Dependency Parsing of Bengali-English Code-Mixed Data enhanced with a Synthetic Treebank

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computational Linguistics
by Research

by

Urmi Ghosh
200925006
urmi.ghosh@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
August 2020

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Dependency Parsing of Bengali-English Code-Mixed Data enhanced witha Synthetic Treebank" by Urmi Ghosh, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Dipti Misra Sharma

To my Parents

Acknowledgments

I would like to express my sincerest gratitude to my thesis adviser Prof. Dipti Misra Sharma for her continuous support of my MS research. Her door was always open whenever I needed guidance, and her knowledge and feedback have been a great help for me throughout the years. She gave me enough independence for this thesis to be my own work, and at the same time steered me in the right direction whenever I needed help. The vast knowledge she has in NLP has enabled me to grow and learn a lot from her. Apart from research, her encouraging words have comforted me during difficult times over the years. Besides my adviser, I also want to thank Dr. Manish Shrivastava for his invaluable input on my research. His feedback and ideas were crucial to my work. I feel honored to be associated with such great mentors.

I would like to thank Dr. Soma Paul and Dr. Aditi Mukherjee for taking time out for my research work whenever I needed their valuable insights and constructive feedback. I also want to thank all the other faculty members at IIIT Hyderabad, they have contributed towards my growth as a researcher and as a person.

I am extremely thankful to my fellow researchers Irshad Bhat and Riyaz Bhat for explaining and guiding me with their research work which played a key role in my research. They were always available to help me. Their guidance and mentorship has been immensely helpful for my thesis.

I thank my (ex-)labmates and dear friends Pruthwik Mishra, Arpita Batra, Kaberi Sau, Sambhav Jain, Himanshu Sharma and Prateek Saxena for the innumerable stimulating discussions that we had. Their support has been invaluable during my time in IIIT.

Last but not the least, I would like to thank my family for their unconditional love and support throughout writing this thesis and my life in general.

Abstract

Code-mixing (CM) refers to the mixing of various linguistic units (morphemes, words, modifiers, phrases, clauses and sentences) primarily from two or more participating grammatical systems within a sentence [10]. The development of CM NLP systems has significantly gained importance in recent times due to an upsurge in the usage of CM data by multilingual speakers. However, this proves to be a challenging task due to the complexities created by the presence of multiple languages together. The complexities get further compounded by the inconsistencies present in the raw data on social media and other platforms.

In this thesis, we explore methods to efficiently parse the immensely popular Bengali-English CM which is widely spoken in India and Bangladesh. We present a neural stack-based dependency parser for CM data of Bengali-English by utilizing pre-existing resources for closely related Hindi-English CM as well as monolingual treebanks for Bengali, Hindi, and English. To address the issue of scarcity of annotated resources for the Bengali-English CM pair, we present a rule-based system to computationally generate synthetic CM dataset from parallel treebanks of Bengali and English. The generated synthetic CM data does not require an overhead of preprocessing for language identification and normalization as we get orthographic norms from standard corpora. Next, we project automatic annotations from Bengali and Hindi-English treebanks to Bengali-English using simple heuristics creating a synthetic CM treebank for Bengali-English (Syn-BE). Incorporating Syn-BE into our neural stacking parser further improves its performance. Our final model achieves an accuracy of 89.63% for POS tagging and 76.24% UAS and 61.41% LAS points for dependency parsing. This model improves parsing results by 1.37% LAS points when compared with a stacking model utilizing pre-existing Hindi-English and other monolingual resources. We also present a dataset of 500 Bengali-English tweets annotated under Universal Dependencies scheme which can be utilized for evaluation of the system as well as providing seed training data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multilingualism is the norm rather than an exception in face-to-face and online communication for millions of speakers around the world [3]. Prevalence of multilingualism naturally leads to the phenomenon of code-mixing which is the juxtaposition, within the same speech utterance, of grammatical units such as words, phrases, and clauses belonging to two or more different languages [24]. Recently, code-mixing which was often only observed in speech, has pervaded almost all forms of communication due to the growing popularity and usage of social media platforms by multilingual speakers [45]. Processing such texts can give us an understanding of social media trends which represent an amalgamation of people's opinions. Standard NLP systems often fail to perform on CM social media data [14]. This is because the current systems are trained on monolingual texts adhering to the language-related syntactic and orthographic norms. Since CM social media data does not follow such norms, there is a need to develop CM NLP systems that can capture the complexities present in code-mixing. In our thesis, we focus on building a dependency parser for Bengali-English CM data. We also explore ways to synthetically generate CM data from standard parallel corpora so that the synthetic CM data adheres to linguistic norms as much as possible.

1.1 Code-Mixing vs Code-Switching

Code-mixing refers to the mixing of various linguistic units (morphemes, words, modifiers, phrases, clauses and sentences) primarily from two participating grammatical systems within a sentence [10]. This is essentially different from code-switching which refers to the co-occurrence of speech extracts belonging to two different grammatical systems [24]. The occurrence can be both inter-sentential or intra-sentential, however, there are strict phrasal boundaries and within one lexical unit, the syntax of only one language is maintained.

Examples 1 and 2 help illustrate the differences between code-mixing and code-switching.

Since the more recent works have not focused on the differences between the two phenomena, we will use these two terms interchangeably.

Apnar eyes er care er jonno aapni kotota aware ?
your eyes of care of for you how much aware ?
"How aware are you about the care of your eyes?"
(1) Code-Mixing

Apnar chokhera dekhashonar jonna aapni kotota jagruka ? You should be careful !
your eyes.*Gen* care.*Gen* for you how much aware ? You should be careful !
"How aware are you about the care of your eyes? You should be careful "
(2) Code-Switching

1.2 Related Work

There has been considerable effort in building CM NLP systems such as language identification [39, 51, 4, 45], normalization and back-transliteration [23]. Part-of-speech (POS) and chunk tagging for code-mixing data for various South Asian languages with English have been attempted with promising results [48, 38]. Ammar et al. (2016) [2] developed a single multilingual parser trained on a multilingual set of treebanks that outperformed monolingually-trained parsers for several target languages. In the CoNLL 2018 shared task, several participating teams developed multilingual dependency parsers that integrated cross-lingual learning for resource-poor languages and were evaluated on monolingual treebanks belonging to 82 unique languages [56]. However, none of these multilingual parsers have been evaluated on code-mixed data or adapted specifically for CM parsing. A dependency parser for Hindi-English code-mixing has been presented by Bhat et al. (2018) [8]. In comparison, Bengali-English code-mixing is left relatively unexplored barring significant works on language identification [18] and POS tagging [27] which serve as preliminary tasks for more advanced parsing applications down the pipeline. The main hindrance to the development of parsing technologies for Bengali-English stems from the lack of annotated resources for the code-mixing of this language pair. In this paper, we try to utilize the pre-existing resources for widely available monolingual Bengali, Hindi and English as well as Hindi-English code-mixing and adapt them for Bengali-English dependency parsing.

1.3 Contribution of the Thesis

The major contributions of the thesis can be summarized as :

1. **Bengali-English CM Treebank:** The first Universal Dependencies based Bengali-English treebank annotated with POS and dependency labels. The size of the treebank is 500.

2. **Code-Mixed Data Generator:** A rule-based system to synthetically generate Bengali-English CM data from Bengali and English parallel corpora through chunk harmonization and replacements.

3. **Synthetic Code-Mixed Treebank:** A proposal to generate synthetic Bengali-English treebank by dependency annotation projection from monolingual Bengali and Hindi-English CM treebank.

4. **State-of-the-art Dependency Parser for Bengali-English CM:** Our neural stacking based model jointly learns POS tagging and parsing and is trained on monolingual as well as gold and synthetic CM data. Our parser achieves an accuracy of 89.63% for POS tagging and 76.24% UAS and 61.41% LAS points for dependency parsing.

1.4 Thesis Overview

This thesis is organized into 7 chapters. We begin this chapter by introducing the phenomenon of code-mixing and the general background of NLP research in this area.

1. **Chapter 2:** In this chapter, we provide the necessary background for understanding Bengali-English code-mixing, its features and similarities with Hindi-English.

2. **Chapter 3:** In this chapter, we describe our method for data collection and annotation of Bengali-English CM tweets. We also describe in detail the Universal Dependencies (UD) Grammar formalism and our methods to adapt it for code-mixing.

3. **Chapter 4:** In this chapter, we present a baseline neural stack-based model to parse Bengali-English CM. We also present a second model that improves the baseline model by incorporating Hindi-English CM treebank to the source model.

4. **Chapter 5:** In this chapter, we present a rule-based system to generate synthetic code-mixed data by harmonizing chunks across parallel corpora and replacing them using some constraints. We provide human data evaluation of our synthetic data.

5. **Chapter 6:** In this chapter, we project automatic dependency annotations into our synthetic code-mixed data from Bengali and Hindi-English using simple heuristics. We incorporate the synthetic treebank into our neural stacking parser to achieve state-of-the-art CM parser for Bengali-English. We do a thorough error analysis for future work in this area.

6. **Chapter 7:** In this chapter, we conclude our thesis by providing a summary of the dissertation and putting forward suggestions for future work in this area.

Chapter 2

General Background

Bengali is widely spoken in India and Bangladesh. It is the second most widely spoken language in India after Hindi. With approximately 228 million native speakers and another 37 million as second language speakers, Bengali is the fifth most-spoken native language and the seventh most spoken language by total number of speakers in the world [11]. Due to British colonization, English continues to be an important language in the Indian subcontinent. It is used in higher education as well as in some areas of the Indian government. According to the 2011 Census, there are 129 million speakers of English in India. Bilingualism in the society leads to an abundance of Bengali-English code-mixed data, made glaringly obvious by the social media texts from the subcontinent.

2.1 Matrix vs Embedded Language

Bengali-English CM can have either of the two languages as the *matrix* language which by definition is the dominant language governing the grammar of the CM sentence. *Embedded* language is the other participating language which is inserted into the morphosyntactic frame of the matrix language [37].

Apnar	eyes	er	care	er	jonno	aapni	kotota	aware	?
your	eyes	of	care	of	for	you	how much	aware	?
How	aware	are	you	about	the	care	of your	eyes	?

(1) Bengali-English CM: Bengali Matrix Language

Bengalis	probably	have	a	“	khub	bhalo	”	button	on	their	keyboard
bengalis	probably	have	a	“	very	good	”	button	on	their	keyboard
Bengalis	probably	have	a	“	very	good	”	button	on	their	keyboard

(2) Bengali-English CM: English Matrix Language

In example 1, the matrix framework of the sentence follows the grammar of Bengali with English words plugged into its matrix. In example 2, the matrix framework of the sentence follows English

grammar with Bengali words inserted into it. Clearly, example 1 represents a Bengali-English CM with Bengali as the matrix language, while example 2 represents English as the matrix language of the CM sentence.

2.2 Features of Bengali-English Code-Mixing

Mixing two or more languages is not arbitrary. There are in-depth linguistic descriptions of code-mixing across different multilingual contexts [30, 46, 43]. Some important features of Bengali-English CM are as follows :

- *Native suffixation*: Inflection of borrowed English words by using Bengali morphological features is a common occurrence.
  For eg: song*era* where *era* is the possessive/genitive marker in Bengali

- *Bilingual Light Verbs*: English verbal nouns are often added with light verbs from Bengali leading to complex bilingual light verb constructions. The verbal inflections are observed by the Bengali light verb.
  For eg: upload *korbo* (”do”) which translates to *“will upload”*.

- *Mixed Word Order*: Due to the mixing of two typologically diverse languages, we observe mixed word orders within sentences.

I	went	home	tomar	kotha	bhabte	bhabte
PRON	VERB	NOUN	PRON	NOUN	VERB	VERB
I	went	home	your	talk	think	think

I went home thinking about you

  In this example the English fragment follows an SVO order while the Bengali fragment follows an SOV word order.

- *Presence of mixed plural marker*: Sometimes, plural markers from both the languages attach simultaneously to the same word. Such constructions occur when English plural words are embedded into the matrix of Bengali sentences. Bengali plural markers attach to such plural English words to emphasize code-mixing .

new	songs	gulo	bhalo	lagchhe	dada	!
JJ	NOUN	NOUN	JJ	VERB	NOUN	
new	songs		good	feels	brother	!

I like the new songs brother

In the example, songs*gulo*, the Bengali plural marker, *gulo* is attached to the plural English word *songs*.

However, such constructions are uncommon as affixes that belong to the embedded language are usually dropped while code-mixing.

2.3 Bengali-English vs Hindi-English Code-Mixing

Bengali and Hindi both belong to the Indo-Aryan group of languages and are typologically very similar. By the virtue of their similarity, we observe a close proximity between Bengali-English and Hindi-English code-mixing as well. Both of these language pairs deal with the challenges of mixing different typologically diverse languages; SOV word order[1] for Hindi/Bengali and SVO word order for English.

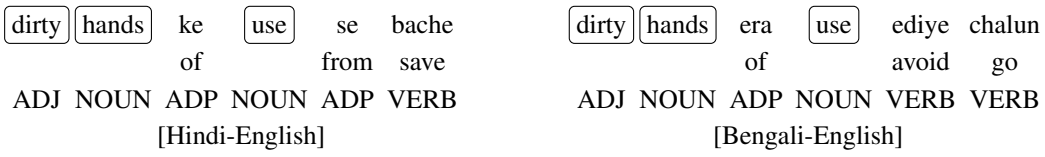


Figure 2.1: "Avoid the use of dirty hands". An example of a parallel Hindi-English and Bengali-English CM

Figure 2.1 illustrates the similarities in the mixing of English fragments with Hindi and Bengali.

