (2) a pipeline system (Vyas et al., 2014). To our knowledge, a comparison between these two POS tagging methods for code-mixed content, i.e. (1) and (2), has not been carried out before. In our study we compare these two POS tagging approaches which is an important contribution of this paper.

In romanised and code-mixed text, words of different languages may take the same lexical form. As a result, language and POS ambiguity are in-

[1] This is a subset of the romanised English-Bengali-Hindi code-mixed corpus described by Barman et al. (2014).

[2] In a stacking approach one learner is used to perform a certain task and the output of this learner is used as features for a second learner performing the same task (in our case POS tagging).

Proceedings of the Second Workshop on Computational Approaches to Code Switching, pages 30–39,
Austin, TX, November 1, 2016.

creased. POS labels often depend on the language in code-mixed content. Thus, modelling the interaction between language labels and POS labels may be useful. Furthermore, joint modelling avoids error propagation. We compare our joint model for LID and POS tagging to the stacked model and the pipeline system. We use Factorial Conditional Random Fields (FCRF) (Sutton et al., 2007) as the joint model in our study.

The rest of the paper is organised as follows: in Section 2, we discuss related work. In Section 3 we describe our data for this task. Our experiments are described in Section 4. Section 5 contains analysis of the results. Finally, we conclude and suggest ways to extend this work in Section 6.

2 Related Work

POS tagging with code-mixed social media content is attracting much attention these days (Das, 2016). Different machine learning solutions are being proposed, e.g. Hidden Markov Models (Sarkar, 2015), Conditional Random Fields (Sharma and Motlani, 2015), Decision Trees (Jamatia and Das, 2014; Pimpale and Patel, 2015) and Support Vector Machines (SVM) (Solorio and Liu, 2008). Combining monolingual taggers in a pipeline (Vyas et al., 2014) is also another approach. The POS tagging methods used in these studies can be divided into the following approaches: (i) using a single machine learning classifier (Sarkar, 2015; Sharma and Motlani, 2015; Jamatia and Das, 2014; Pimpale and Patel, 2015), (ii) stacking (Solorio and Liu, 2008) and (iii) pipeline architectures (Vyas et al., 2014).

POS tagging with Spanish-English code-mixed data is first explored by Solorio and Liu (2008). They use two monolingual POS taggers (Spanish and English) to extract the lemma, POS tag and POS confidence scores for each word according to both taggers. First they investigate heuristic methods. These methods are based on handcrafted rules and use the prediction confidence, the predicted tag and the lemma for a particular word from each POS tagger as well as language information of the word generated from a LID system to select the tag from one of the (English or Spanish) POS taggers. Further, they employ an SVM classifier with the extracted information as features and achieve higher accuracy than their heuristic methods.

Vyas et al. (2014) implement a pipeline approach for POS tagging in English-Hindi code-mixed data. They divide the text into contiguous maximal word chunks which are in the same language according to the language identifier. These chunks are further processed through normalisation and transliteration modules. Normalisation is carried out if the chunk is in English, otherwise transliteration is performed to convert the non-English romanised chunk to its Hindi transliterated form. Afterwards, language-specific POS taggers are applied to predict the POS labels of the word chunks. They identify that normalisation and transliteration are two challenging problems in this pipeline approach.

Our inspiration behind the joint modelling of LID and POS tagging comes from the work of Sutton et al. (2007). They use Factorial Conditional Random Fields (FCRF) to jointly model POS tagging and noun-phrase chunking. In their work the FCRF achieves better accuracy than a cascaded CRF approach. FCRF is also found to be useful in joint labelling of sentence boundaries and punctuations (Lu and Ng, 2010).

3 Data

We use a subset[3] of 1,239 code-mixed posts and comments from the English-Bengali-Hindi corpus (a trilingual code-mixed corpus of 12K Facebook posts and comments) of Barman et al. (2014). This corpus contains word-level language annotations. Each word in the corpus is tagged with one of the following labels: (1) English, (2) Hindi, (3) Bengali, (4) Mixed, (5) Universal, (6) Named Entity and (7) Acronym. The label *Universal* is associated with symbols, punctuation, numbers, emoticons and universal expressions (e.g. *hahaha* and *lol*).