In fact, all of the features of Bengali-English discussed in the previous section are in common with Hindi-English. Keeping this in mind, we are motivated to try and utilize resources for Hindi-English for Bengali-English.

2.4 Features of Social Media Data

In our thesis, we will be dealing with Bengali-English tweets. Such social media data is very noisy in nature making its processing a challenging task. Some common features of social-media data is as follows :

- Non-standard Spellings: Social-media data consists of spelling errors, short forms and other deviations. For eg: 2moro → tomorrow, ur → your
- Social Media Elements: Social media elements when occurring in the middle of a sentence are often treated as lexical units resulting in inconsistencies.
  For eg:

[1] Subject, Object and Verb Order in transitive sentences

  - (1) I liked #mnik a lot
  - (2) Lovely songs from the movie! #mnik.

  In the first example #mnik is treated as an object while in example 2, #mnik is in discourse.

- Lack of Punctuation: User-generated data often lacks punctuation making its processing difficult. Sentence demarcation is often missing, which leads to the issue of mislabelling the root node of the text. For eg: i saw the episode will upload it soon.

- Ellipsis: As social media data is closer to spoken language, many instances of ellipsis is witnessed. Handling ellipsis in a standard text is itself an issue and hence adds a layer of complexity to the processing of social media data. For eg: sounds great...come soon.

2.5 Summary

In this chapter, we briefed upon the basics of code-mixing for Bengali-English. We also discussed the features of Bengali-English CM and its similarity to Hindi-English.

Chapter 3

Universal Dependency Annotation of Bengali-English Tweets

In this chapter, we present our dependency annotated treebank of 500 Bengali-English tweets. We will elaborate on Universal Dependency (UD) Grammar - which is our dependency annotation scheme of choice. Also, we will explain how these guidelines have been adapted to handle the complexities of Bengali-English code-mixed data.

3.1 Introduction

Manually annotated linguistic resources are integral to the training as well as evaluation of dependency parsers. Such manually annotated resources, specifically syntactic treebanks are scarce in the code-mixed domain. Creating such manually annotated treebanks are expensive in terms of time and resources. The currently available datasets for Bengali-English only provide annotations for language and POS tags of a token. Recently, Bhat [7, 8] released a Universal Dependency annotated treebank for Hindi-English code-mixed data consisting of 1,900 tweets. We will use a similar formalism to create our Bengali-English code-mixed data consisting of 500 tweets. To the best of our knowledge, we are the first to publish a dependency treebank dataset for Bengali-English. To understand our annotation approach, we will briefly discuss the Universal Dependency formalism and its features.

3.2 Universal Dependency

Universal Dependencies (UD) is a framework for consistent annotation of grammar (POS, morphological features and syntactic dependencies) across different languages. The main goals of the project consist of facilitating multilingual parser development, cross-lingual learning and parsing research from a language typology perspective [40]. The annotation scheme is based on Stanford dependencies [20, 21, 19], Google universal POS tags [42] and the Interset interlingua for morphosyntactic tagsets [55]. The general philosophy of this framework is to provide a universal guideline scheme for consistent annotation of similar constructions across languages. Since we are dealing with multilingual-

ism within a treebank, following the UD annotation guideline provides for consistency and uniformity for similar constructions across Bengali and English.

3.2.1 Universal Dependency Guidelines: POS Tagsets and Syntactic Relations

Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

Table 3.1: Universal POS Tags

Universal POS	Definitions	Universal POS	Definitions
ADJ	adjective	PART	particle
ADP	adposition	PRON	pronoun
ADV	adverb	PROPN	proper noun
AUX	auxiliary	PUNCT	punctuation
CCONJ	coordinating conjunction	SCONJ	subordinating conjunction
DET	determiner	SYM	symbol
INTJ	interjection	VERB	verb
NOUN	noun	X	other
NUM	numeral		

Table 3.2: Universal POS Tags and their Definitions

- **POS Tags** : POS Tags mark the core part-of-speech categories. The 17 POS tags are organized into three categories:

1. Open Class : Commonly accepts the addition of new words. For eg: nouns, verbs, adjectives
2. Closed Class : New items are very rarely added. For eg: auxiliary verb, determiners, pronouns, conjunctions
3. Other : Miscellaneous category consisting of symbols, punctuation marks and some other special cases where a real POS category cannot be assigned.

The complete set of POS tags are listed in Table 3.1 and defined in Table 3.2.

	Nominals	**Clauses**	**Modifier words**	**Function Words**
Core arguments	nsubj expl iobj	csubj ccomp xcomp		
Non-core dependents	obl vocative expl dislocated	advcl	advmod discourse	aux cop mark
Nominal dependents	nmod appos expl nmod	acl	amod	det clf mark case
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj cc	fixed flat compound	list parataxis	orphan goeswith reparandum	punct root dep

Table 3.3: Universal Dependency Relations

- **Dependency Relations** : The 37 universal syntactic relations used in UD v2 are listed in Table 3.3 and the definition of each label is listed in Table 3.4. The upper part of Table 3.3 consists of rows that represent functional categories in relation to the head (core arguments of clausal predicates, non-core dependents of clausal predicates, and dependents of nominals) while the columns represent structural categories of the dependent (nominals, clauses, modifier words, function words). This arrangement follows the main organizing principles of the UD taxonomy. The lower part of the table lists relations such as those pertaining to coordination, multi-word expressions(MWE), parataxis, ellipsis, punctuation and various spelling errors which are not your typical dependency

UD Relations	Definitions	UD Relations	Definitions
acl	adjectival clause	fixed	fixed multiword expression
advcl	adverbial clause modifier	flat	flat multiword expression
advmod	adverbial modifier	goeswith	goes with
amod	adjectival modifier	iobj	indirect object
appos	appositional modifier	list	list
aux	auxiliary	mark	marker
case	case marking	nmod	nominal modifier
cc	coordinating conjunction	nsubj	nominal subject
ccomp	clausal complement	nummod	numeric modifier
clf	classifier	obj	object
compound	compound	obl	oblique nominal
conj	conjunct	orphan	orphan
cop	copula	parataxis	parataxis
csubj	clausal subject	punct	punctuation
dep	unspecified dependency	reparandum	overridden disfluency
det	determiner	root	root
discourse	discourse element	vocative	vocative
dislocated	dislocated elements	xcomp	open clausal complement
expl	expletive		

Table 3.4: Universal Dependency Relations and their Definitions

relations. Note: The *advmod* relation is used for modifiers for both predicates as well as other modifier words.

3.3 Annotation Procedure

We prepared a dataset of 500 Bengali-English tweets by crawling over Twitter using Tweepy[1] - an API wrapper for Twitter. We identified the Bengali-English tweets by running the tweets through a

[1]http://www.tweepy.org/

language identification system [7] trained on the dataset provided by ICON 2015.[2] We select only those tweets which satisfy a minimum code-mixing ratio of 30:70 (%). Here, the code-mixing ratio is defined as:

$$\frac{1}{n}\sum_{s=1}^{n}\frac{E_s}{M_s + E_s}$$

where *n* is the number of sentences in the dataset, M_s and E_s are the number of words in the matrix and embedded language in sentence *s* respectively. Next, we manually select 500 tweets from the resulting tweets and normalize and/or transliterate each word before annotating them using Universal Dependency guidelines [40] for POS and dependency tags. The language tags are annotated based on the tag set defined in [51, 27].

The dataset was annotated by two annotators who were mentored by two expert annotators who have been involved in major dependency annotation projects for the past 10 years and have previously annotated the Hindi-English treebank [8]. The inter-annotator agreement analysis was carried out by randomly selecting 100 tweets annotated by both the annotators. The inter-annotator agreement has a 97.60% accuracy for POS tagging and a 95.20% UAS and a 93.84% LAS for dependency parsing.

Language Tags	**Token Count**
Bengali (bn)	2840 (46.73%)
English (en)	1781 (29.3%)
Rest (univ,acro,ne)	1457 (23.97%)

Table 3.5: Token-level Data Distribution on 500 Bengali-English tweets.

The resulting dataset is split into three sets consisting of 200 tweets for testing, 160 for tuning and a third set of 140 tweets to be used as the "seed" training set in our stacking model for dependency parsing. The token-level data distribution is illustrated in Table 3.5. The Bengali-English CM dataset is available at `https://github.com/urmig/UD_bn-en`.

3.3.1 Code-Mixed Constructions and their Adaptations in UD

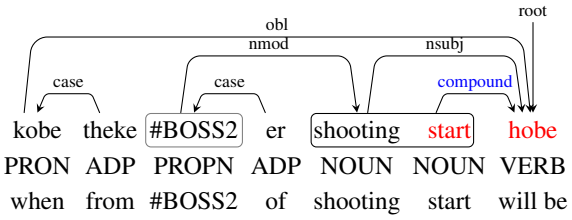
As discussed previously, one of the main motivations behind the UD project is to achieve a sense of consistency across multilingual treebanks. However, multilingualism within a treebank has not been the core motivation for this project. In this section, we will discuss in detail some of the challenges we faced while annotating the code-mixed data and our approach to handle them in UD.

- **Bilingual Light Verb Constructions**

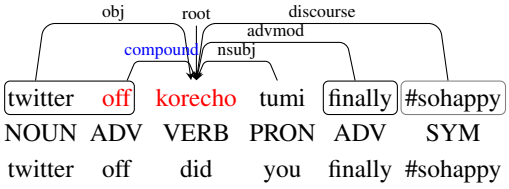
  In Bengali-English, we observe a great deal of Bengali verbification of English verbs/nouns re-

[2] http://ltrc.iiit.ac.in/icon2015/

sulting in *bilingual light verb constructions*. In this kind of linguistic construction, the English verb is either in an infinitive form or is nominalized while the verbal inflections are observed by the Bengali light verb. These code-mixed constructions are analogous to noun-light verb constructions in Bengali and hence we decided to annotate it in a similar manner. We tag the English token as *NOUN* and label the relation between the noun and the Bengali light verb as *compound*. Some English adverbs and adjectives are verbified in similar manner.



(1) "When will the shooting for #BOSS2 start"

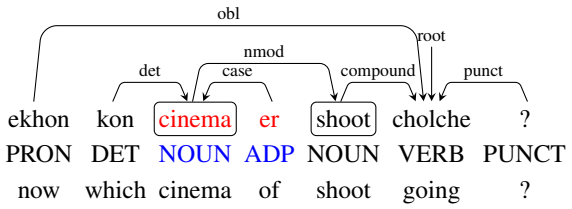


(2) " You finally turned off twitter #sohappy"

In example 1, Bengali verbification of English verb *start* by adding a Bengali light verb *hobe* ("will be") leads to a *hybrid* compound verb *start hobe* ("will start"). Similarly, in example 2, the adverb *off* is verbified by adding the light verb *korecho* ("did") and this translates to *turn off*.

- **Native Suffixation of Borrowed Words**

  Another common phenomenon is inflection of embedded English words according to the morphological structure of Bengali, resulting in unique code-mixed word constructions. We have decided to split these words at the inflection point. This can be achieved by simple suffix matching (Table 3.6) as the stem of the borrowed English words are not affected by the inflection marker. The most common suffix is the genitive marker *-r(a)/-er(a)* and its other variations.



(3) "The shoot of which cinema is going on?"

Case	Singular	Examples	Plural	Examples
Nominative	-ta/-ti	songta	-ra/-gulo/-guli	giftgulo
Accusative	-take/-tike	songtake	-derke/-guloke/-gulike	giftguloke
Genitive	-r/-er/-tar/-tir	songtar/songer	-der/-gulor/-gulir	giftgulor
Locative	-tay/-tate/-tite /-te/-e	songtay/songe	-gulote/-gulite	giftgulote

Table 3.6: Bengali Suffixation of English Words

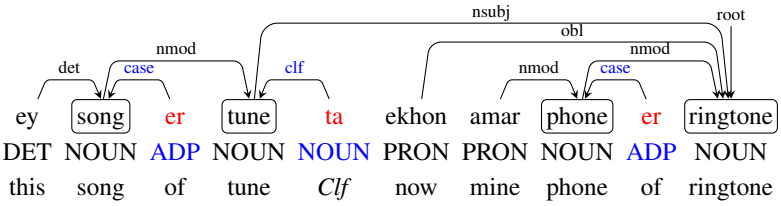
In example 3, the original hybrid word *cinemar* is formed by inflecting the English word *cinema* by the Bengali genitive case-marker *-r(a)*. When annotating, we have split the word into its stem form (NOUN) in English and postposition (ADP) case marker in Bengali.

A lexical item goes from being used as a foreign word to a valid *loanword* indistinguishable from the native vocabulary by virtue of repeated use and adoption of morpho-syntactic features of the recipient language [3]. However, it is generally agreed upon by linguists that it is very difficult to determine whether or not a borrowed word is a loanword or an instance of code-mixing. In our thesis, we consider a word to have been fully borrowed/loaned when it is reflected in the changes in spelling variations according to the phonetics of Bengali. For eg: *cabin* $\rightarrow$ *kebin*, *cinema* $\rightarrow$ *sinema* . In such cases, we have left the word as is and tagged it as 'bn' for Bengali as the word has been fully assimilated into Bengali and can be considered a loanword.

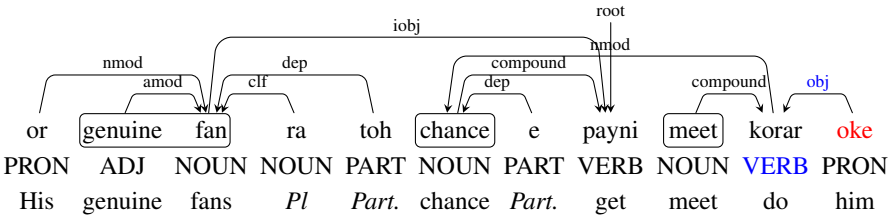
- **Distinct Tokens for Case-Markers and Classifiers**
  Bengali nouns and pronouns are inflected for case, including nominative, accusative, genitive (possessive), and locative [13]. When a classifier such as *-ta* (singular) or *-ra/-gulo* (plural) are added, nouns are also inflected for number. In informal Bengali tweets, it is common for such linguistic units to occur separately from the nouns and pronouns and exist as tokens. In UD, specific dependency labels (*case, clf*) are already present to handle them. For POS, we use ADP which marks the prepositions and postpositions and as *adpositions*. For classifiers, the most appropriate POS is still NOUN according to UD guidelines. Example 4 demonstrates how these tokens are handled in our treebank.

- **Word Order**
  As a head-final language, Bengali follows subject–object–verb (SOV) word order. However, less common but acceptable SOV word order is observed when it is mixed with English which follows SVO word order. We need to pay special attention to annotate such cases correctly.

  In spite of a more common SOV word order, Bengali exhibits free word-order to some extent due to the case-markers carrying relational information between the words. These long-distance



(4) "And the tune of this song is now the ringtone of my phone"

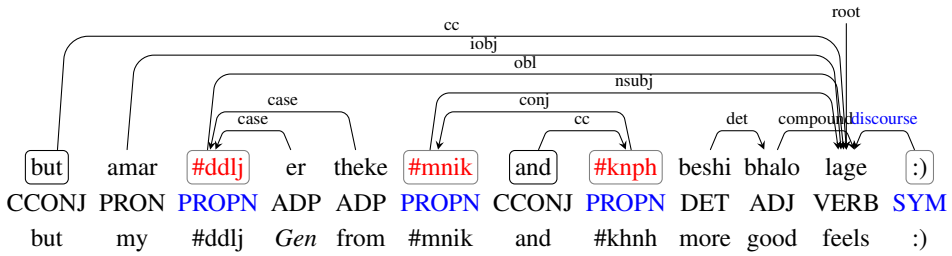


(5) "His genuine fans did not get a chance to meet him"

dependencies result in non-projectivity causing challenges in parsing. Example 5 illustrates SVO word order as well as non-projectivity caused by long-distance dependency.

- **Social Media Elements**

  The social media elements like *hashtags, at-mentions, emoticons* and *URLs* are handled according to the context in which they occur. If the hashtags and at-mentions serve as lexical units within a tweet, they are tagged as the normal word they represent without # or @.



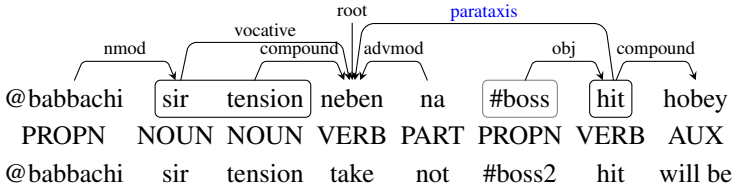
(6) "But I like #mnik and #khnh better than #ddlj :)"

For cases where these social-media elements are present outside of the sentence, we POS tag them as SYM, attach them to the root of the sentence and mark the UD relation as *discourse*. Example 6 demonstrates these two types of occurrences where #ddlj, #mnik and #khnh behave as proper nouns (names of movies) and are marked as PROPN, while the emoticon " :) " behaves as a regular social media element outside of the sentence and is tagged as SYM and attached to the root with a *discourse* relation.

- **Lack of End-of-Sentence Demarcation**

  We observed a large number of tweets with multiple sentences within one tweet without any end-

of-sentence markers. To handle such cases (example 7), we use the UD relation *parataxis* to link the subsequent sentences to the first sentence in a tweet.



(7) "Don't take any tension. #boss2 will be a hit"

In specific cases with distinct punctuation marks for marking the end of the sentence, we divided the tweets and treated them separately.

Some examples of annotated tweets from our Bengali-English treebank are illustrated in Figure 3.1

3.4 Summary

In this chapter, we discussed the Universal Dependency grammar formalism and the adaptations we made to the scheme in order to consistently annotate and build the dependency treebank for Bengali-English code-mixed tweets.

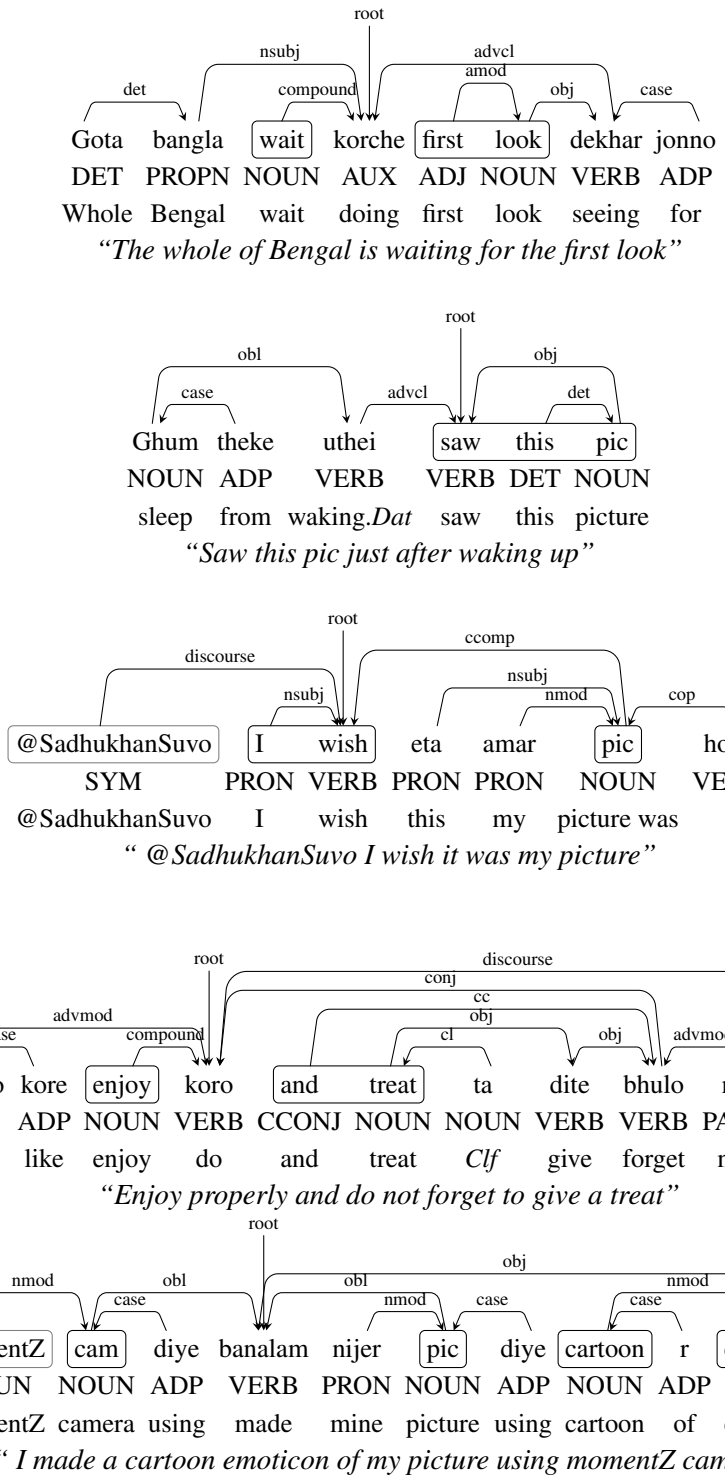


Figure 3.1: Some examples from the Bengali-English Code-Mixed treebank of tweets

Chapter 4

Towards Dependency Parsing of Bengali-English Code-Mixed Data

4.1 Introduction

In this chapter, we present a baseline neural stacking based parser for Bengali-English code-mixing trained on monolingual Bengali and English resources and our code-mixed gold annotated treebank of tweets. We also explore strategies to incorporate the pre-existing Hindi-English code-mixed treebank owing to similarities between Bengali and Hindi.

4.2 Dependency Parser

Our neural stack-based parser is an adaptation of the neural dependency parser by Bhat (2018) [8][1] which is a state-of-the-art parser for Hindi-English code-mixing. It is based on a transition-based parser [32] and is enhanced by neural stacks to incorporate monolingual syntactic knowledge with the CM model. This model jointly learns POS-tagging along with parsing by adapting feature-level neural stacks [57, 16].

We will briefly explain the basic features of the adapted neural stack parser:

4.2.1 Base Model

The base model is inspired by the stack-propagation model which consists of a stack of a POS tagger network and a parser network [57, 8]. The model is trained by optimizing a joint log-likelihood loss for both tagging and parsing. The tagger network is updated according to both tagging and parsing losses. The parser network is updated according to only the parsing loss but is regularized by shared parameters of the tagger network.

The tagger and parser networks each consist of an (a) input layer (b) feature layer and an (c) output layer. The input layers for both the tagger and parser encode the input sentence into word and character embeddings and pass it to the feature layer consisting of a shared bidirectional LSTM (Bi-LSTM) and an

[1] https://github.com/irshadbhat/csnlp

independent Bi-LSTM to the tagger. A feed-forward neural network with a softmax function is used in the output layer for a probability distribution over the POS tags. Next, the hidden representations from the shared Bi-LSTM and the dense representations from the feed-forward network of the tagger are concatenated and passed through the Bi-LSTM specific to the parser. Finally, a feed-forward network is used to predict the labeled transitions for the parser configurations. The working of this model is illustrated by the figure 4.1.

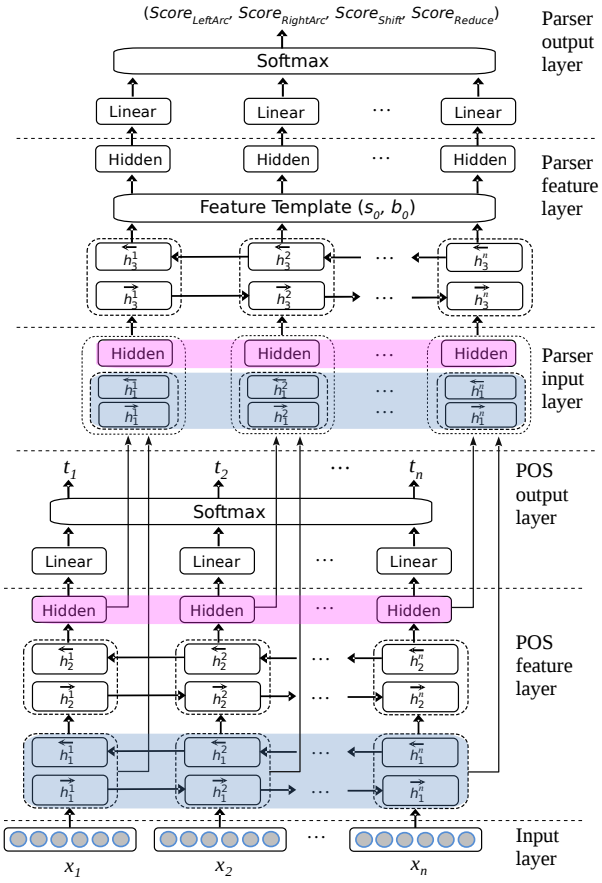


Figure 4.1: POS tagging and parsing network based on stack-propagation model proposed in [57]

4.2.2 Stacking Model

It has been demonstrated that even a limited amount of code-mixed data can significantly improve tagging and parsing performances on top of a system based on monolingual data of participating languages in the code-mixing [8]. The monolingual data helps in parsing monolingual chunks within code-mixed data while the code-mixed data can help in capturing the code-mixed grammar. We implement this by stacking a code-mixed model on top of a source model that incorporates monolingual syntactic information. Figure 4.2 demonstrates such a model for Bengali-English as per the model by Bhat (2018) [8] for Hindi-English. In this stacking model, the base model is trained on augmented

Bengali and English data (both monolingual) which provides for both tagging and parsing information to the code-mixed Bengali-English model trained on a seed training data of limited Bengali-English code-mixed data. In the tagger network, the input layer of the code-mixed tagger on top is augmented with the MLP layer of the base tagger. Similarly, in the parsing network, hidden representations from the parser specific Bi-LSTM of the base parser are augmented with the input layer of the code-mixed parser and the MLP layer of the base parser to the MLP layer of the code-mixed parser.