We manually annotate POS using the universal POS tag set [4] (Petrov et al., 2012). These annotations were performed by an annotator who is proficient in all three languages of the corpus. As we had no second annotator proficient in all three languages,

[3] We are preparing to release the data set. For more information please contact the first author.

[4] An alternative tag set is the one introduced for code-mixed data by Jamatia and Das (2014). However, we prefer the universal tag set because of its simplicity, its applicability to many languages and its popularity within the NLP community.

we cannot present the inter-annotator agreement for the annotations.

The language and POS label distributions for our data set are shown in Table 1 and 2. In terms of tokens, Bengali (47.9%) is the majority language. 23.2% tokens are English but the amount of Hindi tokens is low, only 6.3%. We analyse the ambiguity of word types in this subset. Our subset contains 7,959 word types, among which only 297 (3.7%) types are ambiguous according to language labels and 569 types (7.1%) are ambiguous according to POS labels.

Label	Count
English	6,383
Bengali	13,171
Hindi	1,746
Universal	5,209
Name Entity	712
Acronym	229
Mixed	69

Table 1: Language label distribution.

Label	Count
NOUN	8,376
PRT	1,332
VERB	4,422
ADV	754
DET	893
ADP	1,358
CONJ	745
ADJ	1,999
PUNCT	4,321
PRON	2,484
NUM	164
X	671

Table 2: POS label distribution.

4 Experiments and Results

We divide the experiments into four parts. We implement baselines for POS tagging in Section 4.1. In Section 4.2 we implement pipeline systems. In Section 4.3 we present our stacking systems and in Section 4.4 we present our joint model.

We perform five fold cross-validation with the data and report average cross-validation accuracy. We investigate the use of handcrafted features and features that can be obtained from monolingual POS taggers (stacking). We perform experiments with different combinations of these feature sets. The following are the features used in our experiments.

1. **Handcrafted Features:** Following Barman et al. (2014), we use prefix and suffix character-n-grams (n = 1 to 5), presence in dictionaries, length of the word, capitalisation information and the previous and the next word as handcrafted features.

2. **Stacking Features:** These features are obtained from the output of a POS tagging system. These features are tokens, predicted labels, and prediction confidence of a POS tagging system.

3. **Combined Features:** This feature set is a union of the previous two feature sets.

Following Barman et al. (2014) we train an LID SVM classifier using handcrafted features. Its predictions are used in the POS tagging experiments below. The LID classifier achieves 91.52% average accuracy in 5-fold cross-validation.

4.1 Baseline

This method only uses the *code-mixed romanised data* and handcrafted features. We try an linear kernel SVM and a linear chain CRF classifier (see Table 3). In terms of average cross-validation accuracy, the SVM classifier (85.00% for $C = 0.00097$) performs better than the CRF classifier (83.89%) in optimised settings.

4.2 Pipeline

Following Vyas et al. (2014), we design a pipeline system. The training data for this method is *monolingual non-romanised*. First, it uses an LID system (trained on romanised data) to identify language-specific chunks. After that it applies monolingual POS taggers to the relevant language chunks to produce the output. The component POS taggers are trained on monolingual non-romanised data.

In this system, code-mixed romanised data passes through a pipeline of LID, transliteration and POS tagging modules. For example, for Bengali-English romanised code-mixed content, the LID module produces Bengali and English chunks, and the Bengali chunks are transliterated into Bengali script and are sent to a Bengali tagger. The English chunks are sent to an English tagger as they are. The final output combines the results from the individual taggers. To implement this method we carry out the following steps:

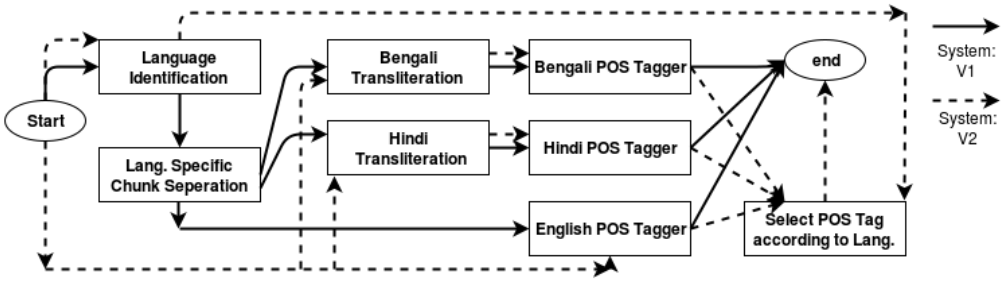

Figure 1: Pipeline systems: system V1 and V2 (Section 4.2).

1. We perform transliteration based on language using Google Transliteration[5] for Hindi and Bengali tokens. (Vyas et al. (2014) use an in-house tool).

2. For the next step of the pipeline, we train monolingual POS taggers for Bengali and Hindi using the SNLTR Bengali and Hindi corpus[6] with TreeTagger[7] (Schmid, 1994). For English we use the default English model which is available with the TreeTagger package[8]. We also use a lightweight Bengali and Hindi stemmer (Ganguly et al., 2012) to provide a stemmed lexicon to TreeTagger during training. We use these taggers to make predictions on English, transliterated Bengali and transliterated Hindi chunks.

The black lines in Figure 1 shows the pipeline of this method (V1). The three training data sets for the three POS taggers follow different tag sets, we map these tags to the universal POS tags after prediction.[9] We achieve 71.12% average cross-validation accuracy with this method (V1) (third row of Table 3).

In method V1, the TreeTagger models are trained on full monolingual sentences. If language-specific text fragments are presented to such monolingual taggers, the taggers may treat these fragments as full sentences. At the start and at the end of the input, the prediction of such taggers may become biased to some specific patterns (e.g. NOUN + PUNCT) that have been observed frequently as a start and an end tag sequence of sentences during training. To avoid this problem we implement a variant (V2) of this system in which we present full sentences (that may contain junk transliteration) to each POS tagger. We perform transliteration as the first component of the system. We present the transliterated content in Bengali script to the Bengali tagger, original romanised content to the English tagger and transliterated content in Hindi script to the Hindi tagger. Finally, we choose from the outputs of these three taggers based on the language prediction by the SVM classifier for the original (romanised) content. The pipeline of this system (V2) is shown by the dotted lines in Figure 1. We achieve 71.27% average cross-validation accuracy in this method (V2) (fourth row of Table 3).

4.3 Stacking

This method uses *non-romanised monolingual and romanised code-mixed data* with handcrafted, stacking and combined features. This method follows the approach of Solorio and Liu (2008) with necessary adjustments. In this method, romanised code-mixed content is transliterated blindly in all languages and is presented to different POS taggers (trained with non-romanised monolingual data) as in method V2. The romanised words and the output from the monolingual taggers are used as features to train an SVM classifier on romanised code-mixed content. To keep our methodology as similar as possible to Solorio and Liu (2008) we follow the steps described below:

1. We train a Bengali and a Hindi TreeTagger (Schmid, 1994) using the SNLTR corpus with

[5]https://developers.google.com/transliterate

[6]http://nltr.org/snltr-software/

[7]http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/

[8]We use the English TreeTagger module to keep our setup as similar as possible to Solorio and Liu (2008). Other taggers such as the CMU ARK tagger (Owoputi et al., 2013) could also be tried.

[9]We also implement a system where all the tags in the SNLTR corpus are converted to universal POS tags before training. This variant does not outperform the current system.

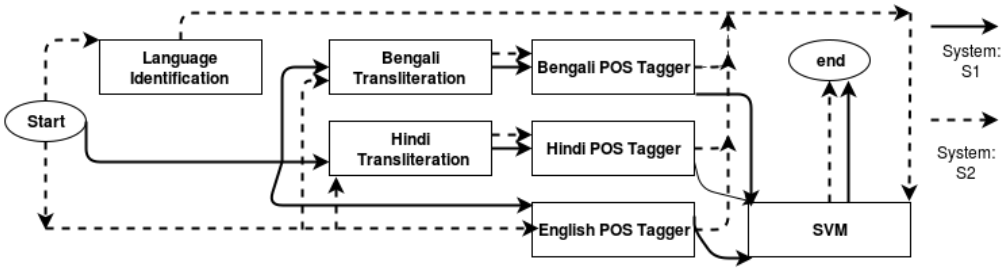

Figure 2: Stacked systems: system S1 and S2 (Section 4.3).

default settings as described in Section 4.2.