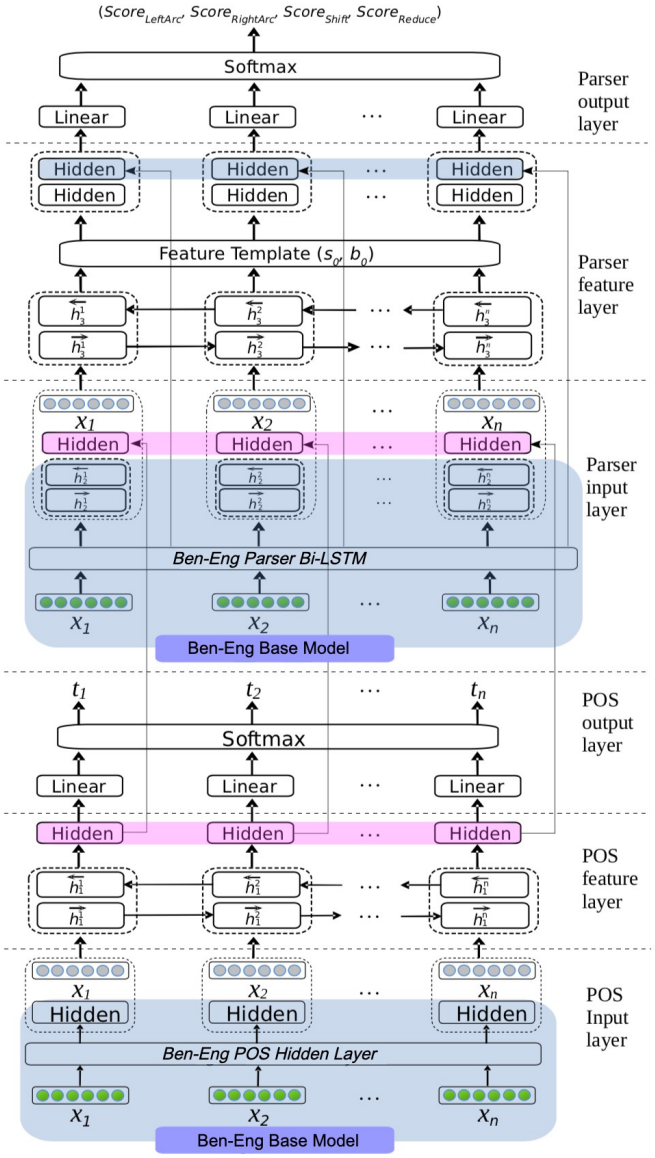


Figure 4.2: Neural Stacking-based parsing architecture for Bengali-English based on [8]

4.3 Experimental Setup

4.3.1 Preliminary Tasks

Language Identification (LID) and Transliteration (TRN) are two indispensable steps before parsing of code-mixed data. Identification of the language of each token is a necessary step before their normalization/transliteration which is important in order to address the inconsistencies prevalent in code-mixed social media texts.

For LID and TRN, we use the language identifier and a 3-step decoding transliterator by Bhat (2018) [8][2]. The LID is essentially a multi-layer perceptron (MLP) stacked on top of a bidirectional LSTM (Bi-LSTM) network. It is trained on the concatenated Bengali-English data set provided in the ICON 2015 shared task and our training set of 140 tweets. For TRN, we implement the 3 step decoder by training it for romanized Bengali words. For noisy English words, we use a pre-trained model provided in [8]. The model uses an attention-based encoder-decoder model [34] with global attention for learning. For Bengali, we train the model on the transliteration pairs extracted from ILCI Bengali-English parallel corpora (21,730). We further synthesize noisy transliteration pairs (25,017) by using simple rules such as dropping non-initial vowels, replacing consonants based on their phonological proximity, to generate synthetic data for normalization [8]. We achieve an accuracy of 95.08% for LID and 79.41% (non-standard tokens) for TRN over our test set (180).

4.3.2 Dataset Information

The monolingual treebanks for English and Hindi are provided by the UD-v2 treebanks.[3] Due to the absence of a Bengali UD treebank, we converted the Paninian annotation scheme [5] present in the Bengali treebank[4] to UD by slightly modifying the rules for Hindi [53] to Bengali. The trivial changes include adding a function to handle *classifiers* which are missing in Hindi but common in Bengali and replacing Hindi lexical files with that of Bengali.

For code-mixed Bengali-English, we use our treebank consisting of 500 tweets, created in Chapter 3 for training, testing and evaluation. The code-mixed Hindi-English data of 1900 tweets is provided by Bhat (2018) [8].[5]

4.3.3 Hyperparameters

The hyperparameters of the experiments are consistent with those in [8] :

[2]https://github.com/irshadbhat/csnli

[3]https://github.com/UniversalDependencies

[4]Developed as a part of the Indian Languages Treebanking Project by Jadavpur University

[5]https://github.com/irshadbhat/UD_Hindi_EnglishHIENCS

- **Lexical Representations**
  The characters are represented by 32-dimensional character embeddings while the words in each language are represented by 64-dimensional word2vec vectors [36] learned using the skip-gram model. The word embeddings are learned using the skip-gram model on Bengali (420k), English (280M) and Hindi (40M) monolingual wiki data. For character embeddings, randomly initialized embeddings within a range of [0.1, +0.1] are used.

- **Hidden Dimensions**
  The POS tagger and parser specific Bi-LSTMs have 128 cells and 256 cells respectively. The character Bi-LSTMs have 32 cells for both the models. The hidden layer of MLP for the tagger has 128 nodes and the parser has 256 nodes. The activation function used is the *tanh* function also known as the Tangent Hyperbolic function.

- **Learning Parameters**
  For learning, we use momentum SGD with a minibatch size of 1. The LSTM weights are initialized with random orthonormal matrices as per [47]. The dropout rate of 30% is set for tagger and parser Bi-LSTM and MLP hidden states. The models are trained for up to 100 epochs, and the best model is chosen based on the development set.

4.4 Experiments and Results

- **Model 1**: For our baseline model, we train the neural stacking model [8] for Bengali-English by training the source model on augmented Bengali and English treebanks and stacking it on a CM model trained on 140 Bengali-English CM (*Gold-BE*) sentences in our training set. Even though the size of the seed training set is limited, we benefit from the presence of unique CM grammar as well as syntactic information of social media elements. Our *bilingual* (augmented Bengali and English) source model serves to transfer both POS tagging and parsing information to the CM model.

  The baseline model gives us 62.78% UAS and 49.38% LAS points as shown in Table 4.2. The POS results give 79.39% accuracy. The lower accuracy for the model is expected due to the small training set for Bengali-English (140) when compared with Hindi-English (1448). Moreover, the significantly lower parser accuracy (a difference of ~9% LAS points) for Bengali in comparison to Hindi negatively impacts the performance of the source model (See Table 4.1).

- **Model 2**: In our next experiment, we aim to utilize the significantly larger treebank for Hindi-English and incorporate that into our model. Because of inherent structural and semantic similarity between Bengali and Hindi, we observe a close proximity between Bengali-English and Hindi-English code-mixing as discussed in Chapter 2. Using this assumption, we train the CM stacking model with 1448 Hindi-English CM data (*Gold-HE*) in addition to our 140 Gold-BE

Language	POS	UAS	LAS
Hindi	97.65	94.36	91.02
Bengali	93.26	87.07	80.1
English	95.80	88.30	85.30
Hindi-English	91.90	81.50	72.44

Table 4.1: POS and Parsing results of neural-stacking model for different languages

sentences. In order to fully capture the Hindi syntactic information in the CM data, we fortify the bilingual source model with the Hindi treebank resulting in a *trilingual* source model. We have attempted to reduce the differences in lexical representations belonging to Hindi and Bengali by using:

1. *Cross-Lingual Word Embeddings* for Hindi and Bengali by projecting the word2vec embeddings for the two languages into the same space by using the projection algorithm of artetxe2016learning and using a bilingual lexicon from ILCI parallel corpora.
2. *WX notation*[6] to represent characters from the two languages and using a common 32-dimensional character embedding space.

Our Model 2 improves both the POS and parser measurements significantly because it enables us to utilize the relatively large Hindi-English CM UD-annotated data. The UAS and LAS show an improvement in accuracy by 11.64% and 10.66% points respectively. The improvement in POS accuracy is ~8%. In this model, we slightly modify the word and character embedding representations in order to mitigate the lexical differences between Hindi and Bengali by using cross-lingual embeddings and a common character space. It should be noted that we have reported the results for gold language identification (LID) and transliteration (TRN). For auto LID and TRN, the accuracy of the tagger and parser diminishes as expected by ~3.7% for POS, ~4.3% for UAS points and ~4.7% for LAS points.

From Table 4.3, we observe that using cross-lingual embeddings improves the accuracy of tagging by 0.76%, UAS by ~0.6% points and LAS by ~0.5% points. Using a common character space by using WX notation further improves the accuracy of both tagging and parsing by ~1.8% and ~2.5% points respectively. The significant improvements in the results confirm the inherent similarity between the code-mixing grammar of Hindi and Bengali with English as both of these language pairs deal with the mixing of two typologically diverse languages.

[6]http://wiki.apertium.org/wiki/WX_notation

	Stacking Models	POS	UAS	LAS	POS	UAS	LAS
	(base) + (stacked)	Gold (LID + TRN)			Auto (LID + TRN)		
Model 1	(Bilingual) + Gold-BE	79.39	62.78	49.38	75.76	58.58	44.67
Model 2	(Trilingual) + Gold-(HE+BE)	87.43	74.42	58.62	83.73	70.04	53.84

Table 4.2: POS and Parser results of neural-stacking models for Bengali-English.

Embeddings	**POS**	**UAS**	**LAS**
Monolingual	84.86	71.32	56.93
Crosslingual	85.62	71.94	57.41
Crosslingual + WX notation	87.43	74.42	60.04

Table 4.3: Effect of embeddings on POS and Parser results for Model 2 = Trilingual + Gold-(HE + BE)

4.5 Conclusion

In this chapter, we have presented a baseline neural stacking based parser for Bengali-English. The baseline model gives us 62.78% UAS and 49.38% LAS points as shown in Table 4.2. The POS results give 79.39% accuracy. We also successfully explored incorporating Hindi-English data to enhance Bengali-English parsing. The UAS and LAS show an improvement in accuracy by 11.64% and 10.66% points respectively. The improvement in POS accuracy is ~8%.

Chapter 5

Generating Synthetic Bengali-English Code-Mixed Data

In this chapter, we will discuss in detail our motivation and methods to synthetically generate code-mixed Bengali-English data. The synthetic code-mixed data generated by our rule-based system achieves impressive scores on human evaluation and helps us in achieving a better understanding of Bengali-English code-mixing.

5.1 Introduction

The motivation behind trying to synthetically generate CM data comes due to a number of issues we face when using CM data available online.

- **Data scarcity:** Even though we have a plethora of data available on the web, the extent to which it is code-mixed might not be enough to train our systems. The majority of such data contains large monolingual chunks, with very few code-switching points.

- **Noisy Data:** The data we obtain from web-scraping is excessively noisy, mainly due to obtaining it from an informal social-media related setting. Such data needs to be pre-processed to further be used to train our CM models. Preliminary NLP tasks such as *language identification* and *normalization* and *transliteration* become a necessity and cause an overhead. The errors in pre-processing this data get further propagated into our CM systems reducing their accuracy.

Hence, we propose a rule-based system to synthetically generate valid code-mixed data using parallel Bengali and English corpora. An attempt has been made to generate code-mixing data for the Spanish-English language pair [44] but none for the Hindi-English or Bengali-English language pair. The Bengali-English pair poses special challenges due to the different word orders of Bengali (SOV) and English (SVO) which commonly violate most code-mixing theories [50]. We also present an annotation projection system to computationally generate a synthetic code-mixing treebank for Bengali and English (Syn-BE) which is used to further improve the accuracy of our dependency parser.

5.2 Code-Mixing Constraints

Kachru (1978) [31] provided us with an insight on code-mixing constraints in Indian languages on their Persianization or Anglicization. Constraints on code-mixing of Spanish-English have been examined as well [43, 46]. Bhatia and Ritchie (1996) [12] examined the different types of constraints on code-mixing proposed by different scholars in literature. Agnihotri (1998) [1] demonstrated a number of code-mixing constraints found in the literature to fail for Hindi-English. Sinha (2005)[50] discussed CM constraints for Hindi-English and came to the conclusion that the phenomenon of code-mixing for this language pair is not entirely arbitrary. We will discuss some of these constraints and examine their appropriateness for Bengali-English.

1. **The Equivalence Constraint**
   According to the Equivalence Constraint [43, 46], the code-mixing points should not violate a syntactic rule of either language, that is, mixing can only occur at points where there is a mapping between the surface structures of the two languages. For Bengali (SOV) and English (SVO) this constraint is violated frequently because they are typologically diverse [6].

2. **The Free Morpheme Constraint**
   The Free Morpheme Constraint (Poplack 1980, Sankoff and Poplack 1981) prohibits the mixing of a stem and a bound morpheme from different languages. However, such code-mixing constructions are very common in Bengali-English wherein an English word is inflected according to the grammar of Bengali. For eg: cinema*gulo* (cinemas), hospital*er* (of hospital)

3. **The Closed Class Constraint**
   The Closed Class constraint [52, 29] states that the matrix language elements within the closed class of grammar (possessives, ordinals, determiners, pronouns) are not allowed in code-mixing. This constraint holds true for Hindi-English [1, 50]. The constraint is tested for Bengali-English by working through the following example :

Apnar	chokher	dekhashonar	jonye	aapni	kotota	icchuk	?
PRON	NOUN	NOUN	ADP	PRON	DET	ADJ	PUNCT
your	eyes.*Gen*	care.*Gen*	for	you	how much	willing	?

(1) Bengali

How	aware	are	you	about	the	care	of	your	eyes	?
ADV	ADJ	VERB	PRON	ADP	DET	NOUN	ADP	PRON	NOUN	?

(2) English

Example 3 demonstrates an unnatural and uncommon code-mixed construction and thus we can conclude that this mixing constraint holds true for Bengali-English CM text as well. We note that

*Your	*chokher*	*about	*the	*dekhashonar*	*you	*how	*icchuk*	?
PRON	NOUN	ADP	DET	NOUN	PRON	ADV	ADJ	
your	eyes.*Gen*	about	the	care.*Gen*	you	how	willing	?

*(3) *Incorrect Code-mixing*

Apnar	eyes	er	care	er	jonno	aapni	kotota	aware	?
PRON	NOUN	ADP	NOUN	ADP	ADP	PRON	DET	ADJ	PUNCT
your	eyes	of	care	of	for	you	how much	aware	?

(4) Correct Code-mixing

the example 4 results in an acceptable code-mixed sentence as the closed class elements from the matrix language Bengali are retained.

5.3 The Code Mixing Pipeline

Based on the token-level data distribution in Table 3.5, we observe that the matrix language in the majority of CM sentences is Bengali. The same is observed for the Hindi-English CM Data by Sharma (2016) [48]. On further investigation of the Bengali-English data we collected in Chapter 3, we observe that the matrix language of only ~2% tweets is English and ~8% tweets contain relative clause construction where the antecedent and the subordinate clause belong to two different languages. Keeping this under consideration, we proceed with the synthetic data generation by mixing English linguistic elements into the matrix of Bengali sentences through direct chunk replacement.

The pipeline for our code-mixing script is as shown in Figure 6.2. Our code-mixing script consists of two modules : (a) Chunk Harmonizer and (b) Rule-based Chunk Replacement

5.3.1 Chunk Harmonizer

The script takes shallow-parsed English and Bengali parallel corpora as inputs. However, there are various structural differences in constituency parsing obtained for English by the Stanford Parser [33] and shallow parsing obtained for Bengali by the Shallow Parser by TDIL Program, Department Of IT Govt. Of India.[1]. Consistency across chunks in parallel sentences is imperative for direct replacement of chunks for code-mixing. To achieve this, our module handles the issue of structural differences in English and Bengali chunks by modifying the English chunks based on the following set of rules:

1. Separate the *coordinating conjunction* and its conjuncts into different chunks as they are treated separately in Bengali. The conjunctions are given the chunk tag CCP.
   For eg: (NP (NP (JJ Fresh) (NN breath)) (CC and) (NP (JJ shiny) (NNS teeth))) → (NP (JJ Fresh) (NN breath)) (CCP (CC and)) (NP (JJ shiny) (NNS teeth))

[1] http://ltrc.iiit.ac.in/analyzer/bengali/

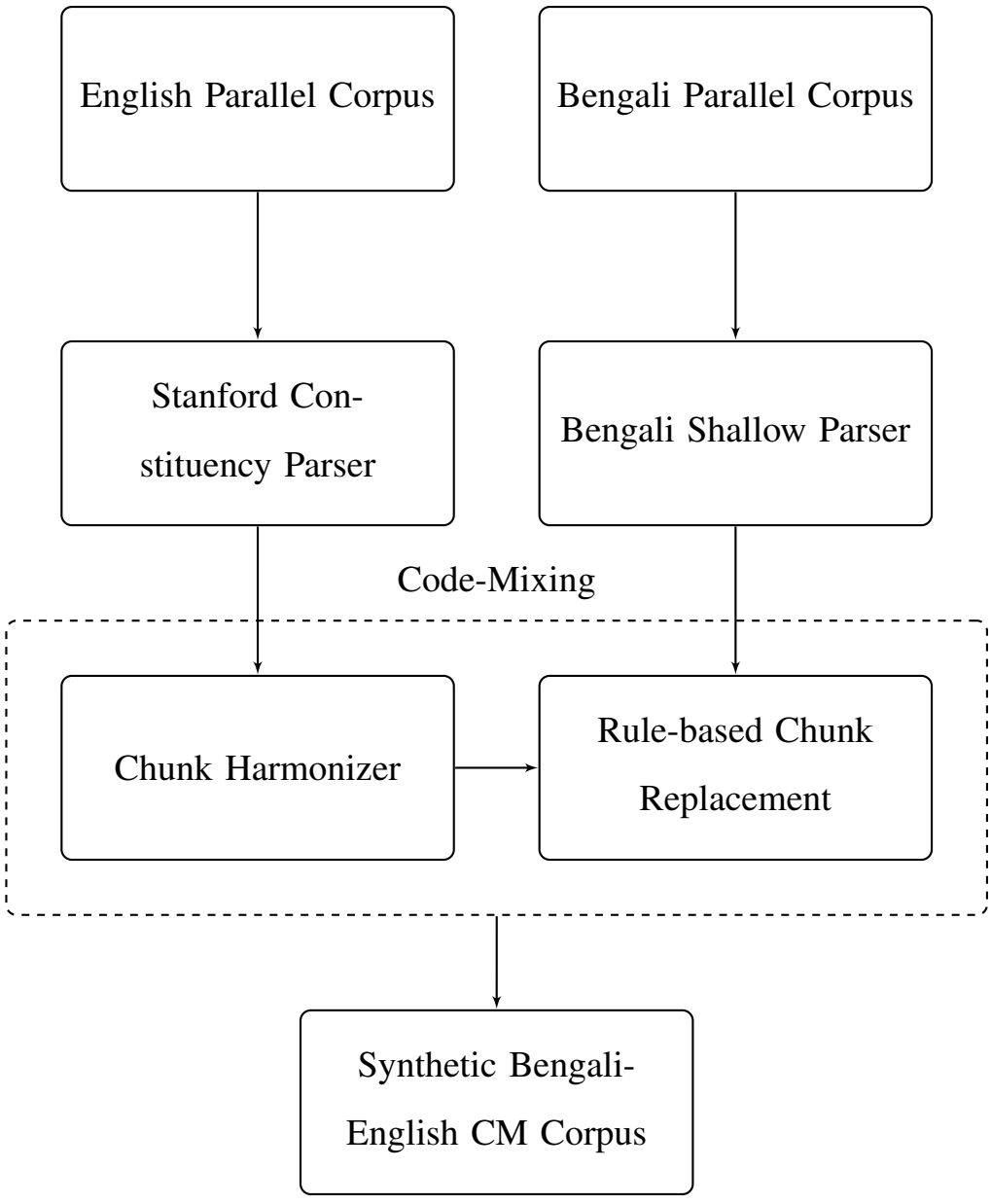


Figure 5.1: Schematic diagram of the Code-Mixing process

2. Combine the *adverbs of degree* (also, too, only) with the preceding noun phrase (NP) as they are classified in Bengali as *particles* (*o* ("also"), *i* ("only"), etc.) and grouped with NP.
   For eg: (NN water) (ADVP (RB only))→ (NP (NN water) (RB only))

3. Convert *prepositional phrase* (PP) to NP by making the head noun of the succeeding NP as the head and separating it from the preceding verb phrase (VP) if present. In Bengali, we have post-positions that are usually inflected in the nouns and pronouns and hence always chunked together in NP with the noun/pronoun as the head.
   For eg: (PP (IN from) (NN outside)) → (NP (IN from) (NN outside))

4. Split NP at *genitives (personal pronouns)* into separate NPs as genitives are considered as separate chunks in Bengali.
   For eg: (NP (PRP$ your) (NNS hands))→ (NP (PRP$ your)) (NP (NNS hands))

The rules described above are demonstrated by the examples below:

(NP Your self-confidence) (ADVP also) (VP increases (PP with (NP teeth))) →
(NP Your) (NP self-confidence also) (VP increases) (NP with teeth)

(5)

which now consistently maps to the corresponding chunks in the parallel Bengali sentence:

(NP daanter "teeth" jonyo "for") (NP aapnaar "your") (NP aatmaviswas "self-confidence" o "also") (VP baadhe "increases")

(6)

Along with harmonizing the chunks, this module marks the heads of each chunk in both the languages using generalized rules defined by Sharma (2006) [49]. For clarity, we have mapped the POS tags from Penn Treebank POS tagsets [35] for English and Bureau Of Indian Standard (BIS) POS tagset [17] for Bengali to the Universal Dependency Tagset [40].

5.3.2 Rule-based Chunk Replacement

The second module in the pipeline facilitates *rule-based chunk replacement* by taking the chunk-harmonized parallel Bengali and English sentences as inputs and replacing some selected Bengali chunks with English. For chunk replacement, we will be closely following *the Closed Class Constraint* which states that the matrix language elements within the closed class of grammar (possessives, ordinals, determiners, pronouns) are not allowed in code-mixing [52, 29]. We extend the closed class constraint to *question words* that can fall in the POS category of *adverbs* (ADV) and PRON as well as for *adpositions* (prepositions and postpositions).

First, the chunks, each represented by the head element, are aligned using word alignments obtained from Giza++ [41] on ILCI parallel corpus [28] consisting of 35k parallel sentences. We also use word2vec [36] vectors to reaffirm these alignments by doing a simple cosine similarity check. The word embeddings are learned using the skip-gram model on Bengali (420k) and English (280M) monolingual data.

Next, we follow the following rules for chunk replacement:

1. *NP Replacement*: A frequently observed phenomenon in CM data is the replacement of noun phrases in one language by the corresponding noun phrase in the other language partially or in entirety [22]. We retain Bengali post-positions and drop English prepositions associated with the heads by checking for the suffixes in Table 3.6. We do not replace NPs where the head noun is a pronoun as it belongs to the closed class.

2. *JJP Replacement*: We extend the noun replacement rules for adjectival phrases and replace the adjectival chunks (JJP) with the corresponding English chunks.

3. *Retainment of Closed Class Elements*: We retain the closed class elements like determiners, ordinals, pronouns, possessives, post-positions and question words within the NPs and JJPs.

4. *VP Retainment*: By keeping the verbal chunks (VP) intact, we ensure that Bengali is retained as the matrix language of the code-mixed sentence. Hybrid compound verbs (see section 3.3.1) are a common occurrence in Bengali-English code-mixing and we can successfully synthesize them by replacing the NP/JJP preceding Bengali light verbs. For eg: (JJP *porishkaara* ("clean")) (VP *koruna* ("do")) $\rightarrow$ (JJP clean) (VP *koruna* ("do")).

Mixing the Bengali sentence (6) with the parallel English sentence (5) will generate:

(NP ***teeth*** *er "of" jonyo "for") (NP aapnaar "your") (NP* ***self-confidence also*** *)*
(VP baadhe "increases")

(7)

This is one of the acceptable combinations of the two sentences to form a CM sentence. We use the parallel corpora for English, Bengali and Hindi provided by Indian Languages Corpora Initiative (ILCI) [28] belonging to the *health* domain. We select a subset of 10,000 parallel sentences from each language and generate code-mixed sentences for both Bengali-English and Hindi-English language pairs following the rules discussed above. Thus, we have a parallel corpora for code-mixed Bengali-English and Hindi-English along with parallel corpora for Bengali, Hindi and English. We obtain only 5,063 code-mixed sentences with a minimum CM ratio of 30:70 (%). The reason for this is attributed to the incorrect alignment of a few heads in many Bengali and Hindi sentences to the heads of the corresponding English sentence. Also, following our strict criteria eliminated the sentence pairs unsuitable for code-mixing.

5.4 Synthetic Code-Mixed Data Evaluation

For data evaluation, we relied on human evaluation and provided our dataset of 200 code-mixed sentences to each of the 2 independent evaluators - a native speaker of Bengali for Bengali-English and a native speaker of Hindi for Hindi-English. Both of them are non-native speakers of English. Our scoring scheme is as described in Table 5.1:

Score	Definition	Description
5	Perfect	natural construction
4	Good	unnatural construction (eg: word-order) but grammatically correct
3	Acceptable	minor errors (eg: incorrect inflection for number and gender)
2	Erroneous	miscellaneous errors but comprehensible
1	Unacceptable	non-comprehensible due to incorrect grammar and word substitution

Table 5.1: Synthetic Code-Mixed Data Evaluation Scheme

Using the scoring scheme, our set of synthetic Bengali-English CM data received a score of 4.49 out of 5 and Hindi-English scored slightly lower at 4.38. The errors observed by the evaluators can be categorized into :

1. **Word Repetition**: We generated a few erroneous sentences with consecutive word repetitions due to inconsistent chunking of multi-word expressions and noun-modifiers. To mitigate these errors in the post-processing step, we removed repeated words at code-mixing points using dictionary and cosine similarity between the words represented by their *cross-lingual embeddings* (see section 4.4).
   Eg: *chiniyukta* (“sugared”) sugared gum → sugared gum

2. **Incorrect Inflection**: When introducing English words in Hindi, inflections on post-positions and verbs are violated because the gender of ”borrowed” English words does not always map to their Hindi translation. Also, the gender mapping for borrowed words is arbitrary and often depends on the discretion of the speaker.
   For eg: *laar* (”saliva”) *ki* (”of”) *khaas* (”important”) *bhumika* (”role”) *hoti hai* (”is”) → saliva *ki* important role *hoti hai* which sounds grammatically unnatural. Here, *laar* and *bhumika* are feminine but their translation in English - *saliva* and *role* are intuitively masculine in Hindi-English. A more natural construction would be: saliva *ka* important role *hota hai*
   Bengali is a gender-neutral language and therefore synthetic Bengali-English is not affected by incorrect inflection due to borrowing. This is the reason why the overall score of synthetic Bengali-English is slightly better.