2. We transliterate each token of a sentence into Hindi and Bengali irrespective of its language using Google Transliteration as in system V2.

3. After transliteration we send each transliterated output to the respective TreeTagger, i.e. we send the original sentence to the English TreeTagger, Bengali transliterated output to the Bengali TreeTagger and the Hindi transliterated output to the Hindi TreeTagger.

After that we follow the stacking approach of Solorio and Liu (2008). Here, we stack an SVM classifier on top of the predictions generated by the TreeTaggers. We train a linear kernel SVM with stacking features and optimise parameter C in five fold cross-validation. The black lines in Figure 2 show the pipeline of this system (S1). The average cross-validation accuracy of this system is shown in the fifth row of Table 3 – 86.57% . Given the setup, we further experiment by using the combined features from romanised and transliterated tokens and also consider SVM language predictions as a feature. We observe that combining these features boosts the accuracy. After trying combinations of these features the best accuracy (87.59%) is achieved by adding all features together (S2) (sixth row of Table 3). The architecture of the system is shown by the dotted lines in Figure 2.

We also investigate the use of pipeline systems in stacking. The idea is to use all the predictions from a pipeline system and feed them into an SVM classifier. The stacked version of V1 (stacked-V1) achieves 85.99% and the stacked version of V2 (stacked-V2) achieves 85.83% average cross-validation accuracy with SVM using combined features. The black lines in Figure 3 show the pipeline of S3, stacked-V1 and dotted lines show the pipeline of S4, stacked-V2. These methods do not outperform our implementation of Solorio and Liu (2008)'s method S1 or its extended version S2.

4.4 Joint Modelling

To reduce error propagation from the LID module to POS tagging, we jointly model these two tasks using a 2-level factorial CRF (FCRF). In a linear-chain CRF, there is only one input level ($x = x_{1:T}$) and one output level ($y = y_{1:T}$) (see Figure 4). The conditional probability in a linear-chain CRF is expressed by Equation 1:

$$p(y|x) = \frac{1}{z(x)} \prod_{t=1}^{T} \psi_t(y_t, y_{t-1}, x_t) \quad (1)$$

$$\psi_t(y_t, y_{t-1}, x_t) = exp \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t). \quad (2)$$

$$z(x) = \sum_{y} \prod_{t=1}^{T} \psi_t(y_{t,l}, y_{t-1,l}, x_t) \quad (3)$$

$$p(y|x) = \frac{1}{z(x)} \prod_{t=1}^{T} \prod_{l=1}^{L} \psi_t(y_{t,l}, y_{t-1,l}, x_t) \varphi_t(y_{t,l}, y_{t,l+1}, x_t) \quad \text{where, } y_{T,L+1} = 1. \quad (4)$$

where, ψ_t represents clique[10] potential functions and is expressed by Equation 2. Here, K is the number of feature functions (f_k). The denominator $z(x)$ is the partition function, which is the sum over all 'y's and it is expressed by Equation 3.

[10] A clique in an undirected graph is formed with two vertices if there exists an edge connection between them.

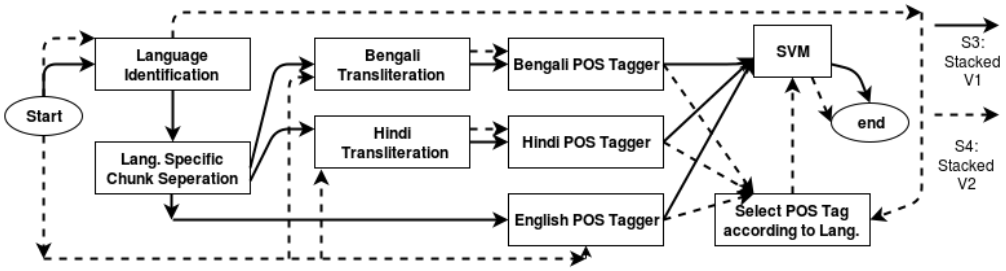

Figure 3: Pipeline systems in stacking: S3 (stacked-V1) and S4 (stacked-V2).