3. **Unnatural Constructions**: The substitution of Bengali chunks or words with their English alignment does not lead to a natural construction every time. Unnatural bilingual light verbs are a subtype of this kind of error.
   For eg:

   - *majboot* (”strong”) *banaye* (”makes”) → strengthen *banaye* (”makes”) instead of strengthen *kore* (”does”) or strong *”banaye”* (”makes”)
   - *click* (”click”) *ki* (”of”) *doori* (”distance”) *(*pe) (”at”) → click *ki* (”of”) away *pe* (”at”)

4. **Incorrect Word Substitution**: In spite of our best efforts, some Bengali/Hindi chunks are incorrectly aligned to English chunks and their substitution leads to incorrect and incomprehensible sentences.

5.5 Summary

In this chapter, we presented a rule-based system to generate synthetic Bengali-English code-mixed data from parallel treebanks for Bengali and English. Our method provides us with a way to generate relatively clean code-mixed data when compared with the informal data present in social media.

Chapter 6

Enhancing Dependency Parsing using Synthetic Bengali-English Code-Mixed Treebank

In this chapter, we present a simple annotation projection algorithm to project automatic annotations from Bengali and Hindi-English onto Bengali-English code-mixed data and create a synthetic code-mixed treebank (Syn-BE). We use this synthetic treebank to train our final parsing model. This model improves the POS tagging accuracy by 2.2% points and parser accuracy by 1.82% UAS points and 1.37% LAS points respectively.

6.1 Synthetic Bengali-English Treebank

In this section, we will discuss our approach to generate automatic dependency annotations for the synthetic Bengali-English code-mixed data we created in chapter 5 so that it can be utilized by our parser.

6.1.1 Cross-lingual Annotation Projection

Cross-lingual annotation projection makes use of parallel data to project annotations from the source language to the target language through automatic word alignment. Hwa (2002) [26] proposed some basic projection heuristics to deal with different kinds of word alignments. Recently, Tiedemann (2014) [54] proposed improvements in the annotation scheme by adding heuristics to remove unnecessary dummy nodes that are introduced in the target treebank to deal with problematic word alignments.

Our main assumption underlying annotation projection is that linguistic analysis for Hindi-English and Bengali will be also valid in Bengali-English. The Bengali-English code-mixed data comprises of grammatical fragments of Bengali and English as well as code-mixed grammar. Since we now have parallel data for Bengali-English (BE), Hindi-English (HE) and Bengali (B), we can investigate the utility of annotation projection from HE (for English as well as code-mixed fragments) and B (for Bengali as well as code-mixed fragments) to BE. Annotation projection for individual grammatical fragments of Bengali and English is a trivial task as the annotation for the subtree is simply copied from

source to target. However, for every dependency involving code-mixing, we have to make a decision whether to transfer annotation from HE or B based on some heuristics.

The grammatical fragments are illustrated in Figure 6.1 with an example.

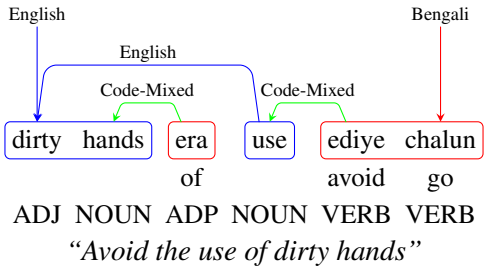


Figure 6.1: Code-mixed tweet with grammatical fragments from Bengali and English.

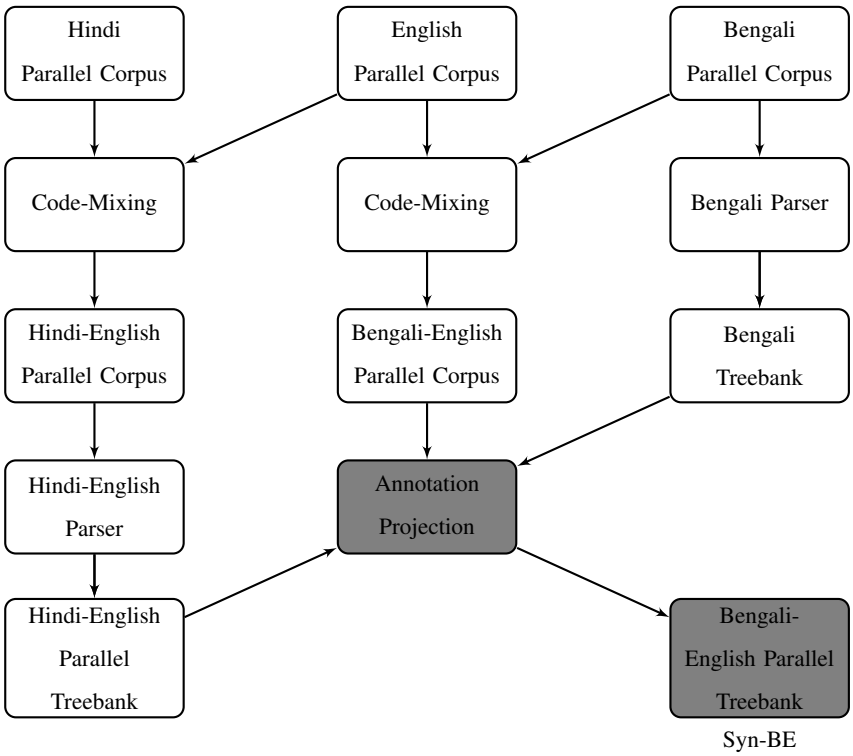


Figure 6.2: Schematic diagram of the Annotation Projection from Hindi-English and Bengali to Bengali-English parallel corpus.

HE treebank is created by parsing the HE generated in section 5.3 using the neural stacking dependency parser for Hindi-English [8].[1] B is generated by parsing the parallel Bengali sentences using the

[1]https://github.com/irshadbhat/csnlp

same neural stacking dependency parser trained on a monolingual Bengali dependency treebank. The POS tagging and parsing accuracy of these two parsers are mentioned in Table 6.1.

Language	**POS**	**UAS**	**LAS**
Hindi	97.65	94.36	91.02
Bengali	93.26	87.07	80.1
Hindi-English	91.90	81.50	72.44

Table 6.1: POS and Parsing results of neural-stacking model for different languages

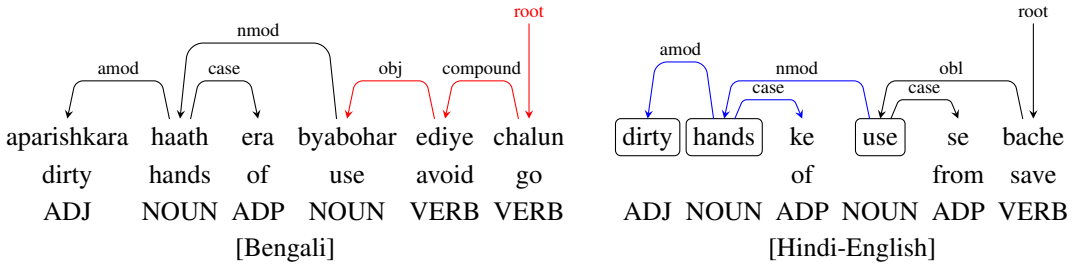
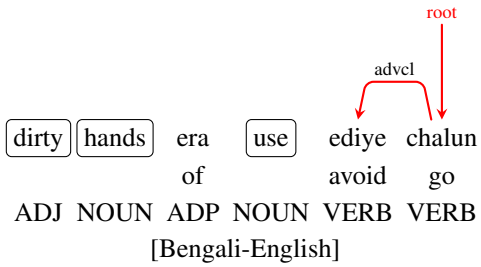


Figure 6.3: An example of a parallel Bengali and Hindi-English tree

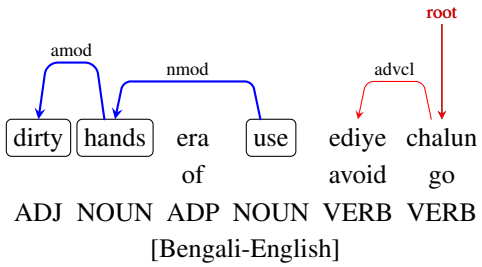
For our annotation projection algorithm, we make an assumption that similar words across languages share the same relationship with a common head. The steps for annotation projection are as follows:

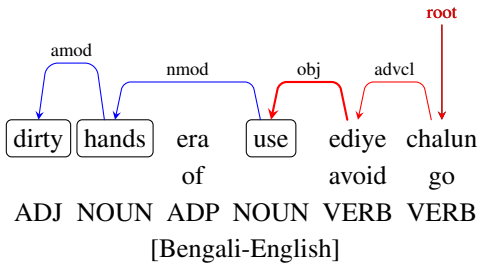
1. Project annotations from B to BE for the matching head word nodes in Bengali and its dependent Bengali nodes.



2. Project annotations from HE to BE for the matching head word nodes in English and its dependent English nodes.

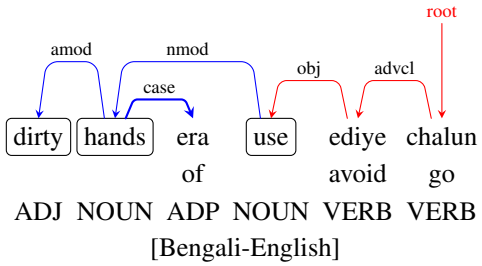
3. For each matching English dependent node in HE and BE with a Hindi head, find the aligned Bengali node in B. If the cosine similarity between the two is above a certain threshold (0.5), project annotations from B to BE.





Here, the English dependent word *use* has a Hindi head - *bache* (”save”) in HE. We cannot directly transfer annotations from HE as the Hindi word *bache* is not present in BE. Therefore, we look for the aligned Bengali dependent word in B - *byabohar* (”use”). The head of *byabohar* is *ediye* (”avoid”) which is already present in BE. We should keep in mind that the dependency label from *bache* to *use* is *obl* and that from *ediye* to *byabohar* is obj because of structural differences in Hindi and Bengali even though semantically they are similar. Next, we check for cosine similarity between *use* and *byabohar* and since it is more than the threshold of 0.5, we transfer the annotation from B to BE.

4. For each matching Bengali dependent node in B and BE with an English head, find the aligned Hindi node in HE. If the cosine similarity between the two is above a certain threshold (0.5), project annotations from HE to BE.



Hence, the annotation for the BE tree is generated by both HE (in blue) and B (in red). We get complete annotated trees for 3643 sentences. Since the sentences in BE, HE and B are essentially parallel, we get a one-to-one mapping of head nodes and do not need to introduce any dummy nodes.

	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj, obj, iobj (9.09, 21.02)	csubj, ccomp, xcomp (5.88, 11.76)		
Non-core dependents	obl, vocative, expl, dislocated (7.20, 14.41)	advcl (10.0, 10.0)	advmod discourse (6.32, 8.86)	aux cop, mark (4.47, 4.47)
Nominal dependents	nmod, appos, nummod (8.33, 9.25)	acl **(25.0, 25.0)	amod (6.34, 9.52)	det, clf case (4.89, 4.89)
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj, cc (20.51, 24.35)	fixed, flat compound (2.70, 9.45)	list, parataxis **(0,0)	orphan, goeswith reparandum **(0, 0)	punct, root dep (3.29, 4.60)

Table 6.2: Error Percentage (Head Attachment error, Labeling error) per Dependency Arc Type for 200 sentences from Syn-BE. **The marked percentages are calculated over label occurrence for < 10 tokens

6.2 Synthetic Code-Mixed Treebank Evaluation

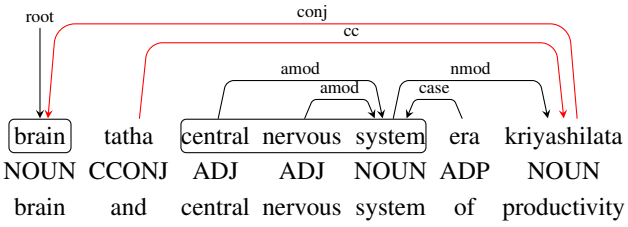
For evaluation purpose, we randomly selected 200 sentences comprising of over 2200 tokens from the code-mixed treebank and manually annotated them using UD scheme. Table 6.2 shows the error percentages per arc type. The error percentage is represented by a tuple of *attachment* and *labeling* error percentages respectively. Attachment error percentage is defined as the percentage of words for which the head (parent) node is assigned incorrectly in a dependency arc while labeling error percentage is defined as the percentage of words that are assigned either incorrect heads or correct heads with incorrect labels.

6.2.1 Head Attachment Errors

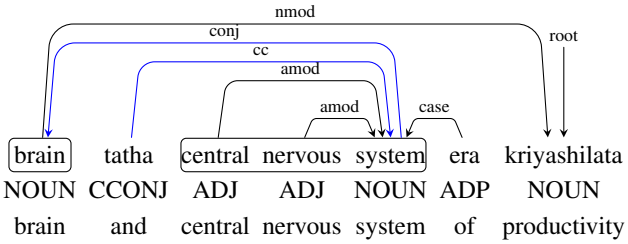
The most frequent dependency arc types leading to head attachment errors are :

- **Coordination**
  We observed the most number of head attachment errors for both of the dependency relations - *cc* and *conj* related to coordination. The prevalence of frequent code-mixing of conjuncts, sometimes

even huge chunks of sub-trees, lead to long-distance dependencies. Since the number of conjuncts are not fixed and occur without explicit markers, recognizing each of them correctly is even more challenging. We also observed attachment ambiguities at genitives and compound verbs [25].