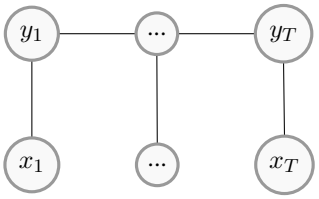

Figure 4: Graphical structure of a linear-chain CRF, where $(y_1, ..., y_T)$ represents language or POS labels and $(x_1, ..., x_T)$ is the observed sequence (tokens).

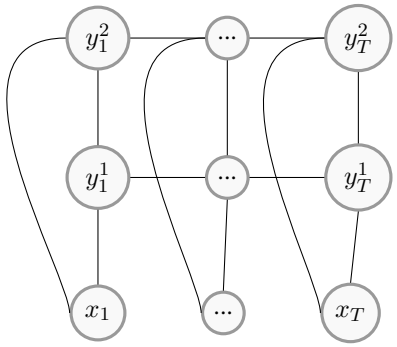

Figure 5: Graphical structure of the 2-Level factorial CRF, where $(y_1^1, ..., y_T^1)$ represents language labels, $(y_1^2, ..., y_T^2)$ represents POS labels and $(x_1, ..., x_T)$ is the observed sequence (tokens).

A factorial CRF (see Figure 5) combines multiple linear-chain CRFs, one for each output level. Unlike linear-chain CRFs, an FCRF deals with a vector of labels. In our case, the vector contains two labels, a language label ($y^1 = y^1_{1:T}$) and a POS label ($y^2 = y^2_{1:T}$). The inputs ($x = x_{1:T}$) are shared among these output labels (e.g. $y^1_{1:T}$ and $y^2_{1:T}$) and the output labels also have interconnections (y^1_i and y^2_i $\forall i = 1, 2, ..., T$). The conditional probability is expressed by Equation 4, where L is the number of levels (in our case L = 2), ψ_t represents transitions in each level (e.g. y^1_1 to y^1_2) and φ_t represents contemporal connections between two levels (e.g. y^1_1 to y^2_1). The denominator $z(x)$ is the partition function.

We implement this FCRF using the GRMM toolkit (Sutton, 2006). We use three different feature sets in our experiments. In cross-validation we find that, using handcrafted features, the average language tagging accuracy is 89.37% and average POS tagging accuracy is 81.77%. Use of stacked features gives 90.60% LID accuracy and 85.28% POS tagging accuracy. Finally, the combined feature set achieves 92.49% accuracy in LID and 85.64% in POS tagging (see Table 6 and the last row of Table 3).

5 Analysis and Discussion

We perform manual error analysis on the first test split of cross-validation. This split is a collection of 246 posts and comments with 5,044 tokens.

5.1 Effect of LID and Transliteration as Pre-processing modules

The most frequent error category for the SVM LID classifier is the confusion of Hindi words as Bengali words. We believe that the reason behind this is the small number of Hindi tokens in our training data. Most of these errors occur for tokens which are lexically identical in Hindi and Bengali, e.g. *'na'*, *'chup'*, *'sale'* and *'toh'*. All systems are trained with our SVM language classifier prediction. To quantify the error propagation from SVM language prediction we repeat the experiments of V1, V2 and S2 with the gold language labels and observe that the performance of each systems is slightly increased (Table 5).

We manually evaluate the accuracy of Google's transliterations for Bengali and Hindi. For Hindi,

Type	Systems	Acc.
Baseline	SVM	85.00%
	CRF	83.89%
Pipeline	V1: Vyas	71.12%
	V2: Extn. of V1	71.27%
Stacking	S1: Solorio	86.57%
	S2: Extn. of S1	**87.59%**
	S3: Stacked-V1	85.99%
	S4: Stacked-V2	85.83%
Joint Model	FCRF	85.64%

Table 3: Average cross-validation accuracy of POS tagging systems.