[1a]



[1b]

" The productivity of brain and the central nervous system. "

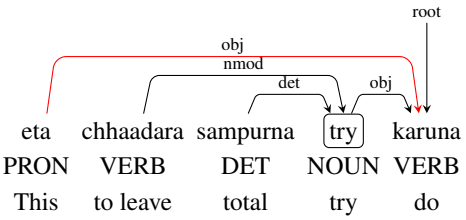
Example 1a represents the annotations from Syn-BE while 1b shows the manually annotated tree for the same phrase. In 1a, we observe that the head of *system* is marked as *kriyashilata* ("productivity") with a dependency arc '*nmod*' because of the genitive marker *era*. However, 1b illustrates that the actual head of *system* is *brain* and the two conjuncts modify *kriyashilata* ("productivity") with '*nmod*' relation.

In some cases, *o* ("and") is incorrectly POS tagged as PRON as *o* can also mean *he/she* (third person, singular) in Bengali. This obviously led to incorrect labeling of related dependency relations.

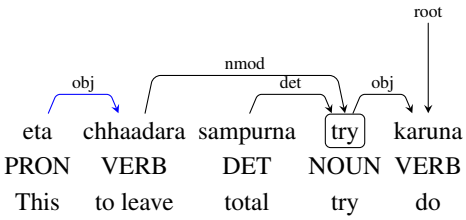
- **Nominal Core Arguments**
  In case of occurrence of multiple verbal predicates within a sentence, nominal core arguments like the subject (*nsubj*) and objects (*obj, iobj*) are often attached to the incorrect verb.

  In example 2a, *eta* ("this") is attached to the main verb *karuna* ("do") as the object ('*obj*'). However, in example 2b, the correct attachment is shown to be with the infinitive *chhadara* ("to leave") as the object ('*obj*').

- **Relative Clauses**
  The dependency label *'acl'* (adjectival clause) stands for finite and non-finite clauses that modify a nominal. A relative clause is an instance of acl, characterized by finiteness and usually omission



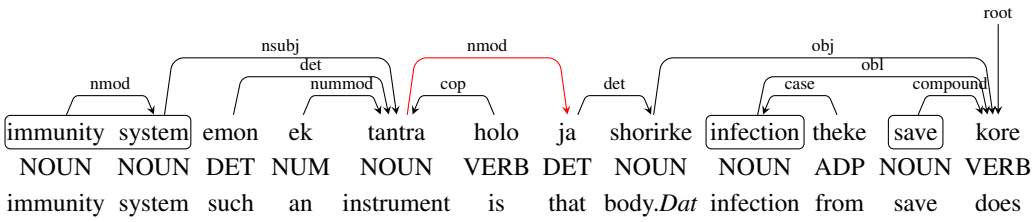
[2a]



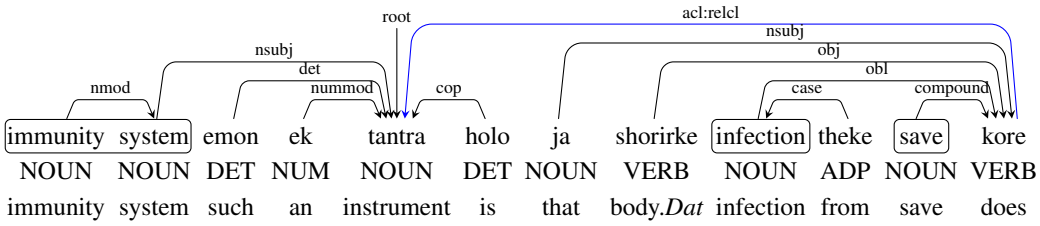
[2b]

" Try to leave this completely "

of the modified noun in the embedded clause. This is a common occurrence in Bengali and hence prevalent in Bengali-English. In Bangla relative clauses have three different orders, they can be left adjoined-placed immediately before their head noun; embedded-placed immediately after the head noun and extraposed-placed post-verbally away from the head noun [9]. Since extraposed relative clauses are separated from the head noun, this dislocation generates discontinuity in the structure and are often mislabelled. This is demonstrated in example 3a where the relative clause is not identified and hence mislabelled. Example 3b shows the correct annotation.



[3a]



[3b]

" Immunity system is one such instrument that saves the body from infection "

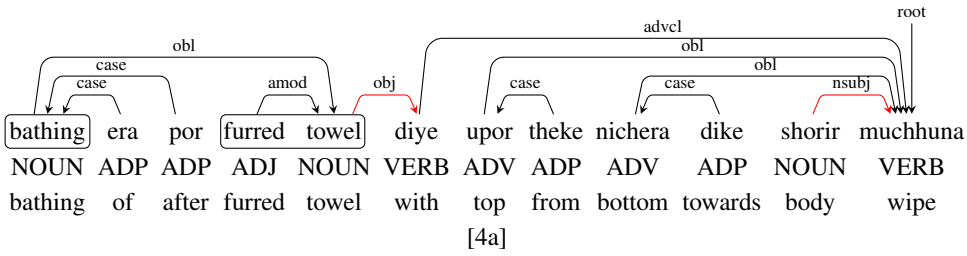
6.2.2 Labeling Errors

The most frequent dependency arc types leading to labeling errors are :

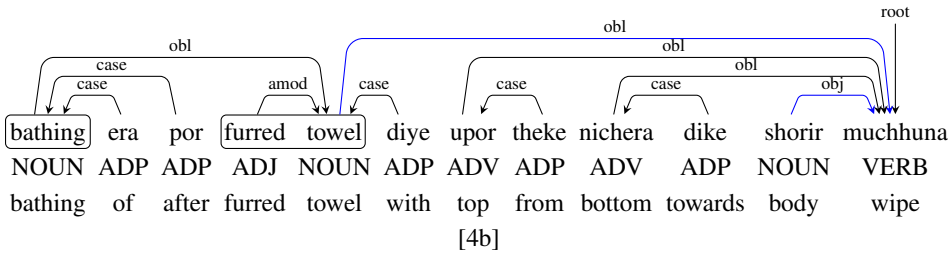
- **Nominal Arguments**

  The nominal arguments of a verbal predicate include core elements such as subjects (*'nsubj'*) and objects (*'obj', 'iobj'*) as well as non-core elements like obliques (*'obl'*) and vocative (*'vocative'*) are often incorrectly labeled because of ambiguous or missing postpositions. As Bengali is a pro-drop language, objects are often mislabeled as the subject in cases of missing pronominal subjects.

  In example 4a, *shorir* ("*body*") is attached to *muchhuna* ("*wipe*") as '*nsubj*' instead of the correct label '*obj*'. This error might be due to subject drop and lack of explicit marker for object. Also, *towel* is incorrectly marked as the object (*'obj'*) of the wrongly identified verb *diye* ("*with*") which is a postposition in this sentence.



[4a]

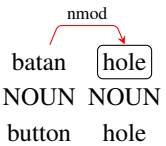


[4b]

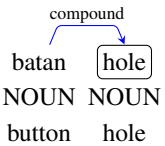
" After bathing, wipe the body from top to bottom using a furred towel "

- **Multi-word Expressions**

  We observed that many of the noun modifiers in the noun compound construction were incorrectly attached to the head as '*nmod*' instead of '*compound*' as illustrated in example 5a and 5b. This is due to the annotation inconsistencies in the Bengali monolingual treebank which we used to train our Bengali parser.



[5a]



[5b]

	Stacking Models	POS	UAS	LAS	POS	UAS	LAS
	(base) + (stacked)	Gold (LID + TRN)			Auto (LID + TRN)		
Model 1	(Bilingual) + Gold-BE	79.39	62.78	49.38	75.76	58.58	44.67
Model 2	(Trilingual) + Gold-(HE+BE)	87.43	74.42	60.04	83.73	70.04	55.95
Model 3	(Trilingual+Syn-BE) + Gold-(HE+BE)	89.63	76.24	61.41	86.21	72.28	56.81

Table 6.3: POS and Parser results of neural-stacking models for Bengali-English.

6.3 Experimental Setup

Model 3 : For our final experiment, we improve upon Model 2 (refer chapter 4) by training the source model on our Synthetic Code-Mixed Bengali-English Treebank (Syn-BE) in addition to the data from our trilingual source model. The syntactic information from our Syn-BE is transferred to the code-mixed stacked model trained on the combined gold code-mixed data belonging to Bengali-English and Hindi-English (Gold BE and HE). The hidden dimensions and learning parameters of the experiment are consistent across all the experiments as mentioned in chapter 4 . For lexical representation, we stick to cross-lingual embeddings and a shared character space between Bengali and Hindi using WX notation for the characters.

6.4 Results and Discussions

Table 6.3 summarises the results of all our models for easy comparison. Model 3, our final model utilizes our Syn-BE CM treebank and hence shows improvements over Model 2 by 1.82% UAS points, 1.37% LAS points and POS tagging accuracy by 2.2% . This improvement is satisfactory considering the errors that propagated into our Syn-BE treebank by annotating projections from automatically parsed Bengali and Hindi-English treebanks. We must also note that the the domain of Syn-BE (*health*) completely lacks social media elements present in the evaluation set and provides little help in dealing with such constructs. Also, Syn-BE only contains code-mixed sentences with Bengali matrix language and does not affect the ~2% data in the test-set with English matrix language greatly. Results with automatic detection of LID and TRN leads to an expected decrease in performance by ~3.4% for POS and ~4% and ~4.6% for UAS and LAS points.

	Nominals	Clauses	Modifier words	Function Words
Core arguments	nsubj, obj, iobj (20.60, 47.63)	csubj, ccomp, xcomp **(15, 75)		
Non-core dependents	obl, vocative, expl, dislocated (22.36, 45.34)	advcl (46.428, 57.14)	advmod discourse (39.34, 45.90)	aux cop, mark (21.87, 28.12)
Nominal dependents	nmod, appos, nummod (29.56, 38.70)	acl **(66.67, 66.67)	amod (8.33, 9.72)	det, clf case (8.65, 14.42)
Coordination	**MWE**	**Loose**	**Special**	**Other**
conj, cc (48.97, 51.02)	fixed, flat compound (8.51, 18.08)	list, parataxis **(50, 75)	orphan, goeswith reparandum **(0,0)	punct, root dep (27.17, 28.97)

Table 6.4: Error Percentage (Attachment error, Labeling error) per Dependency Arc Type. **The marked percentages are calculated over label occurrence for < 10 tokens

6.4.1 Error Analysis and Observations

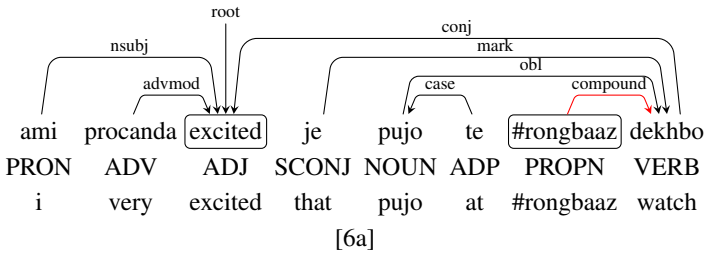
In this section, we will do an error analysis and make further observations on our parser output. As our parser is partially trained on synthetic annotations, it is important to analyze the errors that are propagated to our system.

The most frequent dependency arc types leading to parser (attachment + label) errors are :

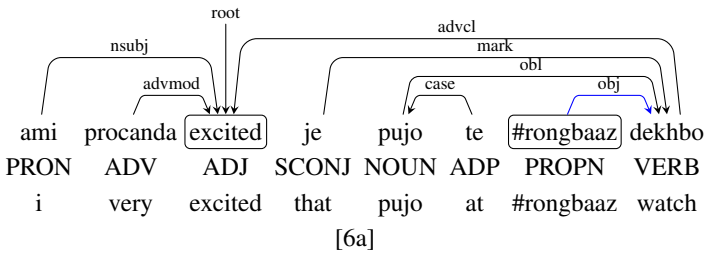
- **Coordination**
  Similar to the Syn-BE, our parser results show the most number of head attachment errors for coordination relations as well. The reasons for this kind of error is the same as mentioned earlier - high prevalence of frequent code-mixing of conjuncts, sometimes even huge chunks of sub-trees, leading to long-distance dependencies. Since the number of conjuncts are not fixed and occur without explicit markers, recognizing each of them correctly is a challenging task. However, the errors have been compounded and increased due to erroneous Syn-BE as our training data.

- **Nominal Arguments**
  Our parser makes the most labeling mistakes when dealing with nominal arguments. Missing

post-positions and subjects often lead to this confusion. However, we found a kind of error that is unique to code-mixing. Whenever a Bengali verb takes an English nominal verb argument preceding its position, our parser often mislabels it as a *compound* relation as bilingual compound verbs are abundant in our training data.