			% of Total Error		
Total	**Gold**	**Pred.**	**V2**	**S2**	**FCRF**
41	NUM	PRT	0.73	0.00	0.00
138	X	NOUN	0.57	0.10	0.00
138	X	ADJ	0.36	0.00	0.00
421	**ADJ**	**NOUN**	**0.32**	**0.26**	**0.29**
150	CONJ	NOUN	0.26	0.06	0.08
246	ADP	NOUN	0.15	0.08	0.12
147	ADV	NOUN	0.23	0.15	0.19
843	VERB	NOUN	0.26	0.12	0.14
246	ADP	PRON	0.09	0.09	0.10

Table 4: Top error categories produced by top three POS tagging systems.

Systems	Gold LID	SVM LID
V1	72.09	71.12
V2	72.07	71.27
S2	88.92	87.59

Table 5: POS tagging accuracy of V1, V2 and S2 with gold language labels and predicted (SVM) language labels.

Features	LID Accuracy	POS Accuracy
Handcrafted	89.37%	81.77%
Stacking	90.60%	85.28%
Combined	92.49%	85.64%

Table 6: Performance of FCRF with handcrafted, stacking and combined feature set. The detail of these features are described in Section 4.

transliteration accuracy is 82.63% and for Bengali it is 86.71%. Most of the transliteration errors occurs for those tokens which (i) have a single character (e.g. *'k'*, *'j'*, *'r'*), (ii) have digits (e.g. *'2mi'*, *'2make'*, *'as6e'*) and (iii) have shortened spellings (e.g. *'amr'*, *'tmr'*, *'hygche'*). Our inspection of transliteration errors reveals that the transliteration accuracy depends on the normalisation of romanised tokens.

5.2 Statistical Significance Testing

For statistical significance testing we use two-sided bootstrap re-sampling (Efron, 1979) by implementing the pseudo-code of Graham et al. (2014). We find that the small improvement of V2 over V1 is statistically significant ($p = 0.0313$). However, the 0.93% improvement of S1 over system FCRF is not. Among other systems, we find that FCRFs and SVMs are significantly better than the monolingual tagger combinations (V1 and V2).

5.3 Stacked vs Pipeline Systems

A reason for the poor accuracy of V1 and V2 is the difference between training and test data. The TreeTaggers are trained on monolingual non-romanised formal content while the test data is romanised code-mixed social media content. Secondly, error propagation through transliteration and LID also have a role to play. We find that the accuracy of Bengali transliteration is 86.71% and for Hindi it is 82.63%. This can be a reason for the poor performance of the Bengali and the Hindi TreeTagger. Furthermore, Table 5 shows that errors introduced by automatic LID cause an absolute loss of accuracy of 0.97% for V1 and 0.80% for V2. The accuracy of these systems improves (12.98% for V1 and 12.97% for V2) when we engage these systems in stacking using in-domain training data (see stacked-V1 and stacked-V2 in Table 3). We find that choosing the tagger(s) based on LID does not help in stacking approaches (e.g. stacked-V1 and stacked-V2) but using all taggers to generate features for the stacked classifier results in higher accuracy (e.g. S1 and S2). We find that the stacked system S2 outperforms other POS tagging systems in our experiments (see Table 3).

5.4 Effect of Joint Modelling

The accuracy of POS tagging in our joint modelling approach using romanised code-mixed data is higher than monolingual tagger combinations V1 and V2, but it is outperformed by S2 and other stacking ap-

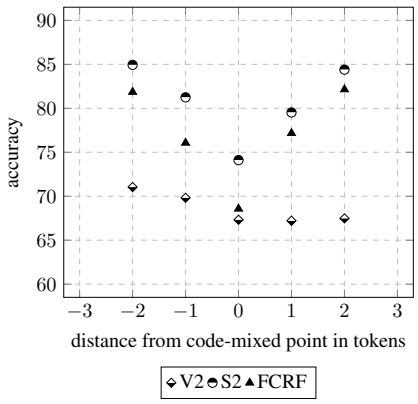

Figure 6: POS accuracy at code-mixed points and surroundings. The analysis is based on the first test split of cross-validation.

proaches. Table 6 shows the performance of FCRF with different feature sets. We find that combining handcrafted and stacking features achieves highest accuracy for both LID and POS tagging. In cross-validation FCRF with the combined feature set outperforms our SVM language classifier and achieves a reasonable cross-validation POS tagging accuracy of 85.64%, which is 2.05% less than the best stacking approach S2.