[6a]



[6a]

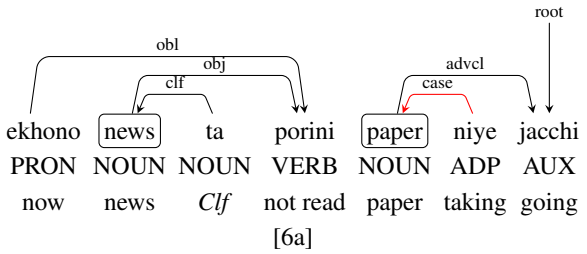
" I am very excited to watch /ronbaaz during the festival "

- **Multi-word Expressions**
  The issue of assignment of '*nmod*' to noun modifiers is witnessed in our parser results as well. We observed that many of the noun modifiers in the noun compound construction were incorrectly attached to the head as '*nmod*' due to annotation inconsistencies.

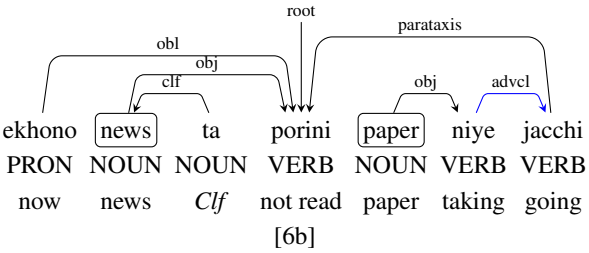
- **Adverbial Clauses and Roots**
  We noticed that our system has issues with recognizing non-finite verbs of adverbial clauses. Some non-finite verbs are also verb rooted postpositions and need to be disambiguated. For eg : *niye* can mean *"by taking"* as well as *"with"* depending on the context. In example 6a, *niye* is tagged as a '*case*' and incorrectly recognized as a postposition.

  In Bengali, the non-finite verbs of the form *a*e (*bale, kare, dhare*, etc.) are also used as finite verb [15] and this leads to the parser marking it as the main verb *'root'* instead of *'advcl'* attached to the main verb.

- **Modifier Words - Social Media Elements**
  Modifier words with dependency label *'discourse'* form a large chunk of our social media elements like hashtags, at-mentions, emoticons, hyperlinks, etc. Such relations are only present in our seed training data of Bengali-English tweets (140) and extremely rare in Hindi-English CM treebank by Bhat (2018). Our test set from twitter contains a reasonable amount of such



[6a]



[6b]

" Haven't read the news till now, leaving with the paper "

social-media elements and are often mislabelled as *'vocative'*, *'nmod'* and *'root'*. We also noticed annotation discrepancies in our manually annotated data with at-mentions being attached to the root with *'vocative'* relation instead of *'discourse'* relation for similar instances.

Hence, we need more training data dealing with these special elements.

In addition to these errors, we observed repeated errors related to dealing with words in Bengali like *ki* (PRON, PART, CCONJ), *bale* (VERB, SCONJ, PART), *o* (PRON, CCONJ, PART) which can belong to completely different POS category, if left ambiguous. Such errors bring down parser accuracy as well.

Since our test set isn't large, we did not get enough data for some labels like loose and special arguments. For cases where the token size is less than 10 for the arc type, no conclusive observation can be made.

6.5 Conclusion

In this chapter, we have presented a synthetic treebank (Syn-BE) to further improve the accuracy of our neural stacking based dependency parser. This model improves the POS tagging accuracy by 2.2% points and parser accuracy by 1.82% UAS points and 1.37% LAS points respectively.

Chapter 7

Summary and Future Work

In this chapter, we summarize the main concepts and contributions of the research work presented in this thesis. We also briefly discuss a few ideas for future work in this area of research.

7.1 Conclusion

In this thesis, we aim to efficiently parse Bengali-English code-mixed data. Mixing of two typologically different languages, English and Bengali leads to significant structural variations and compounds the challenges of parsing code-mixed data. The current NLP systems prove to be inefficient when dealing with the complexities of CM data, as they are trained on monolingual texts adhering to the language-related syntactic and orthographic norms. This leads to a dire need to develop CM NLP systems that can capture these complexities specific to code-mixing.

Because of the scarcity of quality linguistic resources for Bengali-English, we present a neural stack-based dependency parser for CM data of Bengali-English by utilizing pre-existing resources for closely related Hindi-English CM as well as monolingual treebanks for Bengali, Hindi, and English. To address the issue of scarcity of annotated resources for the Bengali-English CM pair, we present a rule-based system to computationally generate a synthetic code-mixed dataset from parallel treebanks for Bengali and English. The generated synthetic code-mixed data does not require an overhead of preprocessing for language identification and normalization. In order to utilize the synthetic code-mixed data, we project automatic annotations from Bengali and Hindi-English treebanks to Bengali-English using simple heuristics and thus creating a synthetic code-mixing treebank for Bengali and English (Syn-BE). Incorporating Syn-BE into our neural stacking parser further improves its performance.

Our final model achieves an accuracy of 89.63% for POS tagging and 76.24% UAS and 61.41% LAS points for dependency parsing. This model improves parsing results by 1.37% LAS points when compared with a stacking model utilizing pre-existing Hindi-English.

7.2 Future Work

In this thesis, we presented a system to efficiently parse Bengali-English CM, however extending the scope of our work is possible. Some of the future research directions include the following:

- The synthetically generated Bengali-English CM, Syn-BE can be extended to include code-mixing sentences with English as the matrix language in order to cover more bases. While we chose a conservative constraint-based method to select code-mixing constructions, more strategies including code-switching at clausal boundaries, translation of Bengali verbs to Bilingual verb constructions, etc. can be explored.
- The synthetic CM Bengali-English corpus generated from parallel corpora for Bengali and English can be utilized for training other CM NLP systems like machine translation, question-answering, etc which require parallel CM corpora.
- The methods and approaches for dependency parsing and code-mixed data generation can be applied to more Indian languages CM paired with English.

Related Publications

Ghosh Urmi, Dipti Misra Sharma, and Simran Khanuja. "Dependency Parser for Bengali-English Code-Mixed Data enhanced with a Synthetic Treebank." Proceedings of the 18th International Workshop on Treebanks and Linguistic Theories (TLT, SyntaxFest 2019). 2019.

Bibliography

[1] R. K. Agnihotri. Mixed codes and their acceptability. *Social psychological perspectives on second language learning*, pages 191–215, 1998.

[2] W. Ammar, G. Mulcaire, M. Ballesteros, C. Dyer, and N. A. Smith. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444, 2016.

[3] P. Auer and L. Wei. *Handbook of multilingualism and multilingual communication*, volume 5. Walter de Gruyter, 2008.

[4] U. Barman, A. Das, J. Wagner, and J. Foster. 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, 2014.

[5] R. Begum, S. Husain, A. Dhwaj, D. M. Sharma, L. Bai, and R. Sangal. Dependency annotation scheme for indian languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008.

[6] A. Bentahila and E. E. Davies. The syntax of arabic-french code-switching. *Lingua*, 59(4):301–330, 1983.

[7] I. A. Bhat, R. A. Bhat, M. Shrivastava, and D. M. Sharma. Joining hands: Exploiting monolingual treebanks for parsing of code-mixing data. *arXiv preprint arXiv:1703.10772*, 2017.

[8] I. A. Bhat, R. A. Bhat, M. Shrivastava, and D. M. Sharma. Universal dependency parsing for hindi-english code-switching. *arXiv preprint arXiv:1804.05868*, 2018.

[9] R. A. Bhat and D. M. Sharma. Non-projective structures in indian language treebanks. In *Proceedings of the 11th workshop on treebanks and linguistic theories (TLT11)*, pages 25–30, 2012.

[10] T. Bhatia and W. Ritchie. *The Handbook of Bilingualism*. 01 2008.

[11] T. K. Bhatia. English and the Vernaculars of India: Contact and Change1. *Applied Linguistics*, III(3):235–245, 10 1982.

[12] T. K. Bhatia and W. C. Ritchie. Bilingual language mixing, universal grammar, and second language acquisition. *Handbook of second language acquisition*, pages 627–688, 1996.

[13] T. Bhattacharya. Bangla; bengali. *Encyclopedia of World's Languages: Past and Present New York: WWWilson*, 2000.

[14] Ö. Çetinoğlu, S. Schulz, and N. T. Vu. Challenges of computational processing of code-switching. *arXiv preprint arXiv:1610.02213*, 2016.

[15] S. Chatterji, N. Datta, A. Dhar, B. Barik, S. Sarkar, and A. Basu. Repairing bengali verb chunks for improved bengali to hindi machine translation. In *Proceedings of the 10th Workshop on Asian Language Resources*, pages 65–74, 2012.

[16] H. Chen, Y. Zhang, and Q. Liu. Neural network for heterogeneous annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 731–741, 2016.

[17] N. Choudhary and G. N. Jha. Creating multilingual parallel corpora in indian languages. In *Language and Technology Conference*, pages 527–537. Springer, 2011.

[18] A. Das and B. Gambäck. Identifying languages at the word level in code-mixed indian social media text. In *Proceedings of the 11th International Conference on Natural Language Processing*, pages 378–387, 2014.

[19] M.-C. De Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C. D. Manning. Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–92, 2014.

[20] M.-C. De Marneffe, B. MacCartney, C. D. Manning, et al. Generating typed dependency parses from phrase structure parses. In *Lrec*, volume 6, pages 449–454, 2006.

[21] M.-C. De Marneffe and C. D. Manning. The stanford typed dependencies representation. In *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, pages 1–8. Association for Computational Linguistics, 2008.

[22] A. Dey and P. Fung. A hindi-english code-switching corpus. In *LREC*, pages 2410–2413, 2014.

[23] S. Dutta, T. Saha, S. Banerjee, and S. K. Naskar. Text normalization in code-mixed social media text. In *2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS)*, pages 378–382. IEEE, 2015.

[24] J. J. Gumperz. *Discourse Strategies*. Studies in Interactional Sociolinguistics. Cambridge University Press, 1982.

[25] S. Husain and B. Agrawal. Analyzing parser errors to improve parsing accuracy and to inform tree banking decisions. *Linguistic Issues in Language Technology*, 7(1), 2012.

[26] R. Hwa, P. Resnik, A. Weinberg, and O. Kolak. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 392–399. Association for Computational Linguistics, 2002.

[27] A. Jamatia, B. Gambäck, and A. Das. 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, 2015.

[28] G. N. Jha. The tdil program and the indian langauge corpora intitiative (ilci). In *LREC*, 2010.

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

[30] B. B. Kachru. Toward structuring code-mixing: An indian perspective. 1976.

[31] B. B. Kachru. Toward structuring code-mixing: An indian perspective. *International Journal of the Sociology of Language*, 1978(16):27–46, 1978.

[32] E. Kiperwasser and Y. Goldberg. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327, 2016.

[33] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[34] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[35] M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[37] C. Myers-Scotton. *Social motivations for codeswitching: Evidence from Africa*. Oxford University Press, 1995.

[38] K. Nelakuditi, D. S. Jitta, and R. Mamidi. Part-of-speech tagging for code mixed english-telugu social media data. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 332–342. Springer, 2016.

[39] D.-P. Nguyen and A. Dogruoz. Word level language identification in online multilingual communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 857–862, United States, 10 2013. Association for Computational Linguistics (ACL).

[40] J. Nivre, M.-C. de Marneffe, F. Ginter, Y. Goldberg, J. Hajic, C. D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira, R. Tsarfaty, and D. Zeman. Universal dependencies v1: A multilingual treebank collection. In N. C. C. Chair), K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA).

[41] F. J. Och and H. Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[42] S. Petrov, D. Das, and R. McDonald. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*, 2011.

[43] C. W. Pfaff. Constraints on language mixing: intrasentential code-switching and borrowing in spanish/english. *Language*, pages 291–318, 1979.

[44] A. Pratapa, G. Bhat, M. Choudhury, S. Sitaram, S. Dandapat, and K. Bali. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1543–1553, 2018.

[45] S. Rijhwani, R. Sequiera, M. Choudhury, K. Bali, and C. S. Maddila. Estimating code-switching on twitter with a novel generalized word-level language detection technique. In *Proceedings of the 55th Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1971–1982, Vancouver, Canada, July 2017. Association for Computational Linguistics.

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

[47] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[48] A. Sharma, S. Gupta, R. Motlani, P. Bansal, M. Srivastava, R. Mamidi, and D. M. Sharma. Shallow parsing pipeline for hindi-english code-mixed social media text. *arXiv preprint arXiv:1604.03136*, 2016.

[49] D. M. Sharma, R. Sangal, L. Bai, R. Begam, and K. Ramakrishnamacharyulu. Anncorra: Treebanks for indian languages (version–1.9). Technical report, Technical report, Language Technologies Research Center IIIT, Hyderabad, India, 2006.

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

[51] T. Solorio, E. Blair, S. Maharjan, S. Bethard, M. Diab, M. Ghoneim, A. Hawwari, F. AlGhamdi, J. Hirschberg, A. Chang, et al. 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*, pages 62–72, 2014.

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

[53] J. Tandon, H. Chaudhry, R. A. Bhat, and D. Sharma. Conversion from paninian karakas to universal dependencies for hindi dependency treebank. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 141–150, 2016.

[54] J. Tiedemann. Rediscovering annotation projection for cross-lingual parser induction. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1854–1864, 2014.

[55] D. Zeman. Reusable tagset conversion using tagset drivers. In *LREC*, volume 2008, pages 28–30, 2008.

[56] D. Zeman, J. Hajič, M. Popel, M. Potthast, M. Straka, F. Ginter, J. Nivre, and S. Petrov. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[57] Y. Zhang and D. Weiss. Stack-propagation: Improved representation learning for syntax. *arXiv preprint arXiv:1603.06598*, 2016.