5.5 Monolingual vs Code-mixed Sentences

We choose the top POS tagging systems of each kind (V2, S2 and FCRF) and analyse the results in more detail on the first test split of cross-validation. First we test the accuracy on code-mixed sentences and on monolingual sentences. The results are depicted in Figure 7. V2 achieves 70.49% accuracy on code-mixed sentences and 72.20% on monolingual sentences. S2 achieves 83.42% on code-mixed sentences and 86.23% on the monolingual sentences. FCRF achieves 81.78% in code-mixed and 84.58% on monolingual sentences. All these systems perform better for monolingual sentences than their performance in code-mixed sentence. This result supports the hypothesis that performing POS tagging is harder on code-mixed sentences than it is on monolingual sentences.

5.6 Known and Unknown Words

Figure 7 also shows the performance of each system for known and unknown words based on the first training fold of romanised code-mixed data. All systems perform better for known words than for unknown words, as expected. We find that S2 and FCRF perform very closely for unknown words. For known words, S2 achieves 2.82% better accuracy than FCRF. The known-unknown analysis for pipeline system, e.g. V2, differs from the stacking (S2) and the FCRF-based methods. All pipeline systems are trained on *non-romanised monolingual data* (SNLTR Bengali and Hindi corpus). On the other hand, stacking and FCRF based systems are trained on *romanised code-mixed data*. Hence, for V2, we compare tokens of the test split with the tokens of the SNLTR Bengali and Hindi corpus to complete the analysis. We find that 52% of test tokens (Bengali and Hindi) are present in the monolingual training data, these are known words to the systems. V2 achieves 78.30% accuracy for the known Bengali and Hindi words and 43.80% for the unknown Bengali and Hindi words. As we use the default English model (distributed with the TreeTagger package) and not an English corpus, we do not perform this analysis for English words for V2.

5.7 Code-mixing Points

We also observe that the POS tagger accuracy depends on the distance to the code-mixed points. We consider a token as a code-mixed point (token-0) if the language of the token has been changed compared to the language of the previous token. Figure 6 shows the result of our analysis, where +1 means one token to right of a code-mixed point and -1 means one token to the left. It can be seen that all tested methods perform poorly at code-mixed points. Performance of these systems increases by the distance to code-mixed points. Among these systems, the ranking is independent of the distance to the code-mixed point.

5.8 Error Categories

The top error categories produced by different systems are shown in Table 4. The most common error pattern produced by all three systems (see fourth row of Table 4) is ADJ-NOUN, i.e. English adjectives that are classified as NOUN. The number of these errors decreases with the better performing models, as expected. We observe that most of the chat-specific tokens (e.g. emoticons) are misclassified by V2. This system is trained with formal content. There-

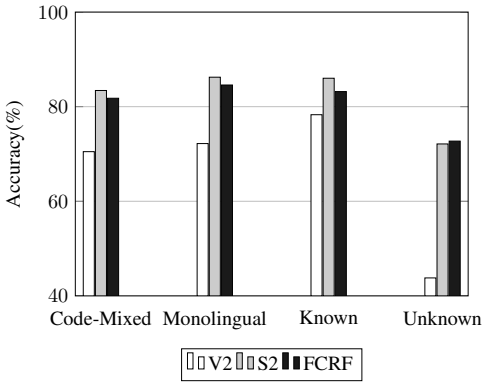

Figure 7: Some evaluation results of POS tagging. The analysis is based on the first test split of cross-validation.

fore, these tokens are misclassified as noun and adjectives by V2. These errors are rectified in S2 and FCRF. Other common error categories produced by the three systems are ADV-NOUN (adverb predicted as noun), VERB-NOUN (verb predicted as noun), CONJ-NOUN and ADP-NOUN.

6 Conclusion

We have presented a trilingual code-mixed corpus with POS annotation. We have performed POS tagging using state-of-the-art methods and also investigated the use of an FCRF-based joint model for this task. We find that the best stacking method (S2) that uses the combined features (see Section 4) performs better than the joint model (FCRF) and the pipeline systems. We also observe that joint modelling outperforms the pipeline systems in our experiments.

FCRF lags behind the best POS tagging system S2. Perhaps, using more training data would help FCRF to achieve better performance than S2. We consider this as a future work. The tagger combinations use either no context or junk context (transliterations) for POS tagger input. As a future work it would be interesting to modify these junk transliterations using a language model to provide meaningful context to the POS tagger.

Acknowledgement

This research is supported by the Science Foundation Ireland (Grant 12/CE/I2267) as part of CNGL (www.cngl.ie) at Dublin City University. The authors wish to acknowledge the DJEI/DES/SFI/HEA for the provision of computational facilities and support.

References

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. EMNLP 2014, Conference on Empirical Methods in Natural Language Processing*, pages 13–23, Doha, Qatar. Association for Computational Linguistics.

Monojit Choudhury, Gokul Chittaranjan, Parth Gupta, and Amitava Das. 2014. Overview of FIRE 2014 Track on Transliterated Search. `http://www.isical.ac.in/~fire/working-notes/2014/MSR/2014-trainslit_search-track_over.pdf`.

Amitava Das. 2016. Tool contest on POS tagging for code-mixed Indian social media (Facebook, Twitter, and Whatsapp) text. `http://amitavadas.com/Code-Mixing.html`, retrieved 2016-06-10.

B. Efron. 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26.

Debasis Ganguly, Johannes Leveling, and Gareth J. F. Jones. 2012. DCU@ FIRE-2012: rule-based stemmers for Bengali and Hindi. In *FIRE 2012, Forum for Information Retrieval and Evaluation*, pages 34–42.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized significance tests in machine translation. In *Proceedings of the ACL 2014 Ninth Workshop on Statistical Machine Translation*, pages 266–274.

Anupam Jamatia and Amitava Das. 2014. Part-of-speech tagging system for Indian social media text on Twitter. In *Social-India 2014, First Workshop on Language Technologies for Indian Social Media Text, at the Eleventh International Conference on Natural Language Processing (ICON-2014)*, volume 2014, pages 21–28.

Wei Lu and Hwee Tou Ng. 2010. Better Punctuation Prediction with Dynamic Conditional Random Fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 177–186, Cambridge, MA, October. Association for Computational Linguistics.

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. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, pages 380–390, Atlanta, Georgia, June. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Prakash B. Pimpale and Raj Nath Patel. 2015. Experiments with POS Tagging Code-mixed Indian Social Media Text. `http://amitavadas.com/ICON2015/CDACM_ICON2015.pdf`.

Kamal Sarkar. 2015. Part-of-Speech Tagging for Code-mixed Indian Social Media Text at ICON 2015. `http://amitavadas.com/ICON2015/JU_ICON2015.pdf`.

Helmut Schmid. 1994. Part-of-speech tagging with neural networks. In *Proceedings of the 15th Conference on Computational Linguistics - Volume 1*, COLING '94, pages 172–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arnav Sharma and Raveesh Motlani. 2015. POS Tagging for Code-Mixed Indian Social Media Text : Systems from IIIT-H for ICON NLP Tools Contest. `http://amitavadas.com/ICON2015/IIITH_ICON2015.pdf`.

Thamar Solorio and Yang Liu. 2008. 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.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steve Bethard, Mona Diab, Mahmoud Gohneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirshberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code-Switching. EMNLP 2014, Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar. Association for Computational Linguistics.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine Learning Research*, 8:693–723.

Charles Sutton. 2006. GRMM: Graphical Models in Mallet. `http://mallet.cs.umass.edu/grmm`, retrieved 2016-08-11.

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, Doha, Qatar, October. Association for Computational Linguistics.